AM1806
Sitara ARM Microprocessor

Technical Reference Manual



Literature Number: SPRUH83C
April 2013–Revised September 2016

Contents

Preface **72**

1 Overview **73**
- 1.1 Introduction 74
- 1.2 ARM Subsystem 74

2 ARM Subsystem **75**
- 2.1 Introduction 76
- 2.2 Operating States/Modes 77
- 2.3 Processor Status Registers 77
- 2.4 Exceptions and Exception Vectors 78
- 2.5 The 16-BIS/32-BIS Concept 79
- 2.6 16-BIS/32-BIS Advantages 79
- 2.7 Co-Processor 15 (CP15) 80
  - 2.7.1 Addresses in an ARM926EJ-S System 80
  - 2.7.2 Memory Management Unit (MMU) 80
  - 2.7.3 Caches and Write Buffer 81

3 System Interconnect **82**
- 3.1 Introduction 83
- 3.2 System Interconnect Block Diagram 84

4 System Memory **85**
- 4.1 Introduction 86
- 4.2 ARM Memories 86
- 4.3 Peripherals 86

5 Memory Protection Unit (MPU) **87**
- 5.1 Introduction 88
  - 5.1.1 Purpose of the MPU 88
  - 5.1.2 Features 88
  - 5.1.3 Block Diagram 88
  - 5.1.4 MPU Default Configuration 89
- 5.2 Architecture 89
  - 5.2.1 Privilege Levels 89
  - 5.2.2 Memory Protection Ranges 90
  - 5.2.3 Permission Structures 91
  - 5.2.4 Protection Check 92
  - 5.2.5 MPU Register Protection 92
  - 5.2.6 Invalid Accesses and Exceptions 92
  - 5.2.7 Reset Considerations 92
  - 5.2.8 Interrupt Support 93
  - 5.2.9 Emulation Considerations 93
- 5.3 MPU Registers 94
  - 5.3.1 Revision Identification Register (REVID) 96
  - 5.3.2 Configuration Register (CONFIG) 96
  - 5.3.3 Interrupt Raw Status/Set Register (IRAWSTAT) 97
  - 5.3.4 Interrupt Enable Status/Clear Register (IENSTAT) 98
  - 5.3.5 Interrupt Enable Set Register (IENSET) 99

Copyright © 2013–2016, Texas Instruments Incorporated

5.3.6 Interrupt Enable Clear Register (IENCLR) 99
5.3.7 Fixed Range Start Address Register (FXD_MPSAR) 100
5.3.8 Fixed Range End Address Register (FXD_MPEAR) 100
5.3.9 Fixed Range Memory Protection Page Attributes Register (FXD_MPPA) 101
5.3.10 Programmable Range *n* Start Address Registers (PROG*n*_MPSAR) 102
5.3.11 Programmable Range *n* End Address Registers (PROG*n*_MPEAR) 103
5.3.12 Programmable Range *n* Memory Protection Page Attributes Register (PROG*n*_MPPA) 104
5.3.13 Fault Address Register (FLTADDRR) 105
5.3.14 Fault Status Register (FLTSTAT) 106
5.3.15 Fault Clear Register (FLTCLR) 107

6 Device Clocking 108
6.1 Overview 109
6.2 Frequency Flexibility 111
6.3 Peripheral Clocking 112
6.3.1 USB Clocking 112
6.3.2 DDR2/mDDR Memory Controller Clocking 113
6.3.3 EMIFA Clocking 115
6.3.4 uPP Clocking 116
6.3.5 McASP Clocking 117
6.3.6 I/O Domains 118

7 Phase-Locked Loop Controller (PLLC) 119
7.1 Introduction 120
7.2 PLL Controllers 120
7.2.1 Device Clock Generation 122
7.2.2 Steps for Programming the PLLs 123
7.3 PLLC Registers 125
7.3.1 PLLC0 Revision Identification Register (REVID) 126
7.3.2 PLLC1 Revision Identification Register (REVID) 127
7.3.3 Reset Type Status Register (RSTYPE) 127
7.3.4 PLLC0 Reset Control Register (RSCTRL) 128
7.3.5 PLLC0 Control Register (PLLCTL) 129
7.3.6 PLLC1 Control Register (PLLCTL) 130
7.3.7 PLLC0 OBSCLK Select Register (OCSEL) 131
7.3.8 PLLC1 OBSCLK Select Register (OCSEL) 132
7.3.9 PLL Multiplier Control Register (PLLM) 133
7.3.10 PLLC0 Pre-Divider Control Register (PREDIV) 133
7.3.11 PLLC0 Divider 1 Register (PLLDIV1) 134
7.3.12 PLLC1 Divider 1 Register (PLLDIV1) 134
7.3.13 PLLC0 Divider 2 Register (PLLDIV2) 135
7.3.14 PLLC1 Divider 2 Register (PLLDIV2) 135
7.3.15 PLLC0 Divider 3 Register (PLLDIV3) 136
7.3.16 PLLC1 Divider 3 Register (PLLDIV3) 136
7.3.17 PLLC0 Divider 4 Register (PLLDIV4) 137
7.3.18 PLLC0 Divider 5 Register (PLLDIV5) 137
7.3.19 PLLC0 Divider 6 Register (PLLDIV6) 138
7.3.20 PLLC0 Divider 7 Register (PLLDIV7) 138
7.3.21 PLLC0 Oscillator Divider 1 Register (OSCDIV) 139
7.3.22 PLLC1 Oscillator Divider 1 Register (OSCDIV) 139
7.3.23 PLL Post-Divider Control Register (POSTDIV) 140
7.3.24 PLL Controller Command Register (PLLCMD) 140
7.3.25 PLL Controller Status Register (PLLSTAT) 141
7.3.26 PLLC0 Clock Align Control Register (ALNCTL) 142
7.3.27 PLLC1 Clock Align Control Register (ALNCTL) 143

Copyright © 2013–2016, Texas Instruments Incorporated

7.3.28 PLLC0 PLLDIV Ratio Change Status Register (DCHANGE) 144
7.3.29 PLLC1 PLLDIV Ratio Change Status Register (DCHANGE) 145
7.3.30 PLLC0 Clock Enable Control Register (CKEN) 146
7.3.31 PLLC1 Clock Enable Control Register (CKEN) 146
7.3.32 PLLC0 Clock Status Register (CKSTAT) 147
7.3.33 PLLC1 Clock Status Register (CKSTAT) 148
7.3.34 PLLC0 SYSCLK Status Register (SYSTAT) 149
7.3.35 PLLC1 SYSCLK Status Register (SYSTAT) 150
7.3.36 Emulation Performance Counter 0 Register (EMUCNT0) 151
7.3.37 Emulation Performance Counter 1 Register (EMUCNT1) 151

8 Power and Sleep Controller (PSC) 152
8.1 Introduction 153
8.2 Power Domain and Module Topology 153
8.2.1 Power Domain States 155
8.2.2 Module States 156
8.3 Executing State Transitions 158
8.3.1 Power Domain State Transitions 158
8.3.2 Module State Transitions 158
8.4 IcePick Emulation Support in the PSC 159
8.5 PSC Interrupts 159
8.5.1 Interrupt Events 159
8.5.2 Interrupt Registers 160
8.5.3 Interrupt Handling 161
8.6 PSC Registers 162
8.6.1 Revision Identification Register (REVID) 163
8.6.2 Interrupt Evaluation Register (INTEVAL) 163
8.6.3 PSC0 Module Error Pending Register 0 (modules 0-15) (MERRPR0) 164
8.6.4 PSC1 Module Error Pending Register 0 (modules 0-31) (MERRPR0) 164
8.6.5 PSC0 Module Error Clear Register 0 (modules 0-15) (MERRCR0) 165
8.6.6 PSC1 Module Error Clear Register 0 (modules 0-31) (MERRCR0) 165
8.6.7 Power Error Pending Register (PERRPR) 166
8.6.8 Power Error Clear Register (PERRCR) 166
8.6.9 Power Domain Transition Command Register (PTCMD) 167
8.6.10 Power Domain Transition Status Register (PTSTAT) 168
8.6.11 Power Domain 0 Status Register (PDSTAT0) 169
8.6.12 Power Domain 1 Status Register (PDSTAT1) 170
8.6.13 Power Domain 0 Control Register (PDCTL0) 171
8.6.14 Power Domain 1 Control Register (PDCTL1) 172
8.6.15 Power Domain 0 Configuration Register (PDCFG0) 173
8.6.16 Power Domain 1 Configuration Register (PDCFG1) 174
8.6.17 Module Status *n* Register (MDSTAT*n*) 175
8.6.18 PSC0 Module Control *n* Register (modules 0-15) (MDCTL*n*) 176
8.6.19 PSC1 Module Control *n* Register (modules 0-31) (MDCTL*n*) 177

9 Power Management 178
9.1 Introduction 179
9.2 Power Consumption Overview 179
9.3 PSC and PLLC Overview 179
9.4 Features 180
9.5 Clock Management 181
9.5.1 Module Clock ON/OFF 181
9.5.2 Module Clock Frequency Scaling 181
9.5.3 PLL Bypass and Power Down 181
9.6 ARM Sleep Mode Management 182

Copyright © 2013–2016, Texas Instruments Incorporated

9.6.1 ARM Wait-For-Interrupt Sleep Mode 182
9.6.2 ARM Clock OFF 183
9.6.3 ARM Subsystem Clock ON 183
9.7 RTC-Only Mode 184
9.8 Dynamic Voltage and Frequency Scaling (DVFS) 184
9.8.1 Frequency Scaling Considerations 184
9.8.2 Voltage Scaling Considerations 185
9.9 Deep Sleep Mode 185
9.9.1 Entering/Exiting Deep Sleep Mode Using Externally Controlled Wake-Up 186
9.9.2 Entering/Exiting Deep Sleep Mode Using RTC Controlled Wake-Up 187
9.9.3 Deep Sleep Sequence 188
9.9.4 Entering/Exiting Deep Sleep Mode Using Software Handshaking 189
9.10 Additional Peripheral Power Management Considerations 189
9.10.1 USB PHY Power Down Control 189
9.10.2 DDR2/mDDR Memory Controller Clock Gating and Self-Refresh Mode 190
9.10.3 LVCMOS I/O Buffer Receiver Disable 190
9.10.4 Pull-Up/Pull-Down Disable 190
10 System Configuration (SYSCFG) Module 191
10.1 Introduction 192
10.2 Protection 192
10.2.1 Privilege Mode Protection 192
10.2.2 Kicker Mechanism Protection 193
10.3 Master Priority Control 193
10.4 Interrupt Support 194
10.4.1 Interrupt Events and Requests 194
10.4.2 Interrupt Multiplexing 195
10.5 SYSCFG Registers 195
10.5.1 Revision Identification Register (REVID) 196
10.5.2 Device Identification Register 0 (DEVIDR0) 197
10.5.3 Boot Configuration Register (BOOTCFG) 197
10.5.4 Chip Revision Identification Register (CHIPREVIDR) 197
10.5.5 Kick Registers (KICK0R-KICK1R) 199
10.5.6 Host 0 Configuration Register (HOST0CFG) 200
10.5.7 Interrupt Registers 201
10.5.8 Fault Registers 204
10.5.9 Master Priority Registers (MSTPRI0-MSTPRI2) 206
10.5.10 Pin Multiplexing Control Registers (PINMUX0-PINMUX19) 209
10.5.11 Suspend Source Register (SUSPSRC) 249
10.5.12 Chip Signal Register (CHIPSIG) 252
10.5.13 Chip Signal Clear Register (CHIPSIG_CLR) 253
10.5.14 Chip Configuration 0 Register (CFGCHIP0) 254
10.5.15 Chip Configuration 1 Register (CFGCHIP1) 255
10.5.16 Chip Configuration 2 Register (CFGCHIP2) 257
10.5.17 Chip Configuration 3 Register (CFGCHIP3) 259
10.5.18 Chip Configuration 4 Register (CFGCHIP4) 260
10.5.19 VTP I/O Control Register (VTPIO_CTL) 261
10.5.20 DDR Slew Register (DDR_SLEW) 263
10.5.21 Deep Sleep Register (DEEPSLEEP) 264
10.5.22 Pullup/Pulldown Enable Register (PUPD_ENA) 265
10.5.23 Pullup/Pulldown Select Register (PUPD_SEL) 265
10.5.24 RXACTIVE Control Register (RXACTIVE) 267
11 ARM Interrupt Controller (AINTC) 268
11.1 Introduction 269

Copyright © 2013–2016, Texas Instruments Incorporated

11.2 Interrupt Mapping 269
11.3 AINTC Methodology 272
11.3.1 Interrupt Processing 272
11.3.2 Interrupt Enabling 273
11.3.3 Interrupt Status Checking 273
11.3.4 Interrupt Channel Mapping 273
11.3.5 Host Interrupt Mapping Interrupts 273
11.3.6 Interrupt Prioritization 274
11.3.7 Interrupt Nesting 274
11.3.8 Interrupt Vectorization 275
11.3.9 Interrupt Status Clearing 276
11.3.10 Interrupt Disabling 276
11.4 AINTC Registers 276
11.4.1 Revision Identification Register (REVID) 277
11.4.2 Control Register (CR) 278
11.4.3 Global Enable Register (GER) 279
11.4.4 Global Nesting Level Register (GNLR) 279
11.4.5 System Interrupt Status Indexed Set Register (SISR) 280
11.4.6 System Interrupt Status Indexed Clear Register (SICR) 280
11.4.7 System Interrupt Enable Indexed Set Register (EISR) 281
11.4.8 System Interrupt Enable Indexed Clear Register (EICR) 281
11.4.9 Host Interrupt Enable Indexed Set Register (HIEISR) 282
11.4.10 Host Interrupt Enable Indexed Clear Register (HIEICR) 282
11.4.11 Vector Base Register (VBR) 283
11.4.12 Vector Size Register (VSR) 283
11.4.13 Vector Null Register (VNR) 284
11.4.14 Global Prioritized Index Register (GPIR) 284
11.4.15 Global Prioritized Vector Register (GPVR) 285
11.4.16 System Interrupt Status Raw/Set Register 1 (SRSR1) 285
11.4.17 System Interrupt Status Raw/Set Register 2 (SRSR2) 286
11.4.18 System Interrupt Status Raw/Set Register 3 (SRSR3) 286
11.4.19 System Interrupt Status Raw/Set Register 4 (SRSR4) 287
11.4.20 System Interrupt Status Enabled/Clear Register 1 (SECR1) 287
11.4.21 System Interrupt Status Enabled/Clear Register 2 (SECR2) 288
11.4.22 System Interrupt Status Enabled/Clear Register 3 (SECR3) 288
11.4.23 System Interrupt Status Enabled/Clear Register 4 (SECR4) 289
11.4.24 System Interrupt Enable Set Register 1 (ESR1) 289
11.4.25 System Interrupt Enable Set Register 2 (ESR2) 290
11.4.26 System Interrupt Enable Set Register 3 (ESR3) 290
11.4.27 System Interrupt Enable Set Register 4 (ESR4) 291
11.4.28 System Interrupt Enable Clear Register 1 (ECR1) 291
11.4.29 System Interrupt Enable Clear Register 2 (ECR2) 292
11.4.30 System Interrupt Enable Clear Register 3 (ECR3) 292
11.4.31 System Interrupt Enable Clear Register 4 (ECR4) 293
11.4.32 Channel Map Registers (CMR0-CMR25) 293
11.4.33 Host Interrupt Prioritized Index Register 1 (HIPIR1) 294
11.4.34 Host Interrupt Prioritized Index Register 2 (HIPIR2) 294
11.4.35 Host Interrupt Nesting Level Register 1 (HINLR1) 295
11.4.36 Host Interrupt Nesting Level Register 2 (HINLR2) 295
11.4.37 Host Interrupt Enable Register (HIER) 296
11.4.38 Host Interrupt Prioritized Vector Register 1 (HIPVR1) 297
11.4.39 Host Interrupt Prioritized Vector Register 2 (HIPVR2) 297
12 Boot Considerations 298

Copyright © 2013–2016, Texas Instruments Incorporated

12.1 Introduction 299
13 Programmable Real-Time Unit Subsystem (PRUSS) 300
13.1 Overview 301
13.2 Description 303
13.3 Constants Table 304
13.4 PRU Module Interface 305
13.4.1 Event Out Mapping (R31): PRU System Events 305
13.4.2 Status Mapping (R31): Interrupt Events Input 305
13.4.3 General Purpose Inputs (R31) 305
13.4.4 General Purpose Outputs (R30) 305
13.5 Instruction Set 306
13.6 Instruction Formats 309
13.7 PRU Interrupt Controller 327
13.7.1 Introduction 327
13.7.2 Interrupt Mapping 327
13.7.3 PRUSS System Events 328
13.7.4 ARM and DSP Interrupt Controller Mapping 330
13.7.5 INTC Methodology 331
13.8 Registers 334
13.8.1 PRUSS Memory Map 335
13.8.2 INTC Registers 342
14 DDR2/mDDR Memory Controller 353
14.1 Introduction 354
14.1.1 Purpose of the Peripheral 354
14.1.2 Features 354
14.1.3 Functional Block Diagram 355
14.1.4 Supported Use Case Statement 355
14.1.5 Industry Standard(s) Compliance Statement 355
14.2 Architecture 356
14.2.1 Clock Control 356
14.2.2 Signal Descriptions 357
14.2.3 Protocol Description(s) 358
14.2.4 Memory Width and Byte Alignment 366
14.2.5 Address Mapping 367
14.2.6 DDR2/mDDR Memory Controller Interface 372
14.2.7 Refresh Scheduling 375
14.2.8 Self-Refresh Mode 375
14.2.9 Partial Array Self Refresh for Mobile DDR 376
14.2.10 Power-Down Mode 376
14.2.11 Reset Considerations 377
14.2.12 VTP IO Buffer Calibration 378
14.2.13 Auto-Initialization Sequence 378
14.2.14 Interrupt Support 381
14.2.15 DMA Event Support 381
14.2.16 Power Management 382
14.2.17 Emulation Considerations 383
14.3 Supported Use Cases 384
14.4 Registers 389
14.4.1 SDRAM Status Register (SDRSTAT) 390
14.4.2 SDRAM Configuration Register (SDCR) 391
14.4.3 SDRAM Refresh Control Register (SDRCR) 394
14.4.4 SDRAM Timing Register 1 (SDTIMR1) 395
14.4.5 SDRAM Timing Register 2 (SDTIMR2) 396

Copyright © 2013–2016, Texas Instruments Incorporated

14.4.6 SDRAM Configuration Register 2 (SDCR2) ... 397
14.4.7 Peripheral Bus Burst Priority Register (PBBPR) ... 398
14.4.8 Performance Counter 1 Register (PC1) ... 399
14.4.9 Performance Counter 2 Register (PC2) ... 399
14.4.10 Performance Counter Configuration Register (PCC) ... 400
14.4.11 Performance Counter Master Region Select Register (PCMRS) ... 402
14.4.12 DDR PHY Reset Control Register (DRPYRCR) ... 403
14.4.13 Interrupt Raw Register (IRR) ... 404
14.4.14 Interrupt Masked Register (IMR) ... 404
14.4.15 Interrupt Mask Set Register (IMSR) ... 405
14.4.16 Interrupt Mask Clear Register (IMCR) ... 406
14.4.17 DDR PHY Control Register (DRPYC1R) ... 407

15 Enhanced Capture (eCAP) Module ... 408

15.1 Introduction ... 409
15.1.1 Purpose of the Peripheral ... 409
15.1.2 Features ... 409
15.2 Architecture ... 410
15.2.1 Capture and APWM Operating Mode ... 411
15.2.2 Capture Mode Description ... 412
15.3 Applications ... 419
15.3.1 Absolute Time-Stamp Operation Rising Edge Trigger Example ... 420
15.3.2 Absolute Time-Stamp Operation Rising and Falling Edge Trigger Example ... 422
15.3.3 Time Difference (Delta) Operation Rising Edge Trigger Example ... 424
15.3.4 Time Difference (Delta) Operation Rising and Falling Edge Trigger Example ... 426
15.3.5 Application of the APWM Mode ... 428
15.4 Registers ... 435
15.4.1 Time-Stamp Counter Register (TSCTR) ... 435
15.4.2 Counter Phase Control Register (CTRPHS) ... 436
15.4.3 Capture 1 Register (CAP1) ... 436
15.4.4 Capture 2 Register (CAP2) ... 437
15.4.5 Capture 3 Register (CAP3) ... 437
15.4.6 Capture 4 Register (CAP4) ... 438
15.4.7 ECAP Control Register 1 (ECCTL1) ... 438
15.4.8 ECAP Control Register 2 (ECCTL2) ... 440
15.4.9 ECAP Interrupt Enable Register (ECEINT) ... 441
15.4.10 ECAP Interrupt Flag Register (ECFLG) ... 443
15.4.11 ECAP Interrupt Clear Register (ECCLR) ... 444
15.4.12 ECAP Interrupt Forcing Register (ECFRC) ... 445
15.4.13 Revision ID Register (REVID) ... 446

16 Enhanced High-Resolution Pulse-Width Modulator (eHRPWM) ... 447

16.1 Introduction ... 448
16.1.1 Introduction ... 448
16.1.2 Submodule Overview ... 448
16.1.3 Register Mapping ... 452
16.2 Architecture ... 453
16.2.1 Overview ... 453
16.2.2 Proper Interrupt Initialization Procedure ... 456
16.2.3 Time-Base (TB) Submodule ... 457
16.2.4 Counter-Compare (CC) Submodule ... 466
16.2.5 Action-Qualifier (AQ) Submodule ... 471
16.2.6 Dead-Band Generator (DB) Submodule ... 489
16.2.7 PWM-Chopper (PC) Submodule ... 493
16.2.8 Trip-Zone (TZ) Submodule ... 497

Copyright © 2013–2016, Texas Instruments Incorporated

16.2.9 Event-Trigger (ET) Submodule 501
16.2.10 High-Resolution PWM (HRPWM) Submodule 505
16.3 Applications to Power Topologies 512
16.3.1 Overview of Multiple Modules 512
16.3.2 Key Configuration Capabilities 513
16.3.3 Controlling Multiple Buck Converters With Independent Frequencies 514
16.3.4 Controlling Multiple Buck Converters With Same Frequencies 517
16.3.5 Controlling Multiple Half H-Bridge (HHB) Converters 520
16.3.6 Controlling Dual 3-Phase Inverters for Motors (ACI and PMSM) 523
16.3.7 Practical Applications Using Phase Control Between PWM Modules 527
16.3.8 Controlling a 3-Phase Interleaved DC/DC Converter 528
16.3.9 Controlling Zero Voltage Switched Full Bridge (ZVSFB) Converter 533
16.4 Registers 536
16.4.1 Time-Base Submodule Registers 536
16.4.2 Counter-Compare Submodule Registers 540
16.4.3 Action-Qualifier Submodule Registers 543
16.4.4 Dead-Band Generator Submodule Registers 547
16.4.5 PWM-Chopper Submodule Register 550
16.4.6 Trip-Zone Submodule Registers 551
16.4.7 Event-Trigger Submodule Registers 555
16.4.8 High-Resolution PWM Submodule Registers 558
17 Enhanced Direct Memory Access (EDMA3) Controller 561
17.1 Introduction 562
17.1.1 Overview 562
17.1.2 Features 562
17.1.3 Functional Block Diagram 565
17.1.4 Terminology Used in This Document 565
17.2 Architecture 567
17.2.1 Functional Overview 567
17.2.2 Types of EDMA3 Transfers 570
17.2.3 Parameter RAM (PaRAM) 573
17.2.4 Initiating a DMA Transfer 583
17.2.5 Completion of a DMA Transfer 586
17.2.6 Event, Channel, and PaRAM Mapping 587
17.2.7 EDMA3 Channel Controller Regions 590
17.2.8 Chaining EDMA3 Channels 592
17.2.9 EDMA3 Interrupts 592
17.2.10 Event Queue(s) 599
17.2.11 EDMA3 Transfer Controller (EDMA3TC) 601
17.2.12 Event Dataflow 604
17.2.13 EDMA3 Prioritization 605
17.2.14 EDMA3CC and EDMA3TC Performance and System Considerations 607
17.2.15 EDMA3 Operating Frequency (Clock Control) 608
17.2.16 Reset Considerations 608
17.2.17 Power Management 608
17.2.18 Emulation Considerations 609
17.3 Transfer Examples 609
17.3.1 Block Move Example 609
17.3.2 Subframe Extraction Example 611
17.3.3 Data Sorting Example 612
17.3.4 Peripheral Servicing Example 614
17.4 Registers 626
17.4.1 Parameter RAM (PaRAM) Entries 626

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2 EDMA3 Channel Controller (EDMA3CC) Registers ... 633
17.4.3 EDMA3 Transfer Controller (EDMA3TC) Registers ... 672
17.5 Tips ... 693
17.5.1 Debug Checklist ... 693
17.5.2 Miscellaneous Programming/Debug Tips ... 694
17.6 Setting Up a Transfer ... 695
18 External Memory Interface A (EMIFA) ... 696
18.1 Introduction ... 697
18.1.1 Purpose of the Peripheral ... 697
18.1.2 Features ... 697
18.1.3 Functional Block Diagram ... 697
18.2 Architecture ... 697
18.2.1 Clock Control ... 698
18.2.2 EMIFA Requests ... 698
18.2.3 Pin Descriptions ... 698
18.2.4 SDRAM Controller and Interface ... 700
18.2.5 Asynchronous Controller and Interface ... 712
18.2.6 Data Bus Parking ... 731
18.2.7 Reset and Initialization Considerations ... 731
18.2.8 Interrupt Support ... 732
18.2.9 EDMA Event Support ... 733
18.2.10 Pin Multiplexing ... 733
18.2.11 Memory Map ... 733
18.2.12 Priority and Arbitration ... 734
18.2.13 System Considerations ... 735
18.2.14 Power Management ... 736
18.2.15 Emulation Considerations ... 737
18.3 Example Configuration ... 738
18.3.1 Hardware Interface ... 738
18.3.2 Software Configuration ... 738
18.4 Registers ... 760
18.4.1 Module ID Register (MIDR) ... 761
18.4.2 Asynchronous Wait Cycle Configuration Register (AWCC) ... 761
18.4.3 SDRAM Configuration Register (SDCR) ... 763
18.4.4 SDRAM Refresh Control Register (SDRCR) ... 765
18.4.5 Asynchronous *n* Configuration Registers (CE2CFG-CE5CFG) ... 766
18.4.6 SDRAM Timing Register (SDTIMR) ... 768
18.4.7 SDRAM Self Refresh Exit Timing Register (SDSRETR) ... 769
18.4.8 EMIFA Interrupt Raw Register (INTRAW) ... 770
18.4.9 EMIFA Interrupt Masked Register (INTMSK) ... 771
18.4.10 EMIFA Interrupt Mask Set Register (INTMSKSET) ... 772
18.4.11 EMIFA Interrupt Mask Clear Register (INTMSKCLR) ... 773
18.4.12 NAND Flash Control Register (NANDFCR) ... 774
18.4.13 NAND Flash Status Register (NANDFSR) ... 776
18.4.14 NAND Flash *n* ECC Registers (NANDF1ECC-NANDF4ECC) ... 777
18.4.15 NAND Flash 4-Bit ECC LOAD Register (NAND4BITECCLOAD) ... 778
18.4.16 NAND Flash 4-Bit ECC Register 1 (NAND4BITECC1) ... 779
18.4.17 NAND Flash 4-Bit ECC Register 2 (NAND4BITECC2) ... 779
18.4.18 NAND Flash 4-Bit ECC Register 3 (NAND4BITECC3) ... 780
18.4.19 NAND Flash 4-Bit ECC Register 4 (NAND4BITECC4) ... 780
18.4.20 NAND Flash 4-Bit ECC Error Address Register 1 (NANDERRADD1) ... 781
18.4.21 NAND Flash 4-Bit ECC Error Address Register 2 (NANDERRADD2) ... 781
18.4.22 NAND Flash 4-Bit ECC Error Value Register 1 (NANDERRVAL1) ... 782

Copyright © 2013–2016, Texas Instruments Incorporated

18.4.23 NAND Flash 4-Bit ECC Error Value Register 2 (NANDERRVAL2) ... 782

19 General-Purpose Input/Output (GPIO) ... 783

19.1 Introduction ... 784
19.1.1 Purpose of the Peripheral ... 784
19.1.2 Features ... 784
19.1.3 Functional Block Diagram ... 784
19.1.4 Industry Standard(s) Compliance Statement ... 784
19.2 Architecture ... 785
19.2.1 Clock Control ... 785
19.2.2 Signal Descriptions ... 785
19.2.3 Pin Multiplexing ... 785
19.2.4 Endianness Considerations ... 785
19.2.5 GPIO Register Structure ... 786
19.2.6 Using a GPIO Signal as an Output ... 789
19.2.7 Using a GPIO Signal as an Input ... 790
19.2.8 Reset Considerations ... 790
19.2.9 Initialization ... 791
19.2.10 Interrupt Support ... 791
19.2.11 EDMA Event Support ... 792
19.2.12 Power Management ... 792
19.2.13 Emulation Considerations ... 792
19.3 Registers ... 793
19.3.1 Revision ID Register (REVID) ... 794
19.3.2 GPIO Interrupt Per-Bank Enable Register (BINTEN) ... 795
19.3.3 GPIO Direction Registers (DIR*n*) ... 796
19.3.4 GPIO Output Data Registers (OUT_DATA*n*) ... 798
19.3.5 GPIO Set Data Registers (SET_DATA*n*) ... 800
19.3.6 GPIO Clear Data Registers (CLR_DATA*n*) ... 802
19.3.7 GPIO Input Data Registers (IN_DATA*n*) ... 804
19.3.8 GPIO Set Rising Edge Interrupt Registers (SET_RIS_TRIG*n*) ... 806
19.3.9 GPIO Clear Rising Edge Interrupt Registers (CLR_RIS_TRIG*n*) ... 808
19.3.10 GPIO Set Falling Edge Interrupt Registers (SET_FAL_TRIG*n*) ... 810
19.3.11 GPIO Clear Falling Edge Interrupt Registers (CLR_FAL_TRIG*n*) ... 812
19.3.12 GPIO Interrupt Status Registers (INTSTAT*n*) ... 814

20 Host Port Interface (HPI) ... 816

20.1 Introduction ... 817
20.1.1 Purpose of the Peripheral ... 817
20.1.2 Features ... 817
20.1.3 Functional Block Diagram ... 818
20.1.4 Industry Standard(s) Compliance Statement ... 819
20.1.5 Terminology Used in This Document ... 819
20.2 Architecture ... 820
20.2.1 Clock Control ... 820
20.2.2 Memory Map ... 820
20.2.3 Signal Descriptions ... 820
20.2.4 Pin Multiplexing and General-Purpose I/O Control Blocks ... 821
20.2.5 Protocol Description ... 822
20.2.6 Operation ... 822
20.2.7 Reset Considerations ... 837
20.2.8 Initialization ... 837
20.2.9 Interrupt Support ... 838
20.2.10 EDMA Event Support ... 839
20.2.11 Power Management ... 839

Copyright © 2013–2016, Texas Instruments Incorporated

20.2.12 Emulation Considerations 840
20.3 Registers 840
20.3.1 Revision Identification Register (REVID) 841
20.3.2 Power and Emulation Management Register (PWREMU_MGMT) 841
20.3.3 GPIO Enable Register (GPIO_EN) 842
20.3.4 GPIO Direction 1 Register (GPIO_DIR1) 843
20.3.5 GPIO Data 1 Register (GPIO_DAT1) 843
20.3.6 GPIO Direction 2 Register (GPIO_DIR2) 844
20.3.7 GPIO Data 2 Register (GPIO_DAT2) 845
20.3.8 Host Port Interface Control Register (HPIC) 846
20.3.9 Host Port Interface Write Address Register (HPIAW) 848
20.3.10 Host Port Interface Read Address Register (HPIAR) 848

21 Inter-Integrated Circuit (I2C) Module 849

21.1 Introduction 850
21.1.1 Purpose of the Peripheral 850
21.1.2 Features 850
21.1.3 Functional Block Diagram 851
21.1.4 Industry Standard(s) Compliance Statement 851
21.2 Architecture 852
21.2.1 Bus Structure 852
21.2.2 Clock Generation 853
21.2.3 Clock Synchronization 854
21.2.4 Signal Descriptions 854
21.2.5 START and STOP Conditions 855
21.2.6 Serial Data Formats 856
21.2.7 Operating Modes 858
21.2.8 NACK Bit Generation 859
21.2.9 Arbitration 860
21.2.10 Reset Considerations 861
21.2.11 Initialization 861
21.2.12 Interrupt Support 862
21.2.13 DMA Events Generated by the I2C Peripheral 863
21.2.14 Power Management 863
21.2.15 Emulation Considerations 863
21.3 Registers 864
21.3.1 I2C Own Address Register (ICOAR) 865
21.3.2 I2C Interrupt Mask Register (ICIMR) 866
21.3.3 I2C Interrupt Status Register (ICSTR) 867
21.3.4 I2C Clock Divider Registers (ICCLKL and ICCLKH) 870
21.3.5 I2C Data Count Register (ICCNT) 871
21.3.6 I2C Data Receive Register (ICDRR) 872
21.3.7 I2C Slave Address Register (ICSAR) 873
21.3.8 I2C Data Transmit Register (ICDXR) 874
21.3.9 I2C Mode Register (ICMDR) 875
21.3.10 I2C Interrupt Vector Register (ICIVR) 879
21.3.11 I2C Extended Mode Register (ICEMDR) 880
21.3.12 I2C Prescaler Register (ICPSC) 881
21.3.13 I2C Revision Identification Register (REVID1) 882
21.3.14 I2C Revision Identification Register (REVID2) 882
21.3.15 I2C DMA Control Register (ICDMAC) 883
21.3.16 I2C Pin Function Register (ICPFUNC) 884
21.3.17 I2C Pin Direction Register (ICPDIR) 885
21.3.18 I2C Pin Data In Register (ICPDIN) 886

Copyright © 2013–2016, Texas Instruments Incorporated

21.3.19 I2C Pin Data Out Register (ICPDOUT) 887
21.3.20 I2C Pin Data Set Register (ICPDSET) 888
21.3.21 I2C Pin Data Clear Register (ICPDCLR) 889

22 Liquid Crystal Display Controller (LCDC) 890

22.1 Introduction 891
22.1.1 Purpose of the Peripheral 891
22.1.2 Features 892
22.1.3 Terminology 892
22.2 Architecture 892
22.2.1 Clocking 892
22.2.2 LCD External I/O Signals 894
22.2.3 DMA Engine 895
22.2.4 LIDD Controller 896
22.2.5 Raster Controller 898
22.3 Registers 908
22.3.1 LCD Revision Identification Register (REVID) 908
22.3.2 LCD Control Register (LCD_CTRL) 909
22.3.3 LCD Status Register (LCD_STAT) 911
22.3.4 LCD LIDD Control Register (LIDD_CTRL) 914
22.3.5 LCD LIDD CS*n* Configuration Registers (LIDD_CS0_CONF and LIDD_CS1_CONF) 916
22.3.6 LCD LIDD CS*n* Address Read/Write Registers (LIDD_CS0_ADDR and LIDD_CS1_ADDR) 917
22.3.7 LCD LIDD CS*n* Data Read/Write Registers (LIDD_CS0_DATA and LIDD_CS1_DATA) 918
22.3.8 LCD Raster Control Register (RASTER_CTRL) 919
22.3.9 LCD Raster Timing Register 0 (RASTER_TIMING_0) 926
22.3.10 LCD Raster Timing Register 1 (RASTER_TIMING_1) 928
22.3.11 LCD Raster Timing Register 2 (RASTER_TIMING_2) 932
22.3.12 LCD Raster Subpanel Display Register (RASTER_SUBPANEL) 936
22.3.13 LCD DMA Control Register (LCDDMA_CTRL) 938
22.3.14 LCD DMA Frame Buffer *n* Base Address Registers (LCDDMA_FB0_BASE and LCDDMA_FB1_BASE) 939
22.3.15 LCD DMA Frame Buffer *n* Ceiling Address Registers (LCDDMA_FB0_CEILING and LCDDMA_FB1_CEILING) 939

23 Multichannel Audio Serial Port (McASP) 940

23.0.16 Features 941
23.0.17 Protocols Supported 942
23.0.18 Functional Block Diagram 943
23.0.19 Definition of Terms 951
23.0.20 Overview 954
23.0.21 Clock and Frame Sync Generators 954
23.0.22 Reset Considerations 995
23.0.23 EDMA Event Support 995
23.0.24 Power Management 995
23.1 Registers 996
23.1.1 Register Bit Restrictions 999
23.1.2 Revision Identification Register (REV) 1000
23.1.3 Pin Function Register (PFUNC) 1001
23.1.4 Pin Direction Register (PDIR) 1003
23.1.5 Pin Data Output Register (PDOUT) 1005
23.1.6 Pin Data Input Register (PDIN) 1007
23.1.7 Pin Data Set Register (PDSET) 1009
23.1.8 Pin Data Clear Register (PDCLR) 1011
23.1.9 Global Control Register (GBLCTL) 1013
23.1.10 Audio Mute Control Register (AMUTE) 1015

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.11 Digital Loopback Control Register (DLBCTL) 1017
23.1.12 Digital Mode Control Register (DITCTL) 1018
23.1.13 Receiver Global Control Register (RGBLCTL) 1019
23.1.14 Receive Format Unit Bit Mask Register (RMASK) 1020
23.1.15 Receive Bit Stream Format Register (RFMT) 1021
23.1.16 Receive Frame Sync Control Register (AFSRCTL) 1023
23.1.17 Receive Clock Control Register (ACLKRCTL) 1024
23.1.18 Receive High-Frequency Clock Control Register (AHCLKRCTL) 1025
23.1.19 Receive TDM Time Slot Register (RTDM) 1026
23.1.20 Receiver Interrupt Control Register (RINTCTL) 1027
23.1.21 Receiver Status Register (RSTAT) 1028
23.1.22 Current Receive TDM Time Slot Registers (RSLOT) 1029
23.1.23 Receive Clock Check Control Register (RCLKCHK) 1030
23.1.24 Receiver DMA Event Control Register (REVTCTL) 1031
23.1.25 Transmitter Global Control Register (XGBLCTL) 1032
23.1.26 Transmit Format Unit Bit Mask Register (XMASK) 1033
23.1.27 Transmit Bit Stream Format Register (XFMT) 1034
23.1.28 Transmit Frame Sync Control Register (AFSXCTL) 1036
23.1.29 Transmit Clock Control Register (ACLKXCTL) 1037
23.1.30 Transmit High-Frequency Clock Control Register (AHCLKXCTL) 1038
23.1.31 Transmit TDM Time Slot Register (XTDM) 1039
23.1.32 Transmitter Interrupt Control Register (XINTCTL) 1040
23.1.33 Transmitter Status Register (XSTAT) 1041
23.1.34 Current Transmit TDM Time Slot Register (XSLOT) 1042
23.1.35 Transmit Clock Check Control Register (XCLKCHK) 1043
23.1.36 Transmitter DMA Event Control Register (XEVTCTL) 1044
23.1.37 Serializer Control Registers (SRCTL*n*) 1045
23.1.38 DIT Left Channel Status Registers (DITCSRA0-DITCSRA5) 1046
23.1.39 DIT Right Channel Status Registers (DITCSRB0-DITCSRB5) 1046
23.1.40 DIT Left Channel User Data Registers (DITUDRA0-DITUDRA5) 1047
23.1.41 DIT Right Channel User Data Registers (DITUDRB0-DITUDRB5) 1047
23.1.42 Transmit Buffer Registers (XBUF*n*) 1048
23.1.43 Receive Buffer Registers (RBUF*n*) 1048
23.1.44 AFIFO Revision Identification Register (AFIFOREV) 1049
23.1.45 Write FIFO Control Register (WFIFOCTL) 1050
23.1.46 Write FIFO Status Register (WFIFOSTS) 1051
23.1.47 Read FIFO Control Register (RFIFOCTL) 1052
23.1.48 Read FIFO Status Register (RFIFOSTS) 1053

24 Multichannel Buffered Serial Port (McBSP) 1054

24.1 Introduction 1055
24.1.1 Purpose of the Peripheral 1055
24.1.2 Features 1055
24.1.3 Functional Block Diagram 1056
24.1.4 Industry Standard Compliance Statement 1056
24.2 Architecture 1057
24.2.1 Clock Control 1057
24.2.2 Signal Descriptions 1057
24.2.3 Pin Multiplexing 1057
24.2.4 Endianness Considerations 1057
24.2.5 Clock, Frames, and Data 1058
24.2.6 McBSP Buffer FIFO (BFIFO) 1072
24.2.7 McBSP Standard Operation 1072
24.2.8 μ-Law/A-Law Companding Hardware Operation 1086

Copyright © 2013–2016, Texas Instruments Incorporated

24.2.9 Multichannel Selection Modes 1088
24.2.10 SPI Operation Using the Clock Stop Mode 1096
24.2.11 Resetting the Serial Port: RRST, XRST, GRST, and RESET 1096
24.2.12 McBSP Initialization Procedure 1097
24.2.13 Interrupt Support 1101
24.2.14 EDMA Event Support 1102
24.2.15 Power Management 1103
24.2.16 Emulation Considerations 1103
24.3 Registers 1104
24.3.1 Data Receive Register (DRR) 1105
24.3.2 Data Transmit Register (DXR) 1105
24.3.3 Serial Port Control Register (SPCR) 1106
24.3.4 Receive Control Register (RCR) 1108
24.3.5 Transmit Control Register (XCR) 1110
24.3.6 Sample Rate Generator Register (SRGR) 1112
24.3.7 Multichannel Control Register (MCR) 1113
24.3.8 Enhanced Receive Channel Enable Registers (RCERE0-RCERE3) 1117
24.3.9 Enhanced Transmit Channel Enable Registers (XCERE0-XCERE3) 1119
24.3.10 Pin Control Register (PCR) 1121
24.3.11 BFIFO Revision Identification Register (BFIFOREV) 1123
24.3.12 Write FIFO Control Register (WFIFOCTL) 1124
24.3.13 Write FIFO Status Register (WFIFOSTS) 1125
24.3.14 Read FIFO Control Register (RFIFOCTL) 1126
24.3.15 Read FIFO Status Register (RFIFOSTS) 1127

25 Multimedia Card (MMC)/Secure Digital (SD) Card Controller 1128

25.1 Introduction 1129
25.1.1 Purpose of the Peripheral 1129
25.1.2 Features 1129
25.1.3 Functional Block Diagram 1129
25.1.4 Supported Use Case Statement 1129
25.1.5 Industry Standard(s) Compliance Statement 1130
25.2 Architecture 1130
25.2.1 Clock Control 1131
25.2.2 Signal Descriptions 1132
25.2.3 Protocol Descriptions 1133
25.2.4 Data Flow in the Input/Output FIFO 1134
25.2.5 Data Flow in the Data Registers (MMCDRR and MMCDXR) 1136
25.2.6 FIFO Operation During Card Read Operation 1137
25.2.7 FIFO Operation During Card Write Operation 1139
25.2.8 Reset Considerations 1139
25.2.9 Initialization 1141
25.2.10 Interrupt Support 1144
25.2.11 DMA Event Support 1145
25.2.12 Power Management 1145
25.2.13 Emulation Considerations 1145
25.3 Procedures for Common Operations 1146
25.3.1 Card Identification Operation 1146
25.3.2 MMC/SD Mode Single-Block Write Operation Using CPU 1149
25.3.3 MMC/SD Mode Single-Block Write Operation Using the EDMA 1151
25.3.4 MMC/SD Mode Single-Block Read Operation Using the CPU 1151
25.3.5 MMC/SD Mode Single-Block Read Operation Using EDMA 1153
25.3.6 MMC/SD Mode Multiple-Block Write Operation Using CPU 1153
25.3.7 MMC/SD Mode Multiple-Block Write Operation Using EDMA 1155

Copyright © 2013–2016, Texas Instruments Incorporated

25.3.8 MMC/SD Mode Multiple-Block Read Operation Using CPU 1155
25.3.9 MMC/SD Mode Multiple-Block Read Operation Using EDMA 1157
25.3.10 SDIO Card Function 1157
25.4 Registers 1158
25.4.1 MMC Control Register (MMCCTL) 1159
25.4.2 MMC Memory Clock Control Register (MMCCLK) 1160
25.4.3 MMC Status Register 0 (MMCST0) 1161
25.4.4 MMC Status Register 1 (MMCST1) 1163
25.4.5 MMC Interrupt Mask Register (MMCIM) 1164
25.4.6 MMC Response Time-Out Register (MMCTOR) 1166
25.4.7 MMC Data Read Time-Out Register (MMCTOD) 1167
25.4.8 MMC Block Length Register (MMCBLEN) 1168
25.4.9 MMC Number of Blocks Register (MMCNBLK) 1169
25.4.10 MMC Number of Blocks Counter Register (MMCNBLC) 1169
25.4.11 MMC Data Receive Register (MMCDRR) 1170
25.4.12 MMC Data Transmit Register (MMCDXR) 1170
25.4.13 MMC Command Register (MMCCMD) 1171
25.4.14 MMC Argument Register (MMCARGHL) 1173
25.4.15 MMC Response Registers (MMCRSP0-MMCRSP7) 1174
25.4.16 MMC Data Response Register (MMCDRSP) 1176
25.4.17 MMC Command Index Register (MMCCIDX) 1176
25.4.18 SDIO Control Register (SDIOCTL) 1177
25.4.19 SDIO Status Register 0 (SDIOST0) 1178
25.4.20 SDIO Interrupt Enable Register (SDIOIEN) 1179
25.4.21 SDIO Interrupt Status Register (SDIOIST) 1179
25.4.22 MMC FIFO Control Register (MMCFIFOCTL) 1180

26 Real-Time Clock (RTC) 1181

26.1 Introduction 1182
26.1.1 Purpose of the Peripheral 1182
26.1.2 Features 1182
26.1.3 Block Diagram 1182
26.2 Architecture 1183
26.2.1 Clock Source 1183
26.2.2 Signal Descriptions 1183
26.2.3 Isolated Power Supply 1183
26.2.4 Operation 1184
26.2.5 Interrupt Requests 1186
26.2.6 Register Protection Against Spurious Writes 1187
26.2.7 General-Purpose Scratch Registers 1188
26.2.8 Real-Time Clock Response to Low Power Modes (Idle Configurations) 1188
26.2.9 Emulation Modes of the Real-Time Clock 1188
26.2.10 Reset Considerations 1188
26.3 Registers 1189
26.3.1 Second Register (SECOND) 1190
26.3.2 Minute Register (MINUTE) 1190
26.3.3 Hour Register (HOUR) 1191
26.3.4 Day of the Month Register (DAY) 1192
26.3.5 Month Register (MONTH) 1192
26.3.6 Year Register (YEAR) 1193
26.3.7 Day of the Week Register (DOTW) 1193
26.3.8 Alarm Second Register (ALARMSECOND) 1194
26.3.9 Alarm Minute Register (ALARMMINUTE) 1194
26.3.10 Alarm Hour Register (ALARMHOUR) 1195

Copyright © 2013–2016, Texas Instruments Incorporated

26.3.11 Alarm Day of the Month Register (ALARMDAY) 1196
26.3.12 Alarm Month Register (ALARMMONTH) 1197
26.3.13 Alarm Year Register (ALARMYEAR) 1197
26.3.14 Control Register (CTRL) 1198
26.3.15 Status Register (STATUS) 1199
26.3.16 Interrupt Register (INTERRUPT) 1200
26.3.17 Compensation (LSB) Register (COMPLSB) 1201
26.3.18 Compensation (MSB) Register (COMPMSB) 1202
26.3.19 Oscillator Register (OSC) 1203
26.3.20 Scratch Registers (SCRATCH0-SCRATCH2) 1204
26.3.21 Kick Registers (KICK0R, KICK1R) 1204

27 Serial Peripheral Interface (SPI) 1205

27.1 Introduction 1206
27.1.1 Purpose of the Peripheral 1206
27.1.2 Features 1206
27.1.3 Functional Block Diagram 1207
27.1.4 Industry Standard(s) Compliance Statement 1207
27.2 Architecture 1208
27.2.1 Clock 1208
27.2.2 Signal Descriptions 1208
27.2.3 Operation Modes 1208
27.2.4 Programmable Registers 1209
27.2.5 Master Mode Settings 1210
27.2.6 Slave Mode Settings 1212
27.2.7 SPI Operation: 3-Pin Mode 1213
27.2.8 SPI Operation: 4-Pin with Chip Select Mode 1214
27.2.9 SPI Operation: 4-Pin with Enable Mode 1216
27.2.10 SPI Operation: 5-Pin Mode 1218
27.2.11 Data Formats 1220
27.2.12 Interrupt Support 1223
27.2.13 DMA Events Support 1224
27.2.14 Robustness Features 1224
27.2.15 Reset Considerations 1226
27.2.16 Power Management 1226
27.2.17 General-Purpose I/O Pin 1227
27.2.18 Emulation Considerations 1227
27.2.19 Initialization 1227
27.2.20 Timing Diagrams 1228
27.3 Registers 1234
27.3.1 SPI Global Control Register 0 (SPIGCR0) 1234
27.3.2 SPI Global Control Register 1 (SPIGCR1) 1235
27.3.3 SPI Interrupt Register (SPIINT0) 1237
27.3.4 SPI Interrupt Level Register (SPILVL) 1239
27.3.5 SPI Flag Register (SPIFLG) 1240
27.3.6 SPI Pin Control Register 0 (SPIPC0) 1242
27.3.7 SPI Pin Control Register 1 (SPIPC1) 1243
27.3.8 SPI Pin Control Register 2 (SPIPC2) 1244
27.3.9 SPI Pin Control Register 3 (SPIPC3) 1245
27.3.10 SPI Pin Control Register 4 (SPIPC4) 1246
27.3.11 SPI Pin Control Register 5 (SPIPC5) 1247
27.3.12 SPI Transmit Data Register 0 (SPIDAT0) 1248
27.3.13 SPI Transmit Data Register 1 (SPIDAT1) 1249
27.3.14 SPI Receive Buffer Register (SPIBUF) 1250

Copyright © 2013–2016, Texas Instruments Incorporated

27.3.15 SPI Emulation Register (SPIEMU) ... 1252
27.3.16 SPI Delay Register (SPIDELAY) ... 1253
27.3.17 SPI Default Chip Select Register (SPIDEF) ... 1256
27.3.18 SPI Data Format Registers (SPIFMT*n*) ... 1257
27.3.19 SPI Interrupt Vector Register 1 (INTVEC1) ... 1259

28 64-Bit Timer Plus ... 1260

28.1 Introduction ... 1261
28.1.1 Purpose of the Peripheral ... 1261
28.1.2 Features ... 1261
28.1.3 Block Diagram ... 1262
28.1.4 Industry Standard Compatibility Statement ... 1262
28.1.5 Architecture – General-Purpose Timer Mode ... 1262
28.1.6 Architecture – Watchdog Timer Mode ... 1274
28.1.7 Reset Considerations ... 1276
28.1.8 Interrupt Support ... 1276
28.1.9 DMA Event Support ... 1276
28.1.10 TM64P_OUT Event Support ... 1277
28.1.11 Interrupt/DMA Event Generation Control and Status ... 1278
28.1.12 Power Management ... 1278
28.1.13 Emulation Considerations ... 1278
28.2 Registers ... 1279
28.2.1 Revision ID Register (REVID) ... 1281
28.2.2 Emulation Management Register (EMUMGT) ... 1281
28.2.3 GPIO Interrupt Control and Enable Register (GPINTGPEN) ... 1282
28.2.4 GPIO Data and Direction Register (GPDATGPDIR) ... 1283
28.2.5 Timer Counter Registers (TIM12 and TIM34) ... 1284
28.2.6 Timer Period Registers (PRD12 and PRD34) ... 1285
28.2.7 Timer Control Register (TCR) ... 1286
28.2.8 Timer Global Control Register (TGCR) ... 1288
28.2.9 Watchdog Timer Control Register (WDTCR) ... 1289
28.2.10 Timer Reload Register 12 (REL12) ... 1290
28.2.11 Timer Reload Register 34 (REL34) ... 1290
28.2.12 Timer Capture Register 12 (CAP12) ... 1291
28.2.13 Timer Capture Register 34 (CAP34) ... 1291
28.2.14 Timer Interrupt Control and Status Register (INTCTLSTAT) ... 1292

29 Universal Asynchronous Receiver/Transmitter (UART) ... 1294

29.1 Introduction ... 1295
29.1.1 Purpose of the Peripheral ... 1295
29.1.2 Features ... 1295
29.1.3 Functional Block Diagram ... 1295
29.1.4 Industry Standard(s) Compliance Statement ... 1295
29.2 Peripheral Architecture ... 1297
29.2.1 Clock Generation and Control ... 1297
29.2.2 Signal Descriptions ... 1299
29.2.3 Pin Multiplexing ... 1299
29.2.4 Protocol Description ... 1299
29.2.5 Operation ... 1301
29.2.6 Reset Considerations ... 1305
29.2.7 Initialization ... 1305
29.2.8 Interrupt Support ... 1305
29.2.9 DMA Event Support ... 1307
29.2.10 Power Management ... 1307
29.2.11 Emulation Considerations ... 1307

Copyright © 2013–2016, Texas Instruments Incorporated

29.2.12 Exception Processing 1307
29.3 Registers 1308
29.3.1 Receiver Buffer Register (RBR) 1309
29.3.2 Transmitter Holding Register (THR) 1310
29.3.3 Interrupt Enable Register (IER) 1311
29.3.4 Interrupt Identification Register (IIR) 1312
29.3.5 FIFO Control Register (FCR) 1313
29.3.6 Line Control Register (LCR) 1315
29.3.7 Modem Control Register (MCR) 1317
29.3.8 Line Status Register (LSR) 1318
29.3.9 Modem Status Register (MSR) 1321
29.3.10 Scratch Pad Register (SCR) 1322
29.3.11 Divisor Latches (DLL and DLH) 1322
29.3.12 Revision Identification Registers (REVID1 and REVID2) 1324
29.3.13 Power and Emulation Management Register (PWREMU_MGMT) 1325
29.3.14 Mode Definition Register (MDR) 1326
30 Universal Parallel Port (uPP) 1327
30.1 Introduction 1328
30.1.1 Purpose of the Peripheral 1328
30.1.2 Features 1328
30.1.3 Functional Block Diagram 1328
30.2 Architecture 1331
30.2.1 Clock Generation and Control 1331
30.2.2 Signal Description 1332
30.2.3 Pin Multiplexing 1333
30.2.4 Internal DMA Controller Description 1333
30.2.5 Protocol Description 1336
30.2.6 Initialization and Operation 1341
30.2.7 Reset Considerations 1346
30.2.8 Interrupt Support 1346
30.2.9 Power Management 1347
30.2.10 Emulation Considerations 1347
30.2.11 Transmit and Receive FIFOs 1347
30.3 Registers 1348
30.3.1 uPP Peripheral Identification Register (UPPID) 1348
30.3.2 uPP Peripheral Control Register (UPPCR) 1349
30.3.3 uPP Digital Loopback Register (UPDLB) 1350
30.3.4 uPP Channel Control Register (UPCTL) 1351
30.3.5 uPP Interface Configuration Register (UPICR) 1353
30.3.6 uPP Interface Idle Value Register (UPIVR) 1355
30.3.7 uPP Threshold Configuration Register (UPTCR) 1356
30.3.8 uPP Interrupt Raw Status Register (UPISR) 1357
30.3.9 uPP Interrupt Enabled Status Register (UPIER) 1359
30.3.10 uPP Interrupt Enable Set Register (UPIES) 1361
30.3.11 uPP Interrupt Enable Clear Register (UPIEC) 1363
30.3.12 uPP End of Interrupt Register (UPEOI) 1365
30.3.13 uPP DMA Channel I Descriptor 0 Register (UPID0) 1365
30.3.14 uPP DMA Channel I Descriptor 1 Register (UPID1) 1366
30.3.15 uPP DMA Channel I Descriptor 2 Register (UPID2) 1366
30.3.16 uPP DMA Channel I Status 0 Register (UPIS0) 1367
30.3.17 uPP DMA Channel I Status 1 Register (UPIS1) 1367
30.3.18 uPP DMA Channel I Status 2 Register (UPIS2) 1368
30.3.19 uPP DMA Channel Q Descriptor 0 Register (UPQD0) 1369

Copyright © 2013–2016, Texas Instruments Incorporated

30.3.20 uPP DMA Channel Q Descriptor 1 Register (UPQD1) 1369
30.3.21 uPP DMA Channel Q Descriptor 2 Register (UPQD2) 1370
30.3.22 uPP DMA Channel Q Status 0 Register (UPQS0) 1371
30.3.23 uPP DMA Channel Q Status 1 Register (UPQS1) 1371
30.3.24 uPP DMA Channel Q Status 2 Register (UPQS2) 1372

31 Universal Serial Bus 2.0 (USB) Controller 1373

31.1 Introduction 1374
31.1.1 Purpose of the Peripheral 1374
31.1.2 Features 1374
31.1.3 Functional Block Diagram 1374
31.1.4 Industry Standard(s) Compliance Statement 1375
31.2 Architecture 1375
31.2.1 Clock Control 1375
31.2.2 Signal Descriptions 1376
31.2.3 Indexed and Non-Indexed Registers 1376
31.2.4 USB PHY Initialization 1377
31.2.5 VBUS Voltage Sourcing Control 1377
31.2.6 Dynamic FIFO Sizing 1377
31.2.7 USB Controller Host and Peripheral Modes Operation 1378
31.2.8 Communications Port Programming Interface (CPPI) 4.1 DMA Overview 1414
31.2.9 Test Modes 1438
31.2.10 Reset Considerations 1440
31.2.11 Interrupt Support 1440
31.2.12 DMA Event Support 1440
31.2.13 Power Management 1440
31.3 Use Cases 1441
31.3.1 User Case 1: Example of How to Initialize the USB Controller 1441
31.3.2 User Case 2: Example of How to Program the USB Endpoints in Peripheral Mode 1445
31.3.3 User Case 3: Example of How to Program the USB Endpoints in Host Mode 1446
31.3.4 User Case 4: Example of How to Program the USB DMA Controller 1448
31.4 Registers 1453
31.4.1 Revision Identification Register (REVID) 1460
31.4.2 Control Register (CTRLR) 1460
31.4.3 Status Register (STATR) 1461
31.4.4 Emulation Register (EMUR) 1461
31.4.5 Mode Register (MODE) 1462
31.4.6 Auto Request Register (AUTOREQ) 1464
31.4.7 SRP Fix Time Register (SRPFIXTIME) 1465
31.4.8 Teardown Register (TEARDOWN) 1465
31.4.9 USB Interrupt Source Register (INTSRCR) 1466
31.4.10 USB Interrupt Source Set Register (INTSETR) 1467
31.4.11 USB Interrupt Source Clear Register (INTCLRR) 1468
31.4.12 USB Interrupt Mask Register (INTMSKR) 1469
31.4.13 USB Interrupt Mask Set Register (INTMSKSETR) 1470
31.4.14 USB Interrupt Mask Clear Register (INTMSKCLRR) 1471
31.4.15 USB Interrupt Source Masked Register (INTMASKEDR) 1472
31.4.16 USB End of Interrupt Register (EOIR) 1473
31.4.17 Generic RNDIS EP1 Size Register (GENRNDISSZ1) 1473
31.4.18 Generic RNDIS EP2 Size Register (GENRNDISSZ2) 1474
31.4.19 Generic RNDIS EP3 Size Register (GENRNDISSZ3) 1474
31.4.20 Generic RNDIS EP4 Size Register (GENRNDISSZ4) 1475
31.4.21 Function Address Register (FADDR) 1475
31.4.22 Power Management Register (POWER) 1476

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.23 Interrupt Register for Endpoint 0 Plus Transmit Endpoints 1 to 4 (INTRTX) 1477
31.4.24 Interrupt Register for Receive Endpoints 1 to 4 (INTRRX) 1478
31.4.25 Interrupt Enable Register for INTRTX (INTRTXE) 1479
31.4.26 Interrupt Enable Register for INTRRX (INTRRXE) 1479
31.4.27 Interrupt Register for Common USB Interrupts (INTRUSB) 1480
31.4.28 Interrupt Enable Register for INTRUSB (INTRUSBE) 1481
31.4.29 Frame Number Register (FRAME) 1481
31.4.30 Index Register for Selecting the Endpoint Status and Control Registers (INDEX) 1482
31.4.31 Register to Enable the USB 2.0 Test Modes (TESTMODE) 1482
31.4.32 Maximum Packet Size for Peripheral/Host Transmit Endpoint (TXMAXP) 1483
31.4.33 Control Status Register for Endpoint 0 in Peripheral Mode (PERI_CSR0) 1484
31.4.34 Control Status Register for Endpoint 0 in Host Mode (HOST_CSR0) 1485
31.4.35 Control Status Register for Peripheral Transmit Endpoint (PERI_TXCSR) 1486
31.4.36 Control Status Register for Host Transmit Endpoint (HOST_TXCSR) 1487
31.4.37 Maximum Packet Size for Peripheral Host Receive Endpoint (RXMAXP) 1488
31.4.38 Control Status Register for Peripheral Receive Endpoint (PERI_RXCSR) 1489
31.4.39 Control Status Register for Host Receive Endpoint (HOST_RXCSR) 1490
31.4.40 Count 0 Register (COUNT0) 1491
31.4.41 Receive Count Register (RXCOUNT) 1491
31.4.42 Type Register (Host mode only) (HOST_TYPE0) 1492
31.4.43 Transmit Type Register (Host mode only) (HOST_TXTYPE) 1492
31.4.44 NAKLimit0 Register (Host mode only) (HOST_NAKLIMIT0) 1493
31.4.45 Transmit Interval Register (Host mode only) (HOST_TXINTERVAL) 1493
31.4.46 Receive Type Register (Host mode only) (HOST_RXTYPE) 1494
31.4.47 Receive Interval Register (Host mode only) (HOST_RXINTERVAL) 1495
31.4.48 Configuration Data Register (CONFIGDATA) 1496
31.4.49 Transmit and Receive FIFO Register for Endpoint 0 (FIFO0) 1497
31.4.50 Transmit and Receive FIFO Register for Endpoint 1 (FIFO1) 1497
31.4.51 Transmit and Receive FIFO Register for Endpoint 2 (FIFO2) 1498
31.4.52 Transmit and Receive FIFO Register for Endpoint 3 (FIFO3) 1498
31.4.53 Transmit and Receive FIFO Register for Endpoint 4 (FIFO4) 1499
31.4.54 Device Control Register (DEVCTL) 1499
31.4.55 Transmit Endpoint FIFO Size (TXFIFOSZ) 1500
31.4.56 Receive Endpoint FIFO Size (RXFIFOSZ) 1500
31.4.57 Transmit Endpoint FIFO Address (TXFIFOADDR) 1501
31.4.58 Receive Endpoint FIFO Address (RXFIFOADDR) 1501
31.4.59 Hardware Version Register (HWVERS) 1502
31.4.60 Transmit Function Address (TXFUNCADDR) 1503
31.4.61 Transmit Hub Address (TXHUBADDR) 1503
31.4.62 Transmit Hub Port (TXHUBPORT) 1503
31.4.63 Receive Function Address (RXFUNCADDR) 1504
31.4.64 Receive Hub Address (RXHUBADDR) 1504
31.4.65 Receive Hub Port (RXHUBPORT) 1504
31.4.66 CDMA Revision Identification Register (DMAREVID) 1505
31.4.67 CDMA Teardown Free Descriptor Queue Control Register (TDFDQ) 1505
31.4.68 CDMA Emulation Control Register (DMAEMU) 1506
31.4.69 CDMA Transmit Channel n Global Configuration Registers (TXGCR[0]-TXGCR[3]) 1506
31.4.70 CDMA Receive Channel n Global Configuration Registers (RXGCR[0]-RXGCR[3]) 1507
31.4.71 CDMA Receive Channel n Host Packet Configuration Registers A (RXHPCRA[0]-RXHPCRA[3]) 1508
31.4.72 CDMA Receive Channel n Host Packet Configuration Registers B (RXHPCRB[0]-RXHPCRB[3]) 1509
31.4.73 CDMA Scheduler Control Register (DMA_SCHED_CTRL) 1510
31.4.74 CDMA Scheduler Table Word n Registers (WORD[0]-WORD[63]) 1510

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.75 Queue Manager Revision Identification Register (QMGRREVID) 1512
31.4.76 Queue Manager Queue Diversion Register (DIVERSION) 1512
31.4.77 Queue Manager Free Descriptor/Buffer Starvation Count Register 0 (FDBSC0) 1513
31.4.78 Queue Manager Free Descriptor/Buffer Starvation Count Register 1 (FDBSC1) 1514
31.4.79 Queue Manager Free Descriptor/Buffer Starvation Count Register 2 (FDBSC2) 1515
31.4.80 Queue Manager Free Descriptor/Buffer Starvation Count Register 3 (FDBSC3) 1516
31.4.81 Queue Manager Linking RAM Region 0 Base Address Register (LRAM0BASE) 1516
31.4.82 Queue Manager Linking RAM Region 0 Size Register (LRAM0SIZE) 1517
31.4.83 Queue Manager Linking RAM Region 1 Base Address Register (LRAM1BASE) 1517
31.4.84 Queue Manager Queue Pending Register 0 (PEND0) 1518
31.4.85 Queue Manager Queue Pending Register 1 (PEND1) 1518
31.4.86 Queue Manager Memory Region *R* Base Address Registers (QMEMRBASE[0]-QMEMRBASE[15]) 1519
31.4.87 Queue Manager Memory Region *R* Control Registers (QMEMRCTRL[0]-QMEMRCTRL[15]) 1520
31.4.88 Queue Manager Queue N Control Register D (CTRLD[0]-CTRLD[63]) 1521
31.4.89 Queue Manager Queue *N* Status Register A (QSTATA[0]-QSTATA[63]) 1522
31.4.90 Queue Manager Queue *N* Status Register B (QSTATB[0]-QSTATB[63]) 1522
31.4.91 Queue Manager Queue *N* Status Register C (QSTATC[0]-QSTATC[63]) 1523

32 Video Port Interface (VPIF) 1524

32.1 Introduction 1525
32.1.1 Overview 1525
32.1.2 Features 1526
32.1.3 Features Not Supported 1526
32.1.4 Functional Block Diagram 1526
32.2 Architecture 1529
32.2.1 Clock Control 1529
32.2.2 Signal Descriptions 1529
32.2.3 Memory Interface 1531
32.2.4 Video Transmit 1533
32.2.5 Video Receive 1534
32.2.6 Raw Data Capture 1535
32.2.7 VBI Ancillary Data 1538
32.2.8 Reset Considerations 1540
32.2.9 Initialization 1541
32.2.10 Interrupt Support 1541
32.3 Registers 1550
32.3.1 VPIF Revision Register ID (REVID) 1553
32.3.2 Channel 0 Control Register (C0CTRL) 1553
32.3.3 Channel 1 Control Register (C1CTRL) 1555
32.3.4 Channel 2 Control Register (C2CTRL) 1556
32.3.5 Channel 3 Control Register (C3CTRL) 1558
32.3.6 Interrupt Enable Register (INTEN) 1560
32.3.7 Interrupt Enable Set Register (INTSET) 1561
32.3.8 Interrupt Enable Clear Register (INTCLR) 1562
32.3.9 Interrupt Status Register (INTSTAT) 1563
32.3.10 Interrupt Status Clear Register (INTSTATCLR) 1564
32.3.11 Emulation Suspend Control Register (EMUCTRL) 1565
32.3.12 DMA Size Control Register (REQSIZE) 1565
32.3.13 Channel *n* Top Field Luminance Address Register (CnTLUMA) 1566
32.3.14 Channel *n* Bottom Field Luminance Address Register (CnBLUMA) 1566
32.3.15 Channel *n* Top Field Chrominance Address Register (C*n*TCHROMA) 1567
32.3.16 Channel *n* Bottom Field Chrominance Address Register (C*n*BCHROMA) 1567
32.3.17 Channel *n* Top Field Horizontal Ancillary Address Register (C*n*THANC) 1568

Copyright © 2013–2016, Texas Instruments Incorporated

32.3.18 Channel *n* Bottom Field Horizontal Ancillary Address Register (C*n*BHANC) 1568
32.3.19 Channel *n* Top Field Vertical Ancillary Address Register (C*n*TVANC) 1569
32.3.20 Channel *n* Bottom Field Vertical Ancillary Address Register (C*n*BVANC) 1569
32.3.21 Channel *n* Image Address Offset Register (C*n*IMGOFFSET) 1570
32.3.22 Channel *n* Horizontal Ancillary Address Offset Register (C*n*HANCOFFSET) 1570
32.3.23 Channel *n* Horizontal Size Configuration Register (C0HCFG and C1HCFG) 1571
32.3.24 Channel *n* Vertical Size Configuration 0 Register (C0VCFG0 and C1VCFG0) 1572
32.3.25 Channel *n* Vertical Size Configuration 1 Register (C0VCFG1 and C1VCFG1) 1572
32.3.26 Channel *n* Vertical Size Configuration 2 Register (C0VCFG2 and C1VCFG2) 1573
32.3.27 Channel *n* Vertical Image Size Register (C0VSIZE and C1VSIZE) 1573
32.3.28 Channel *n* Horizontal Size Configuration Register (C2HCFG and C3HCFG) 1574
32.3.29 Channel *n* Vertical Size Configuration 0 Register (C2VCFG0 and C3VCFG0) 1575
32.3.30 Channel *n* Vertical Size Configuration 1 Register (C2VCFG1 and C3VCFG1) 1575
32.3.31 Channel *n* Vertical Size Configuration 2 Register (C2VCFG2 and C3VCFG2) 1576
32.3.32 Channel *n* Vertical Image Size Register (C2VSIZE and C3VSIZE) 1576
32.3.33 Channel *n* Top Field Horizontal Ancillary Position Register (C2THANCPOS and C3THANCPOS) 1577
32.3.34 Channel *n* Top Field Horizontal Ancillary Size Register (C2THANCSIZE and C3THANCSIZE) 1578
32.3.35 Channel *n* Bottom Field Horizontal Ancillary Position Register (C2BHANCPOS and C3BHANCPOS) 1579
32.3.36 Channel *n* Bottom Field Horizontal Ancillary Size Register (C2BHANCSIZE and C3BHANCSIZE) 1580
32.3.37 Channel *n* Top Field Vertical Ancillary Position Register (C2TVANCPOS and C3TVANCPOS) 1581
32.3.38 Channel *n* Top Field Vertical Ancillary Size Register (C2TVANCSIZE and C3TVANCSIZE) 1582
32.3.39 Channel *n* Bottom Field Vertical Ancillary Position Register (C2BVANCPOS and C3BVANCPOS) 1583
32.3.40 Channel *n* Bottom Field Vertical Ancillary Size Register (C2BVANCSIZE and C3BVANCSIZE) 1584

Revision History 1585

Copyright © 2013–2016, Texas Instruments Incorporated

List of Figures

1-1. AM1806 ARM Microprocessor Block Diagram 74
3-1. System Interconnect Block Diagram 84
5-1. MPU Block Diagram 88
5-2. Permission Fields 91
5-3. Revision ID Register (REVID) 96
5-4. Configuration Register (CONFIG) 96
5-5. Interrupt Raw Status/Set Register (IRAWSTAT) 97
5-6. Interrupt Enable Status/Clear Register (IENSTAT) 98
5-7. Interrupt Enable Set Register (IENSET) 99
5-8. Interrupt Enable Clear Register (IENCLR) 99
5-9. Fixed Range Start Address Register (FXD_MPSAR) 100
5-10. Fixed Range End Address Register (FXD_MPEAR) 100
5-11. Fixed Range Memory Protection Page Attributes Register (FXD_MPPA) 101
5-12. MPU1 Programmable Range *n* Start Address Register (PROG*n*_MPSAR) 102
5-13. MPU2 Programmable Range *n* Start Address Register (PROG*n*_MPSAR) 102
5-14. MPU1 Programmable Range *n* End Address Register (PROG*n*_MPEAR) 103
5-15. MPU2 Programmable Range *n* End Address Register (PROG*n*_MPEAR) 103
5-16. Programmable Range Memory Protection Page Attributes Register (PROG*n*_MPPA) 104
5-17. Fault Address Register (FLTADDRR) 105
5-18. Fault Status Register (FLTSTAT) 106
5-19. Fault Clear Register (FLTCLR) 107
6-1. Overall Clocking Diagram 110
6-2. USB Clocking Diagram 112
6-3. DDR2/mDDR Memory Controller Clocking Diagram 114
6-4. EMIFA Clocking Diagram 115
6-5. uPP Clocking Diagram 116
6-6. McASP Clocking Diagram 117
7-1. PLLC Structure 121
7-2. PLLC0 Revision Identification Register (REVID) 126
7-3. PLLC1 Revision Identification Register (REVID) 127
7-4. Reset Type Status Register (RSTYPE) 127
7-5. Reset Control Register (RSCTRL) 128
7-6. PLLC0 Control Register (PLLCTL) 129
7-7. PLLC1 Control Register (PLLCTL) 130
7-8. PLLC0 OBSCLK Select Register (OCSEL) 131
7-9. PLLC1 OBSCLK Select Register (OCSEL) 132
7-10. PLL Multiplier Control Register (PLLM) 133
7-11. PLLC0 Pre-Divider Control Register (PREDIV) 133
7-12. PLLC0 Divider 1 Register (PLLDIV1) 134
7-13. PLLC1 Divider 1 Register (PLLDIV1) 134
7-14. PLLC0 Divider 2 Register (PLLDIV2) 135
7-15. PLLC1 Divider 2 Register (PLLDIV2) 135
7-16. PLLC0 Divider 3 Register (PLLDIV3) 136
7-17. PLLC1 Divider 3 Register (PLLDIV3) 136
7-18. PLLC0 Divider 4 Register (PLLDIV4) 137
7-19. PLLC0 Divider 5 Register (PLLDIV5) 137
7-20. PLLC0 Divider 6 Register (PLLDIV6) 138

Copyright © 2013–2016, Texas Instruments Incorporated

7-21. PLLC0 Divider 7 Register (PLLDIV7) 138
7-22. PLLC0 Oscillator Divider 1 Register (OSCDIV) 139
7-23. PLLC1 Oscillator Divider 1 Register (OSCDIV) 139
7-24. PLL Post-Divider Control Register (POSTDIV) 140
7-25. PLL Controller Command Register (PLLCMD) 140
7-26. PLL Controller Status Register (PLLSTAT) 141
7-27. PLLC0 Clock Align Control Register (ALNCTL) 142
7-28. PLLC1 Clock Align Control Register (ALNCTL) 143
7-29. PLLC0 PLLDIV Ratio Change Status Register (DCHANGE) 144
7-30. PLLC1 PLLDIV Ratio Change Status Register (DCHANGE) 145
7-31. PLLC0 Clock Enable Control Register (CKEN) 146
7-32. PLLC1 Clock Enable Control Register (CKEN) 146
7-33. PLLC0 Clock Status Register (CKSTAT) 147
7-34. PLLC1 Clock Status Register (CKSTAT) 148
7-35. PLLC0 SYSCLK Status Register (SYSTAT) 149
7-36. PLLC1 SYSCLK Status Register (SYSTAT) 150
7-37. Emulation Performance Counter 0 Register (EMUCNT0) 151
7-38. Emulation Performance Counter 1 Register (EMUCNT1) 151
8-1. Revision Identification Register (REVID) 163
8-2. Interrupt Evaluation Register (INTEVAL) 163
8-3. PSC0 Module Error Pending Register 0 (MERRPR0) 164
8-4. PSC1 Module Error Pending Register 0 (MERRPR0) 164
8-5. PSC0 Module Error Clear Register 0 (MERRCR0) 165
8-6. PSC1 Module Error Clear Register 0 (MERRCR0) 165
8-7. Power Error Pending Register (PERRPR) 166
8-8. Power Error Clear Register (PERRCR) 166
8-9. Power Domain Transition Command Register (PTCMD) 167
8-10. Power Domain Transition Status Register (PTSTAT) 168
8-11. Power Domain 0 Status Register (PDSTAT0) 169
8-12. Power Domain 1 Status Register (PDSTAT1) 170
8-13. Power Domain 0 Control Register (PDCTL0) 171
8-14. Power Domain 1 Control Register (PDCTL1) 172
8-15. Power Domain 0 Configuration Register (PDCFG0) 173
8-16. Power Domain 1 Configuration Register (PDCFG1) 174
8-17. Module Status *n* Register (MDSTAT*n*) 175
8-18. PSC0 Module Control *n* Register (MDCTL*n*) 176
8-19. PSC1 Module Control *n* Register (MDCTL*n*) 177
9-1. Deep Sleep Mode Sequence 188
10-1. Revision Identification Register (REVID) 196
10-2. Device Identification Register 0 (DEVIDR0) 197
10-3. Boot Configuration Register (BOOTCFG) 197
10-4. Chip Revision Identification Register (CHIPREVIDR) 197
10-5. Kick 0 Register (KICK0R) 199
10-6. Kick 1 Register (KICK1R) 199
10-7. Host 0 Configuration Register (HOST0CFG) 200
10-8. Interrupt Raw Status/Set Register (IRAWSTAT) 201
10-9. Interrupt Enable Status/Clear Register (IENSTAT) 202
10-10. Interrupt Enable Register (IENSET) 203
10-11. Interrupt Enable Clear Register (IENCLR) 203

Copyright © 2013–2016, Texas Instruments Incorporated

10-12. End of Interrupt Register (EOI) 204
10-13. Fault Address Register (FLTADDRR) 204
10-14. Fault Status Register (FLTSTAT) 205
10-15. Master Priority 0 Register (MSTPRI0) 206
10-16. Master Priority 1 Register (MSTPRI1) 207
10-17. Master Priority 2 Register (MSTPRI2) 208
10-18. Pin Multiplexing Control 0 Register (PINMUX0) 209
10-19. Pin Multiplexing Control 1 Register (PINMUX1) 211
10-20. Pin Multiplexing Control 2 Register (PINMUX2) 213
10-21. Pin Multiplexing Control 3 Register (PINMUX3) 215
10-22. Pin Multiplexing Control 4 Register (PINMUX4) 217
10-23. Pin Multiplexing Control 5 Register (PINMUX5) 219
10-24. Pin Multiplexing Control 6 Register (PINMUX6) 221
10-25. Pin Multiplexing Control 7 Register (PINMUX7) 223
10-26. Pin Multiplexing Control 8 Register (PINMUX8) 225
10-27. Pin Multiplexing Control 9 Register (PINMUX9) 227
10-28. Pin Multiplexing Control 10 Register (PINMUX10) 229
10-29. Pin Multiplexing Control 11 Register (PINMUX11) 231
10-30. Pin Multiplexing Control 12 Register (PINMUX12) 233
10-31. Pin Multiplexing Control 13 Register (PINMUX13) 235
10-32. Pin Multiplexing Control 14 Register (PINMUX14) 237
10-33. Pin Multiplexing Control 15 Register (PINMUX15) 239
10-34. Pin Multiplexing Control 16 Register (PINMUX16) 241
10-35. Pin Multiplexing Control 17 Register (PINMUX17) 243
10-36. Pin Multiplexing Control 18 Register (PINMUX18) 245
10-37. Pin Multiplexing Control 19 Register (PINMUX19) 247
10-38. Suspend Source Register (SUSPSRC) 249
10-39. Chip Signal Register (CHIPSIG) 252
10-40. Chip Signal Clear Register (CHIPSIG_CLR) 253
10-41. Chip Configuration 0 Register (CFGCHIP0) 254
10-42. Chip Configuration 1 Register (CFGCHIP1) 255
10-43. Chip Configuration 2 Register (CFGCHIP2) 257
10-44. Chip Configuration 3 Register (CFGCHIP3) 259
10-45. Chip Configuration 4 Register (CFGCHIP4) 260
10-46. VTP I/O Control Register (VTPIO_CTL) 261
10-47. DDR Slew Register (DDR_SLEW) 263
10-48. Deep Sleep Register (DEEPSLEEP) 264
10-49. Pullup/Pulldown Enable Register (PUPD_ENA) 265
10-50. Pullup/Pulldown Select Register (PUPD_SEL) 265
10-51. RXACTIVE Control Register (RXACTIVE) 267
11-1. AINTC Interrupt Mapping 269
11-2. Flow of System Interrupts to Host 272
11-3. Revision Identification Register (REVID) 277
11-4. Control Register (CR) 278
11-5. Global Enable Register (GER) 279
11-6. Global Nesting Level Register (GNLR) 279
11-7. System Interrupt Status Indexed Set Register (SISR) 280
11-8. System Interrupt Status Indexed Clear Register (SICR) 280
11-9. System Interrupt Enable Indexed Set Register (EISR) 281

Copyright © 2013–2016, Texas Instruments Incorporated

11-10. System Interrupt Enable Indexed Clear Register (EICR) 281
11-11. Host Interrupt Enable Indexed Set Register (HEISR) 282
11-12. Host Interrupt Enable Indexed Clear Register (HIEICR) 282
11-13. Vector Base Register (VBR) 283
11-14. Vector Size Register (VSR) 283
11-15. Vector Null Register (VNR) 284
11-16. Global Prioritized Index Register (GPIR) 284
11-17. Global Prioritized Vector Register (GPVR) 285
11-18. System Interrupt Status Raw/Set Register 1 (SRSR1) 285
11-19. System Interrupt Status Raw/Set Register 2 (SRSR2) 286
11-20. System Interrupt Status Raw/Set Register 3 (SRSR3) 286
11-21. System Interrupt Status Raw/Set Register 4 (SRSR4) 287
11-22. System Interrupt Status Enabled/Clear Register 1 (SECR1) 287
11-23. System Interrupt Status Enabled/Clear Register 2 (SECR2) 288
11-24. System Interrupt Status Enabled/Clear Register 3 (SECR3) 288
11-25. System Interrupt Status Enabled/Clear Register 4 (SECR4) 289
11-26. System Interrupt Enable Set Register 1 (ESR1) 289
11-27. System Interrupt Enable Set Register 2 (ESR2) 290
11-28. System Interrupt Enable Set Register 3 (ESR3) 290
11-29. System Interrupt Enable Set Register 4 (ESR4) 291
11-30. System Interrupt Enable Clear Register 1 (ECR1) 291
11-31. System Interrupt Enable Clear Register 2 (ECR2) 292
11-32. System Interrupt Enable Clear Register 3 (ECR3) 292
11-33. System Interrupt Enable Clear Register 4 (ECR4) 293
11-34. Channel Map Registers (CMR*n*) 293
11-35. Host Interrupt Prioritized Index Register 1 (HIPIR1) 294
11-36. Host Interrupt Prioritized Index Register 2 (HIPIR2) 294
11-37. Host Interrupt Nesting Level Register 1 (HINLR1) 295
11-38. Host Interrupt Nesting Level Register 2 (HINLR2) 295
11-39. Host Interrupt Enable Register (HIER) 296
11-40. Host Interrupt Prioritized Vector Register 1 (HIPVR1) 297
11-41. Host Interrupt Prioritized Vector Register 2 (HIPVR2) 297
13-1. PRU Block Diagram 303
13-2. Format 1a: (All Arithmetic and Logical Functions – Register Op2) 309
13-3. Format 1b: (All Arithmetic and Logical Functions – Immediate Op2) 311
13-4. Format 2 312
13-5. Format 2a: (JMP,JAL – Register Op2) 313
13-6. Format 2b: (JMP, JAL – Immediate Op2) 314
13-7. Format 2c: (LDI) 315
13-8. Format 2d: (LMBD - Leftmost Bit Detect - Register Op2) 316
13-9. Format 2e: (LMBD - Immediate Op2) 317
13-10. Format 2f: (SCAN - Register Op2) 318
13-11. Format 2g: (SCAN - Immediate Op2) 319
13-12. Format 2h: (HALT) 319
13-13. Format 2i: (SLP) 319
13-14. Format 4a: (Quick Arithmetic Test and Branch – Register Op2) 320
13-15. Format 4b: (Quick Arithmetic Test and Branch – Immediate Op2) 321
13-16. Format 5a: (Quick Bit Test and Branch – Register Op2) 322
13-17. Format 5b: (Quick Bit Test and Branch – Immediate Op2) 323

Copyright © 2013–2016, Texas Instruments Incorporated

13-18. Format 6a: (LBBO/SBBO - Register Offset) 324
13-19. Format 6b: (LBBO/SBBO - Immediate Offset) 325
13-20. Format 6c: (LBCO/SBCO - Register Offset) 326
13-21. Format 6d: (LBCO/SBCO - Immediate Offset) 327
13-22. 328
13-23. 331
13-24. CONTROL Register 336
13-25. STATUS Register 338
13-26. WAKEUP Register 338
13-27. CYCLECNT Register 338
13-28. STALLCNT Register 339
13-29. CONTABBLKIDX0 Register 339
13-30. CONTABPROPTR0 Register 340
13-31. CONTABPROPTR1 Register 340
13-32. INTGPR0 to INTGPR31 Register 341
13-33. INTCTER0 to INTCTER31 Register 341
14-1. Data Paths to DDR2/mDDR Memory Controller 355
14-2. DDR2/mDDR Memory Controller Clock Block Diagram 356
14-3. DDR2/mDDR Memory Controller Signals 357
14-4. Refresh Command 360
14-5. DCAB Command 361
14-6. DEAC Command 362
14-7. ACTV Command 363
14-8. DDR2/mDDR READ Command 364
14-9. DDR2/mDDR WRT Command 365
14-10. DDR2/mDDR MRS and EMRS Command 366
14-11. Byte Alignment 366
14-12. DDR2/mDDR SDRAM Column, Row, and Bank Access 370
14-13. Address Mapping Diagram (IBANKPOS = 1) 371
14-14. SDRAM Column, Row, Bank Access (IBANKPOS = 1) 372
14-15. DDR2/mDDR Memory Controller FIFO Block Diagram 373
14-16. DDR2/mDDR Memory Controller Reset Block Diagram 377
14-17. DDR2/mDDR Memory Controller Power Sleep Controller Diagram 382
14-18. Connecting DDR2/mDDR Memory Controller to a 16-Bit DDR2 Memory 384
14-19. Revision ID Register (REVID) 389
14-20. SDRAM Status Register (SDRSTAT) 390
14-21. SDRAM Configuration Register (SDCR) 391
14-22. SDRAM Refresh Control Register (SDRCR) 394
14-23. SDRAM Timing Register 1 (SDTIMR1) 395
14-24. SDRAM Timing Register 2 (SDTIMR2) 396
14-25. SDRAM Configuration Register 2 (SDCR2) 397
14-26. Peripheral Bus Burst Priority Register (PBBPR) 398
14-27. Performance Counter 1 Register (PC1) 399
14-28. Performance Counter 2 Register (PC2) 399
14-29. Performance Counter Configuration Register (PCC) 400
14-30. Performance Counter Master Region Select Register (PCMRS) 402
14-31. Performance Counter Time Register (PCT) 403
14-32. DDR PHY Reset Control Register (DRPYRCR) 403
14-33. Interrupt Raw Register (IRR) 404

Copyright © 2013–2016, Texas Instruments Incorporated

14-34. Interrupt Masked Register (IMR) 404
14-35. Interrupt Mask Set Register (IMSR) 405
14-36. Interrupt Mask Clear Register (IMCR) 406
14-37. DDR PHY Control Register 1 (DRPYC1R) 407
15-1. Multiple eCAP Modules 410
15-2. Capture and APWM Modes of Operation 411
15-3. Capture Function Diagram 412
15-4. Event Prescale Control 413
15-5. Prescale Function Waveforms 413
15-6. Continuous/One-shot Block Diagram 414
15-7. Counter and Synchronization Block Diagram 415
15-8. Interrupts in eCAP Module 417
15-9. PWM Waveform Details Of APWM Mode Operation 418
15-10. Capture Sequence for Absolute Time-Stamp, Rising Edge Detect 420
15-11. Capture Sequence for Absolute Time-Stamp, Rising and Falling Edge Detect 422
15-12. Capture Sequence for Delta Mode Time-Stamp, Rising Edge Detect 424
15-13. Capture Sequence for Delta Mode Time-Stamp, Rising and Falling Edge Detect 426
15-14. PWM Waveform Details of APWM Mode Operation 428
15-15. Multichannel PWM Example Using 4 eCAP Modules 430
15-16. Multiphase (channel) Interleaved PWM Example Using 3 eCAP Modules 433
15-17. Time-Stamp Counter Register (TSCTR) 435
15-18. Counter Phase Control Register (CTRPHS) 436
15-19. Capture 1 Register (CAP1) 436
15-20. Capture 2 Register (CAP2) 437
15-21. Capture 3 Register (CAP3) 437
15-22. Capture 4 Register (CAP4) 438
15-23. ECAP Control Register 1 (ECCTL1) 438
15-24. ECAP Control Register 2 (ECCTL2) 440
15-25. ECAP Interrupt Enable Register (ECEINT) 442
15-26. ECAP Interrupt Flag Register (ECFLG) 443
15-27. ECAP Interrupt Clear Register (ECCLR) 444
15-28. ECAP Interrupt Forcing Register (ECFRC) 445
15-29. Revision ID Register (REVID) 446
16-1. Multiple ePWM Modules 449
16-2. Submodules and Signal Connections for an ePWM Module 450
16-3. ePWM Submodules and Critical Internal Signal Interconnects 451
16-4. Time-Base Submodule Block Diagram 457
16-5. Time-Base Submodule Signals and Registers 458
16-6. Time-Base Frequency and Period 460
16-7. Time-Base Counter Synchronization Scheme 1 461
16-8. Time-Base Up-Count Mode Waveforms 463
16-9. Time-Base Down-Count Mode Waveforms 464
16-10. Time-Base Up-Down-Count Waveforms, TBCTL[PHSDIR = 0] Count Down on Synchronization Event 464
16-11. Time-Base Up-Down Count Waveforms, TBCTL[PHSDIR = 1] Count Up on Synchronization Event 465
16-12. Counter-Compare Submodule 466
16-13. Counter-Compare Submodule Signals and Registers 466
16-14. Counter-Compare Event Waveforms in Up-Count Mode 469
16-15. Counter-Compare Events in Down-Count Mode 469
16-16. Counter-Compare Events in Up-Down-Count Mode, TBCTL[PHSDIR = 0] Count Down on

Copyright © 2013–2016, Texas Instruments Incorporated

Synchronization Event 470
16-17. Counter-Compare Events in Up-Down-Count Mode, TBCTL[PHSDIR = 1] Count Up on Synchronization Event 470
16-18. Action-Qualifier Submodule 471
16-19. Action-Qualifier Submodule Inputs and Outputs 472
16-20. Possible Action-Qualifier Actions for EPWMxA and EPWMxB Outputs 473
16-21. Up-Down-Count Mode Symmetrical Waveform 476
16-22. Up, Single Edge Asymmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB—Active High 477
16-23. Up, Single Edge Asymmetric Waveform With Independent Modulation on EPWMxA and EPWMxB—Active Low 479
16-24. Up-Count, Pulse Placement Asymmetric Waveform With Independent Modulation on EPWMxA 481
16-25. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Active Low 483
16-26. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Complementary 485
16-27. Up-Down-Count, Dual Edge Asymmetric Waveform, With Independent Modulation on EPWMxA—Active Low 487
16-28. Dead-Band Generator Submodule 489
16-29. Configuration Options for the Dead-Band Generator Submodule 490
16-30. Dead-Band Waveforms for Typical Cases (0% < Duty < 100%) 492
16-31. PWM-Chopper Submodule 493
16-32. PWM-Chopper Submodule Signals and Registers 494
16-33. Simple PWM-Chopper Submodule Waveforms Showing Chopping Action Only 495
16-34. PWM-Chopper Submodule Waveforms Showing the First Pulse and Subsequent Sustaining Pulses 495
16-35. PWM-Chopper Submodule Waveforms Showing the Pulse Width (Duty Cycle) Control of Sustaining Pulses 496
16-36. Trip-Zone Submodule 497
16-37. Trip-Zone Submodule Mode Control Logic 500
16-38. Trip-Zone Submodule Interrupt Logic 500
16-39. Event-Trigger Submodule 501
16-40. Event-Trigger Submodule Inter-Connectivity to Interrupt Controller 502
16-41. Event-Trigger Submodule Showing Event Inputs and Prescaled Outputs 502
16-42. Event-Trigger Interrupt Generator 504
16-43. HRPWM System Interface 505
16-44. Resolution Calculations for Conventionally Generated PWM 506
16-45. Operating Logic Using MEP 507
16-46. Required PWM Waveform for a Requested Duty = 40.5% 509
16-47. Low % Duty Cycle Range Limitation Example When PWM Frequency = 1 MHz 511
16-48. High % Duty Cycle Range Limitation Example when PWM Frequency = 1 MHz 511
16-49. Simplified ePWM Module 512
16-50. EPWM1 Configured as a Typical Master, EPWM2 Configured as a Slave 513
16-51. Control of Four Buck Stages. (Note: $F_{PWM1} \neq F_{PWM2} \neq F_{PWM3} \neq F_{PWM4}$) 514
16-52. Buck Waveforms for (Note: Only three bucks shown here) 515
16-53. Control of Four Buck Stages. (Note: $F_{PWM2} = N \times F_{PWM1}$) 517
16-54. Buck Waveforms for (Note: $F_{PWM2} = F_{PWM1}$) 518
16-55. Control of Two Half-H Bridge Stages ($F_{PWM2} = N \times F_{PWM1}$) 520
16-56. Half-H Bridge Waveforms for (Note: $F_{PWM2} = F_{PWM1}$) 521
16-57. Control of Dual 3-Phase Inverter Stages as Is Commonly Used in Motor Control 523
16-58. 3-Phase Inverter Waveforms for (Only One Inverter Shown) 524
16-59. Configuring Two PWM Modules for Phase Control 527

Copyright © 2013–2016, Texas Instruments Incorporated

16-60. Timing Waveforms Associated With Phase Control Between 2 Modules 528
16-61. Control of a 3-Phase Interleaved DC/DC Converter 529
16-62. 3-Phase Interleaved DC/DC Converter Waveforms for 530
16-63. Controlling a Full-H Bridge Stage (F_{PWM2} = F_{PWM1}) 533
16-64. ZVS Full-H Bridge Waveforms 534
16-65. Time-Base Control Register (TBCTL) 536
16-66. Time-Base Status Register (TBSTS) 538
16-67. Time-Base Phase Register (TBPHS) 539
16-68. Time-Base Counter Register (TBCNT) 539
16-69. Time-Base Period Register (TBPRD) 540
16-70. Counter-Compare Control Register (CMPCTL) 541
16-71. Counter-Compare A Register (CMPA) 542
16-72. Counter-Compare B Register (CMPB) 543
16-73. Action-Qualifier Output A Control Register (AQCTLA) 544
16-74. Action-Qualifier Output B Control Register (AQCTLB) 545
16-75. Action-Qualifier Software Force Register (AQSFRC) 546
16-76. Action-Qualifier Continuous Software Force Register (AQCSFRC) 547
16-77. Dead-Band Generator Control Register (DBCTL) 548
16-78. Dead-Band Generator Rising Edge Delay Register (DBRED) 549
16-79. Dead-Band Generator Falling Edge Delay Register (DBFED) 549
16-80. PWM-Chopper Control Register (PCCTL) 550
16-81. Trip-Zone Select Register (TZSEL) 551
16-82. Trip-Zone Control Register (TZCTL) 552
16-83. Trip-Zone Enable Interrupt Register (TZEINT) 552
16-84. Trip-Zone Flag Register (TZFLG) 553
16-85. Trip-Zone Clear Register (TZCLR) 554
16-86. Trip-Zone Force Register (TZFRC) 554
16-87. Event-Trigger Selection Register (ETSEL) 555
16-88. Event-Trigger Prescale Register (ETPS) 556
16-89. Event-Trigger Flag Register (ETFLG) 557
16-90. Event-Trigger Clear Register (ETCLR) 557
16-91. Event-Trigger Force Register (ETFRC) 558
16-92. Time-Base Phase High-Resolution Register (TBPHSHR) 559
16-93. Counter-Compare A High-Resolution Register (CMPAHR) 559
16-94. HRPWM Configuration Register (HRCNFG) 560
17-1. EDMA3 Controller Block Diagram 565
17-2. EDMA3 Channel Controller (EDMA3CC) Block Diagram 568
17-3. EDMA3 Transfer Controller (EDMA3TC) Block Diagram 569
17-4. Definition of ACNT, BCNT, and CCNT 570
17-5. A-Synchronized Transfers (ACNT = n, BCNT = 4, CCNT = 3) 571
17-6. AB-Synchronized Transfers (ACNT = n, BCNT = 4, CCNT = 3) 572
17-7. PaRAM Set 573
17-8. Linked Transfer Example 581
17-9. Link-to-Self Transfer Example 582
17-10. QDMA Channel to PaRAM Mapping 589
17-11. Shadow Region Registers 591
17-12. Interrupt Diagram 595
17-13. Error Interrupt Operation 598
17-14. EDMA3 Prioritization 605

Copyright © 2013–2016, Texas Instruments Incorporated

17-15. Block Move Example 609
17-16. Block Move Example PaRAM Configuration 610
17-17. Subframe Extraction Example 611
17-18. Subframe Extraction Example PaRAM Configuration 611
17-19. Data Sorting Example 612
17-20. Data Sorting Example PaRAM Configuration 613
17-21. Servicing Incoming McBSP Data Example 614
17-22. Servicing Incoming McBSP Data Example PaRAM 615
17-23. Servicing Peripheral Burst Example 616
17-24. Servicing Peripheral Burst Example PaRAM 616
17-25. Servicing Continuous McBSP Data Example 617
17-26. Servicing Continuous McBSP Data Example PaRAM 618
17-27. Servicing Continuous McBSP Data Example Reload PaRAM 618
17-28. Ping-Pong Buffering for McBSP Data Example 621
17-29. Ping-Pong Buffering for McBSP Example PaRAM 622
17-30. Ping-Pong Buffering for McBSP Example Pong PaRAM 622
17-31. Ping-Pong Buffering for McBSP Example Ping PaRAM 623
17-32. Intermediate Transfer Completion Chaining Example 625
17-33. Single Large Block Transfer Example 625
17-34. Smaller Packet Data Transfers Example 626
17-35. Channel Options Parameter (OPT) 627
17-36. Channel Source Address Parameter (SRC) 629
17-37. A Count/B Count Parameter (A_B_CNT) 629
17-38. Channel Destination Address Parameter (DST) 630
17-39. Source B Index/Destination B Index Parameter (SRC_DST_BIDX) 630
17-40. Link Address/B Count Reload Parameter (LINK_BCNTRLD) 631
17-41. Source C Index/Destination C Index Parameter (SRC_DST_CIDX) 632
17-42. C Count Parameter (CCNT) 632
17-43. Revision ID Register (REVID) 636
17-44. EDMA3CC Configuration Register (CCCFG) 636
17-45. QDMA Channel *n* Mapping Register (QCHMAP*n*) 638
17-46. DMA Channel Queue Number Register *n* (DMAQNUM*n*) 639
17-47. QDMA Channel Queue Number Register (QDMAQNUM) 640
17-48. Event Missed Register (EMR) 641
17-49. Event Missed Clear Register (EMCR) 642
17-50. QDMA Event Missed Register (QEMR) 643
17-51. QDMA Event Missed Clear Register (QEMCR) 644
17-52. EDMA3CC Error Register (CCERR) 645
17-53. EDMA3CC Error Clear Register (CCERRCLR) 646
17-54. Error Evaluate Register (EEVAL) 647
17-55. DMA Region Access Enable Register for Region *m* (DRAE*m*) 648
17-56. QDMA Region Access Enable for Region *m* (QRAE*m*) 649
17-57. Event Queue Entry Registers (Q*x*E*y*) 650
17-58. Queue *n* Status Register (QSTAT*n*) 651
17-59. Queue Watermark Threshold A Register (QWMTHRA) 652
17-60. EDMA3CC Status Register (CCSTAT) 653
17-61. Event Register (ER) 655
17-62. Event Clear Register (ECR) 656
17-63. Event Set Register (ESR) 657

Copyright © 2013–2016, Texas Instruments Incorporated

17-64. Chained Event Register (CER) 658
17-65. Event Enable Register (EER) 659
17-66. Event Enable Clear Register (EECR) 660
17-67. Event Enable Set Register (EESR) 660
17-68. Secondary Event Register (SER) 661
17-69. Secondary Event Clear Register (SECR) 661
17-70. Interrupt Enable Register (IER) 662
17-71. Interrupt Enable Clear Register (IECR) 663
17-72. Interrupt Enable Set Register (IESR) 663
17-73. Interrupt Pending Register (IPR) 664
17-74. Interrupt Clear Register (ICR) 665
17-75. Interrupt Evaluate Register (IEVAL) 666
17-76. QDMA Event Register (QER) 667
17-77. QDMA Event Enable Register (QEER) 668
17-78. QDMA Event Enable Clear Register (QEECR) 669
17-79. QDMA Event Enable Set Register (QEESR) 669
17-80. QDMA Secondary Event Register (QSER) 670
17-81. QDMA Secondary Event Clear Register (QSECR) 671
17-82. Revision ID Register (REVID) 673
17-83. EDMA3TC Configuration Register (TCCFG) 674
17-84. EDMA3TC Channel Status Register (TCSTAT) 675
17-85. Error Status Register (ERRSTAT) 676
17-86. Error Enable Register (ERREN) 677
17-87. Error Clear Register (ERRCLR) 678
17-88. Error Details Register (ERRDET) 679
17-89. Error Interrupt Command Register (ERRCMD) 680
17-90. Read Command Rate Register (RDRATE) 681
17-91. Source Active Options Register (SAOPT) 682
17-92. Source Active Source Address Register (SASRC) 683
17-93. Source Active Count Register (SACNT) 683
17-94. Source Active Destination Address Register (SADST) 684
17-95. Source Active B-Index Register (SABIDX) 684
17-96. Source Active Memory Protection Proxy Register (SAMPPRXY) 685
17-97. Source Active Count Reload Register (SACNTRLD) 686
17-98. Source Active Source Address B-Reference Register (SASRCBREF) 686
17-99. Source Active Destination Address B-Reference Register (SADSTBREF) 687
17-100. Destination FIFO Set Count Reload Register (DFCNTRLD) 687
17-101. Destination FIFO Set Source Address B-Reference Register (DFSRCBREF) 688
17-102. Destination FIFO Set Destination Address B-Reference Register (DFDSTBREF) 688
17-103. Destination FIFO Options Register *n* (DFOPT*n*) 689
17-104. Destination FIFO Source Address Register *n* (DFSRC*n*) 690
17-105. Destination FIFO Count Register *n* (DFCNT*n*) 690
17-106. Destination FIFO Destination Address Register *n* (DFDST*n*) 691
17-107. Destination FIFO B-Index Register *n* (DFBIDX*n*) 691
17-108. Destination FIFO Memory Protection Proxy Register *n* (DFMPPRXY*n*) 692
18-1. EMIFA Functional Block Diagram 697
18-2. Timing Waveform of SDRAM PRE Command 701
18-3. EMIFA to 2M × 16 × 4 bank SDRAM Interface 702
18-4. EMIFA to 512K × 16 × 2 bank SDRAM Interface 702

Copyright © 2013–2016, Texas Instruments Incorporated

18-5. Timing Waveform for Basic SDRAM Read Operation 709
18-6. Timing Waveform for Basic SDRAM Write Operation 710
18-7. EMIFA Asynchronous Interface 712
18-8. EMIFA to 8-bit/16-bit Memory Interface 713
18-9. Common Asynchronous Interface 713
18-10. Timing Waveform of an Asynchronous Read Cycle in Normal Mode 718
18-11. Timing Waveform of an Asynchronous Write Cycle in Normal Mode 720
18-12. Timing Waveform of an Asynchronous Read Cycle in Select Strobe Mode 722
18-13. Timing Waveform of an Asynchronous Write Cycle in Select Strobe Mode 724
18-14. EMIFA to NAND Flash Interface 726
18-15. ECC Value for 8-Bit NAND Flash 728
18-16. EMIFA Reset Block Diagram 731
18-17. EMIFA PSC Block Diagram 736
18-18. Example Configuration Interface 739
18-19. SDRAM Timing Register (SDTIMR) 740
18-20. SDRAM Self Refresh Exit Timing Register (SDSRETR) 741
18-21. SDRAM Refresh Control Register (SDRCR) 741
18-22. SDRAM Configuration Register (SDCR) 742
18-23. Timing Waveform of an ASRAM Read 744
18-24. Timing Waveform of an ASRAM Write 745
18-25. Timing Waveform of an ASRAM Read with PCB Delays 747
18-26. Timing Waveform of an ASRAM Write with PCB Delays 748
18-27. Timing Waveform of a NAND Flash Read 753
18-28. Timing Waveform of a NAND Flash Command Write 755
18-29. Timing Waveform of a NAND Flash Address Write 755
18-30. Timing Waveform of a NAND Flash Data Write 756
18-31. Module ID Register (MIDR) 761
18-32. Asynchronous Wait Cycle Configuration Register (AWCCR) 761
18-33. SDRAM Configuration Register (SDCR) 763
18-34. SDRAM Refresh Control Register (SDRCR) 765
18-35. Asynchronous *n* Configuration Register (CE*n*CFG) 766
18-36. SDRAM Timing Register (SDTIMR) 768
18-37. SDRAM Self Refresh Exit Timing Register (SDSRETR) 769
18-38. EMIFA Interrupt Raw Register (INTRAW) 770
18-39. EMIFA Interrupt Mask Register (INTMSK) 771
18-40. EMIFA Interrupt Mask Set Register (INTMSKSET) 772
18-41. EMIFA Interrupt Mask Clear Register (INTMSKCLR) 773
18-42. NAND Flash Control Register (NANDFCR) 774
18-43. NAND Flash Status Register (NANDFSR) 776
18-44. NAND Flash *n* ECC Register (NANDF*n*ECC) 777
18-45. NAND Flash 4-Bit ECC LOAD Register (NAND4BITECCLOAD) 778
18-46. NAND Flash 4-Bit ECC Register 1 (NAND4BITECC1) 779
18-47. NAND Flash 4-Bit ECC Register 2 (NAND4BITECC2) 779
18-48. NAND Flash 4-Bit ECC Register 3 (NAND4BITECC3) 780
18-49. NAND Flash 4-Bit ECC Register 4 (NAND4BITECC4) 780
18-50. NAND Flash 4-Bit ECC Error Address Register 1 (NANDERRADD1) 781
18-51. NAND Flash 4-Bit ECC Error Address Register 2 (NANDERRADD2) 781
18-52. NAND Flash 4-Bit ECC Error Value Register 1 (NANDERRVAL1) 782
18-53. NAND Flash 4-Bit ECC Error Value Register 2 (NANDERRVAL2) 782

Copyright © 2013–2016, Texas Instruments Incorporated

19-1. GPIO Block Diagram 785
19-2. Revision ID Register (REVID) 794
19-3. GPIO Interrupt Per-Bank Enable Register (BINTEN) 795
19-4. GPIO Banks 0 and 1 Direction Register (DIR01) 796
19-5. GPIO Banks 2 and 3 Direction Register (DIR23) 796
19-6. GPIO Banks 4 and 5 Direction Register (DIR45) 796
19-7. GPIO Banks 6 and 7 Direction Register (DIR67) 796
19-8. GPIO Bank 8 Direction Register (DIR8) 797
19-9. GPIO Banks 0 and 1 Output Data Register (OUT_DATA01) 798
19-10. GPIO Banks 2 and 3 Output Data Register (OUT_DATA23) 798
19-11. GPIO Banks 4 and 5 Output Data Register (OUT_DATA45) 798
19-12. GPIO Banks 6 and 7 Output Data Register (OUT_DATA67) 798
19-13. GPIO Bank 8 Output Data Register (OUT_DATA8) 799
19-14. GPIO Banks 0 and 1 Set Data Register (SET_DATA01) 800
19-15. GPIO Banks 2 and 3 Set Data Register (SET_DATA23) 800
19-16. GPIO Banks 4 and 5 Set Data Register (SET_DATA45) 800
19-17. GPIO Banks 6 and 7 Set Data Register (SET_DATA67) 800
19-18. GPIO Bank 8 Set Data Register (SET_DATA8) 801
19-19. GPIO Banks 0 and 1 Clear Data Register (CLR_DATA01) 802
19-20. GPIO Banks 2 and 3 Clear Data Register (CLR_DATA23) 802
19-21. GPIO Banks 4 and 5 Clear Data Register (CLR_DATA45) 802
19-22. GPIO Banks 6 and 7 Clear Data Register (CLR_DATA67) 802
19-23. GPIO Bank 8 Clear Data Register (CLR_DATA8) 803
19-24. GPIO Banks 0 and 1 Input Data Register (IN_DATA01) 804
19-25. GPIO Banks 2 and 3 Input Data Register (IN_DATA23) 804
19-26. GPIO Banks 4 and 5 Input Data Register (IN_DATA45) 804
19-27. GPIO Banks 6 and 7 Input Data Register (IN_DATA67) 804
19-28. GPIO Bank 8 Input Data Register (IN_DATA8) 805
19-29. GPIO Banks 0 and 1 Set Rise Trigger Register (SET_RIS_TRIG01) 806
19-30. GPIO Banks 2 and 3 Set Rise Trigger Register (SET_RIS_TRIG23) 806
19-31. GPIO Banks 4 and 5 Set Rise Trigger Register (SET_RIS_TRIG45) 806
19-32. GPIO Banks 6 and 7 Set Rise Trigger Register (SET_RIS_TRIG67) 806
19-33. GPIO Bank 8 Set Rise Trigger Register (SET_RIS_TRIG8) 807
19-34. GPIO Banks 0 and 1 Clear Rise Trigger Register (CLR_RIS_TRIG01) 808
19-35. GPIO Banks 2 and 3 Clear Rise Trigger Register (CLR_RIS_TRIG23) 808
19-36. GPIO Banks 4 and 5 Clear Rise Trigger Register (CLR_RIS_TRIG45) 808
19-37. GPIO Banks 6 and 7 Clear Rise Trigger Register (CLR_RIS_TRIG67) 808
19-38. GPIO Bank 8 Clear Rise Trigger Register (CLR_RIS_TRIG8) 809
19-39. GPIO Banks 0 and 1 Set Rise Trigger Register (SET_FAL_TRIG01) 810
19-40. GPIO Banks 2 and 3 Set Rise Trigger Register (SET_FAL_TRIG23) 810
19-41. GPIO Banks 4 and 5 Set Rise Trigger Register (SET_FAL_TRIG45) 810
19-42. GPIO Banks 6 and 7 Set Rise Trigger Register (SET_FAL_TRIG67) 810
19-43. GPIO Bank 8 Set Rise Trigger Register (SET_FAL_TRIG8) 811
19-44. GPIO Banks 0 and 1 Clear Rise Trigger Register (CLR_FAL_TRIG01) 812
19-45. GPIO Banks 2 and 3 Clear Rise Trigger Register (CLR_FAL_TRIG23) 812
19-46. GPIO Banks 4 and 5 Clear Rise Trigger Register (CLR_FAL_TRIG45) 812
19-47. GPIO Banks 6 and 7 Clear Rise Trigger Register (CLR_FAL_TRIG67) 812
19-48. GPIO Bank 8 Clear Rise Trigger Register (CLR_FAL_TRIG8) 813
19-49. GPIO Banks 0 and 1 Interrupt Status Register (INTSTAT01) 814

Copyright © 2013–2016, Texas Instruments Incorporated

19-50. GPIO Banks 2 and 3 Interrupt Status Register (INTSTAT23) 814
19-51. GPIO Banks 4 and 5 Interrupt Status Register (INTSTAT45) 814
19-52. GPIO Banks 6 and 7 Interrupt Status Register (INTSTAT67) 814
19-53. GPIO Bank 8 Interrupt Status Register (INTSTAT8) 815
20-1. HPI Block Diagram 818
20-2. Example of Host-Processor Signal Connections 823
20-3. HPI Strobe and Select Logic 825
20-4. Multiplexed-Mode Host Read Cycle 827
20-5. Multiplexed-Mode Host Write Cycle 828
20-6. Multiplexed-Mode Single-Halfword HPIC Cycle (Read or Write) 829
20-7. $\overline{\text{UHPI_HRDY}}$ Behavior During an HPIC or HPIA Read Cycle in the Multiplexed Mode 830
20-8. $\overline{\text{UHPI_HRDY}}$ Behavior During a Data Read Operation in the Multiplexed Mode (Case 1: HPIA Write Cycle Followed by Nonautoincrement HPID Read Cycle) 830
20-9. $\overline{\text{UHPI_HRDY}}$ Behavior During a Data Read Operation in the Multiplexed Mode (Case 2: HPIA Write Cycle Followed by Autoincrement HPID Read Cycles) 830
20-10. $\overline{\text{UHPI_HRDY}}$ Behavior During an HPIC Write Cycle in the Multiplexed Mode 831
20-11. $\overline{\text{UHPI_HRDY}}$ Behavior During a Data Write Operation in the Multiplexed Mode (Case 1: No Autoincrementing) 831
20-12. $\overline{\text{UHPI_HRDY}}$ Behavior During a Data Write Operation in the Multiplexed Mode (Case 2: Autoincrementing Selected, FIFO Empty Before Write) 831
20-13. $\overline{\text{UHPI_HRDY}}$ Behavior During a Data Write Operation in the Multiplexed Mode (Case 3: Autoincrementing Selected, FIFO Not Empty Before Write) 832
20-14. FIFOs in the HPI 833
20-15. Host-to-CPU Interrupt State Diagram 838
20-16. CPU-to-Host Interrupt State Diagram 839
20-17. Revision Identification Register (REVID) 841
20-18. Power and Emulation Management Register (PWREMU_MGMT) 841
20-19. GPIO Enable Register (GPIO_EN) 842
20-20. GPIO Direction 1 Register (GPIO_DIR1) 843
20-21. GPIO Data 1 Register (GPIO_DAT1) 843
20-22. GPIO Direction 2 Register (GPIO_DIR2) 844
20-23. GPIO Data 2 Register (GPIO_DAT2) 845
20-24. Host Port Interface Control Register (HPIC)–Host Access Permissions 846
20-25. Host Port Interface Control Register (HPIC)–CPU Access Permissions 846
20-26. Host Port Interface Write Address Register (HPIAW) 848
20-27. Host Port Interface Read Address Register (HPIAR) 848
21-1. I2C Peripheral Block Diagram 851
21-2. Multiple I2C Modules Connected 852
21-3. Clocking Diagram for the I2C Peripheral 853
21-4. Synchronization of Two I2C Clock Generators During Arbitration 854
21-5. Bit Transfer on the I2C-Bus 855
21-6. I2C Peripheral START and STOP Conditions 855
21-7. I2C Peripheral Data Transfer 856
21-8. I2C Peripheral 7-Bit Addressing Format (FDF = 0, XA = 0 in ICMDR) 856
21-9. I2C Peripheral 10-Bit Addressing Format With Master-Transmitter Writing to Slave-Receiver (FDF = 0, XA = 1 in ICMDR) 857
21-10. I2C Peripheral Free Data Format (FDF = 1 in ICMDR) 857
21-11. I2C Peripheral 7-Bit Addressing Format With Repeated START Condition (FDF = 0, XA = 0 in ICMDR) 857
21-12. Arbitration Procedure Between Two Master-Transmitters 860
21-13. I2C Own Address Register (ICOAR) 865

Copyright © 2013–2016, Texas Instruments Incorporated

21-14. I2C Interrupt Mask Register (ICIMR) 866
21-15. I2C Interrupt Status Register (ICSTR) 867
21-16. I2C Clock Low-Time Divider Register (ICCLKL) 870
21-17. I2C Clock High-Time Divider Register (ICCLKH) 870
21-18. I2C Data Count Register (ICCNT) 871
21-19. I2C Data Receive Register (ICDRR) 872
21-20. I2C Slave Address Register (ICSAR) 873
21-21. I2C Data Transmit Register (ICDXR) 874
21-22. I2C Mode Register (ICMDR) 875
21-23. Block Diagram Showing the Effects of the Digital Loopback Mode (DLB) Bit 878
21-24. I2C Interrupt Vector Register (ICIVR) 879
21-25. I2C Extended Mode Register (ICEMDR) 880
21-26. I2C Prescaler Register (ICPSC) 881
21-27. I2C Revision Identification Register 1 (REVID1) 882
21-28. I2C Revision Identification Register 2 (REVID2) 882
21-29. I2C DMA Control Register (ICDMAC) 883
21-30. I2C Pin Function Register (ICPFUNC) 884
21-31. I2C Pin Direction Register (ICPDIR) 885
21-32. I2C Pin Data In Register (ICPDIN) 886
21-33. I2C Pin Data Out Register (ICPDOUT) 887
21-34. I2C Pin Data Set Register (ICPDSET) 888
21-35. I2C Pin Data Clear Register (ICPDCLR) 889
22-1. LCD Controller 891
22-2. Input and Output Clocks 892
22-3. Logical Data Path for Raster Controller 899
22-4. Frame Buffer Structure 900
22-5. 16-Entry Palette/Buffer Format (1, 2, 4, 12, 16 BPP) 901
22-6. 256-Entry Palette/Buffer Format (8 BPP) 902
22-7. 16-BPP Data Memory Organization (TFT Mode Only)—Little Endian 902
22-8. 12-BPP Data Memory Organization—Little Endian 903
22-9. 8-BPP Data Memory Organization 903
22-10. 4-BPP Data Memory Organization 903
22-11. 2-BPP Data Memory Organization 904
22-12. 1-BPP Data Memory Organization 904
22-13. Monochrome and Color Output 906
22-14. Raster Mode Display Format 907
22-15. LCD Revision Identification Register (REVID) 908
22-16. LCD Control Register (LCD_CTRL) 909
22-17. LCD Status Register (LCD_STAT) 911
22-18. LCD LIDD Control Register (LIDD_CTRL) 914
22-19. LCD LIDD CS*n* Configuration Register (LIDD_CS*n*_CONF) 916
22-20. LCD LIDD CS*n* Address Read/Write Register (LIDD_CS*n*_ADDR) 917
22-21. LCD LIDD CS*n* Data Read/Write Register (LIDD_CS*n*_DATA) 918
22-22. LCD Raster Control Register (RASTER_CTRL) 919
22-23. Monochrome Passive Mode Pixel Clock and Data Pin Timing 922
22-24. Color Passive Mode Pixel Clock and Data Pin Timing 922
22-25. Active Mode Pixel Clock and Data Pin Timing 923
22-26. TFT Alternate Signal Mapping Output 924
22-27. 12-Bit STN Data in Frame Buffer 925

Copyright © 2013–2016, Texas Instruments Incorporated

22-28. 16-Bit STN Data in Frame Buffer 925
22-29. 16-BPP STN Mode 925
22-30. LCD Raster Timing Register 0 (RASTER_TIMING_0) 926
22-31. LCD Raster Timing Register 1 (RASTER_TIMING_1) 928
22-32. Vertical Synchronization Pulse Width (VSW) - Active Mode 929
22-33. Vertical Front Porch (VFP) 930
22-34. Vertical Back Porch (VBP) 931
22-35. LCD Raster Timing Register 2 (RASTER_TIMING_2) 932
22-36. SYNC_CTRL = 0, IPC = 1 in TFT Mode 934
22-37. SYNC_CTRL = 1, SYNC_EDGE = 0, and IPC = 1 935
22-38. LCD Raster Subpanel Display Register (RASTER_SUBPANEL) 936
22-39. Subpanel Display: SPEN = 1, HOLS = 1 937
22-40. Subpanel Display: SPEN = 1, HOLS = 0 937
22-41. LCD DMA Control Register (LCDDMA_CTRL) 938
22-42. LCD DMA Frame Buffer *n* Base Address Register (LCDDMA_FB*n*_BASE) 939
22-43. LCD DMA Frame Buffer *n* Ceiling Address Register (LCDDMA_FB*n*_CEILING) 939
23-1. McASP Block Diagram 943
23-2. McASP to Parallel 2-Channel DACs 944
23-3. McASP to 6-Channel DAC and 2-Channel DAC 944
23-4. McASP to Digital Amplifier 945
23-5. McASP as Digital Audio Encoder 945
23-6. TDM Format–6 Channel TDM Example 946
23-7. TDM Format Bit Delays from Frame Sync 947
23-8. Inter-IC Sound (I2S) Format 947
23-9. Biphase-Mark Code (BMC) 948
23-10. S/PDIF Subframe Format 949
23-11. S/PDIF Frame Format 950
23-12. Definition of Bit, Word, and Slot 951
23-13. Bit Order and Word Alignment Within a Slot Examples 952
23-14. Definition of Frame and Frame Sync Width 953
23-15. Transmit Clock Generator Block Diagram 955
23-16. Receive Clock Generator Block Diagram 956
23-17. Frame Sync Generator Block Diagram 957
23-18. Individual Serializer and Connections Within McASP 958
23-19. Receive Format Unit 959
23-20. Transmit Format Unit 960
23-21. McASP I/O Pin Control Block Diagram 962
23-22. McASP I/O Pin to Control Register Mapping 963
23-23. Burst Frame Sync Mode 968
23-24. Transmit DMA Event (AXEVT) Generation in TDM Time Slots 971
23-25. DSP Service Time Upon Transmit DMA Event (AXEVT) 976
23-26. DSP Service Time Upon Receive DMA Event (AREVT) 978
23-27. DMA Events in an Audio Example–Two Events 980
23-28. McASP Audio FIFO (AFIFO) Block Diagram 981
23-29. Data Flow Through Transmit Format Unit 984
23-30. Data Flow Through Receive Format Unit 986
23-31. Audio Mute (AMUTE) Block Diagram 988
23-32. Transmit Clock Failure Detection Circuit Block Diagram 992
23-33. Receive Clock Failure Detection Circuit Block Diagram 993

Copyright © 2013–2016, Texas Instruments Incorporated

23-34. Serializers in Loopback Mode 994
23-35. Revision Identification Register (REV) 1000
23-36. Pin Function Register (PFUNC) 1001
23-37. Pin Direction Register (PDIR) 1003
23-38. Pin Data Output Register (PDOUT) 1005
23-39. Pin Data Input Register (PDIN) 1007
23-40. Pin Data Set Register (PDSET) 1009
23-41. Pin Data Clear Register (PDCLR) 1011
23-42. Global Control Register (GBLCTL) 1013
23-43. Audio Mute Control Register (AMUTE) 1015
23-44. Digital Loopback Control Register (DLBCTL) 1017
23-45. Digital Mode Control Register (DITCTL) 1018
23-46. Receiver Global Control Register (RGBLCTL) 1019
23-47. Receive Format Unit Bit Mask Register (RMASK) 1020
23-48. Receive Bit Stream Format Register (RFMT) 1021
23-49. Receive Frame Sync Control Register (AFSRCTL) 1023
23-50. Receive Clock Control Register (ACLKRCTL) 1024
23-51. Receive High-Frequency Clock Control Register (AHCLKRCTL) 1025
23-52. Receive TDM Time Slot Register (RTDM) 1026
23-53. Receiver Interrupt Control Register (RINTCTL) 1027
23-54. Receiver Status Register (RSTAT) 1028
23-55. Current Receive TDM Time Slot Registers (RSLOT) 1029
23-56. Receive Clock Check Control Register (RCLKCHK) 1030
23-57. Receiver DMA Event Control Register (REVTCTL) 1031
23-58. Transmitter Global Control Register (XGBLCTL) 1032
23-59. Transmit Format Unit Bit Mask Register (XMASK) 1033
23-60. Transmit Bit Stream Format Register (XFMT) 1034
23-61. Transmit Frame Sync Control Register (AFSXCTL) 1036
23-62. Transmit Clock Control Register (ACLKXCTL) 1037
23-63. Transmit High-Frequency Clock Control Register (AHCLKXCTL) 1038
23-64. Transmit TDM Time Slot Register (XTDM) 1039
23-65. Transmitter Interrupt Control Register (XINTCTL) 1040
23-66. Transmitter Status Register (XSTAT) 1041
23-67. Current Transmit TDM Time Slot Register (XSLOT) 1042
23-68. Transmit Clock Check Control Register (XCLKCHK) 1043
23-69. Transmitter DMA Event Control Register (XEVTCTL) 1044
23-70. Serializer Control Registers (SRCTL*n*) 1045
23-71. DIT Left Channel Status Registers (DITCSRA0-DITCSRA5) 1046
23-72. DIT Right Channel Status Registers (DITCSRB0-DITCSRB5) 1046
23-73. DIT Left Channel User Data Registers (DITUDRA0-DITUDRA5) 1047
23-74. DIT Right Channel User Data Registers (DITUDRB0-DITUDRB5) 1047
23-75. Transmit Buffer Registers (XBUF*n*) 1048
23-76. Receive Buffer Registers (RBUF*n*) 1048
23-77. AFIFO Revision Identification Register (AFIFOREV) 1049
23-78. Write FIFO Control Register (WFIFOCTL) 1050
23-79. Write FIFO Status Register (WFIFOSTS) 1051
23-80. Read FIFO Control Register (RFIFOCTL) 1052
23-81. Read FIFO Status Register (RFIFOSTS) 1053
24-1. McBSP Block Diagram 1056

Copyright © 2013–2016, Texas Instruments Incorporated

24-2. Clock and Frame Generation 1058
24-3. Transmit Data Clocking 1059
24-4. Receive Data Clocking 1059
24-5. Sample Rate Generator Block Diagram 1060
24-6. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 1 1063
24-7. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 3 1063
24-8. Digital Loopback Mode 1064
24-9. Programmable Frame Period and Width 1066
24-10. Dual-Phase Frame Example 1068
24-11. Single-Phase Frame of Four 8-Bit Elements 1069
24-12. Single-Phase Frame of One 32-Bit Element 1070
24-13. Data Delay 1070
24-14. 2-Bit Data Delay Used to Discard Framing Bit 1071
24-15. McBSP Standard Operation 1072
24-16. Receive Operation 1073
24-17. Transmit Operation 1073
24-18. Maximum Frame Frequency for Transmit and Receive 1074
24-19. Unexpected Frame Synchronization With (R/X)FIG = 0 1075
24-20. Unexpected Frame Synchronization With (R/X)FIG = 1 1076
24-21. Maximum Frame Frequency Operation With 8-Bit Data 1076
24-22. Data Packing at Maximum Frame Frequency With (R/X)FIG = 1 1077
24-23. Serial Port Receive Overrun 1078
24-24. Serial Port Receive Overrun Avoided 1078
24-25. Decision Tree Response to Receive Frame Synchronization Pulse 1079
24-26. Unexpected Receive Frame Synchronization Pulse 1080
24-27. Transmit With Data Overwrite 1080
24-28. Transmit Empty 1081
24-29. Transmit Empty Avoided 1081
24-30. Decision Tree Response to Transmit Frame Synchronization Pulse 1083
24-31. Unexpected Transmit Frame Synchronization Pulse 1083
24-32. McBSP Buffer FIFO (BFIFO) Block Diagram 1084
24-33. Companding Flow 1086
24-34. Companding Data Formats 1086
24-35. Transmit Data Companding Format in DXR 1086
24-36. Companding of Internal Data 1087
24-37. DX Timing for Multichannel Operation 1089
24-38. Alternating Between the Channels of Partition A and the Channels of Partition B 1091
24-39. Reassigning Channel Blocks Throughout a McBSP Data Transfer 1091
24-40. McBSP Data Transfer in the 8-Partition Mode 1092
24-41. Activity on McBSP Pins for the Possible Values of XMCM 1095
24-42. Data Receive Register (DRR) 1105
24-43. Data Transmit Register (DXR) 1105
24-44. Serial Port Control Register (SPCR) 1106
24-45. Receive Control Register (RCR) 1108
24-46. Transmit Control Register (XCR) 1110
24-47. Sample Rate Generator Register (SRGR) 1112
24-48. Multichannel Control Registers (MCR) 1113
24-49. Enhanced Receive Channel Enable Register *n* (RCERE*n*) 1117
24-50. Enhanced Transmit Channel Enable Register *n* (XCERE*n*) 1119

Copyright © 2013–2016, Texas Instruments Incorporated

24-51. Pin Control Register (PCR) 1121
24-52. BFIFO Revision Identification Register (BFIFOREV) 1123
24-53. Write FIFO Control Register (WFIFOCTL) 1124
24-54. Write FIFO Status Register (WFIFOSTS) 1125
24-55. Read FIFO Control Register (RFIFOCTL) 1126
24-56. Read FIFO Status Register (RFIFOSTS) 1127
25-1. MMC/SD Card Controller Block Diagram 1129
25-2. MMC/SD Controller Interface Diagram 1130
25-3. MMC Configuration and SD Configuration Diagram 1131
25-4. MMC/SD Controller Clocking Diagram 1132
25-5. MMC/SD Mode Write Sequence Timing Diagram 1133
25-6. MMC/SD Mode Read Sequence Timing Diagram 1134
25-7. FIFO Operation Diagram 1135
25-8. Little-Endian Access to MMCDXR/MMCDRR from the CPU or the EDMA 1136
25-9. FIFO Operation During Card Read Diagram 1138
25-10. FIFO Operation During Card Write Diagram 1140
25-11. MMC Card Identification Procedure 1147
25-12. SD Card Identification Procedure 1148
25-13. MMC/SD Mode Single-Block Write Operation 1150
25-14. MMC/SD Mode Single-Block Read Operation 1152
25-15. MMC/SD Multiple-Block Write Operation 1154
25-16. MMC/SD Mode Multiple-Block Read Operation 1156
25-17. MMC Control Register (MMCCTL) 1159
25-18. MMC Memory Clock Control Register (MMCCLK) 1160
25-19. MMC Status Register 0 (MMCST0) 1161
25-20. MMC Status Register 1 (MMCST1) 1163
25-21. MMC Interrupt Mask Register (MMCIM) 1164
25-22. MMC Response Time-Out Register (MMCTOR) 1166
25-23. MMC Data Read Time-Out Register (MMCTOD) 1167
25-24. MMC Block Length Register (MMCBLEN) 1168
25-25. MMC Number of Blocks Register (MMCNBLK) 1169
25-26. MMC Number of Blocks Counter Register (MMCNBLC) 1169
25-27. MMC Data Receive Register (MMCDRR) 1170
25-28. MMC Data Transmit Register (MMCDXR) 1170
25-29. MMC Command Register (MMCCMD) 1171
25-30. Command Format 1172
25-31. MMC Argument Register (MMCARGHL) 1173
25-32. MMC Response Register 0 and 1 (MMCRSP01) 1174
25-33. MMC Response Register 2 and 3 (MMCRSP23) 1174
25-34. MMC Response Register 4 and 5 (MMCRSP45) 1174
25-35. MMC Response Register 6 and 7 (MMCRSP67) 1174
25-36. MMC Data Response Register (MMCDRSP) 1176
25-37. MMC Command Index Register (MMCCIDX) 1176
25-38. SDIO Control Register (SDIOCTL) 1177
25-39. SDIO Status Register 0 (SDIOST0) 1178
25-40. SDIO Interrupt Enable Register (SDIOIEN) 1179
25-41. SDIO Interrupt Status Register (SDIOIST) 1179
25-42. MMC FIFO Control Register (MMCFIFOCTL) 1180
26-1. Real-Time Clock Block Diagram 1182

Copyright © 2013–2016, Texas Instruments Incorporated

26-2. 32-kHz Oscillator Counter Compensation 1186
26-3. Kick State Machine 1187
26-4. Second Register (SECOND) 1190
26-5. Minute Register (MINUTE) 1190
26-6. Hour Register (HOUR) 1191
26-7. Days Register (DAY) 1192
26-8. Month Register (MONTH) 1192
26-9. Year Register (YEAR) 1193
26-10. Day of the Week Register (DOTW) 1193
26-11. Alarm Second Register (ALARMSECOND) 1194
26-12. Alarm Minute Register (ALARMMINUTE) 1194
26-13. Alarm Hour Register (ALARMHOUR) 1195
26-14. Alarm Day Register (ALARMDAY) 1196
26-15. Alarm Month Register (ALARMMONTH) 1197
26-16. Alarm Year Register (ALARMYEAR) 1197
26-17. Control Register (CTRL) 1198
26-18. Status Register (STATUS) 1199
26-19. Interrupt Register (INTERRUPT) 1200
26-20. Compensation (LSB) Register (COMPLSB) 1201
26-21. Compensation (MSB) Register (COMPMSB) 1202
26-22. Oscillator Register (OSC) 1203
26-23. Scratch Registers (SCRATCH*n*) 1204
26-24. Kick Registers (KICK*n*R) 1204
27-1. SPI Block Diagram 1207
27-2. SPI 3-Pin Option 1213
27-3. SPI 4-Pin Option with $\overline{\text{SPIx_SCS[n]}}$ 1215
27-4. SPI 4-Pin Option with $\overline{\text{SPIx_ENA}}$ 1217
27-5. SPI 5-Pin Option with $\overline{\text{SPIx_ENA}}$ and $\overline{\text{SPIx_SCS[n]}}$ 1219
27-6. Format for Transmitting 12-Bit Word 1220
27-7. Format for 10-Bit Received Word 1220
27-8. Clock Mode with POLARITY = 0 and PHASE = 0 1221
27-9. Clock Mode with POLARITY = 0 and PHASE = 1 1222
27-10. Clock Mode with POLARITY = 1 and PHASE = 0 1222
27-11. Clock Mode with POLARITY = 1 and PHASE = 1 1222
27-12. Five Bits per Character (5-Pin Option) 1223
27-13. SPI 3-Pin Master Mode with WDELAY 1228
27-14. SPI 4-Pin with $\overline{\text{SPIx_SCS[n]}}$ Mode with T2CDELAY, WDELAY, and C2TDELAY 1229
27-15. SPI 4-Pin with $\overline{\text{SPIx_ENA}}$ Mode Demonstrating T2EDELAY and WDELAY 1230
27-16. SPI 5-Pin Mode Demonstrating T2CDELAY, T2EDELAY, and WDELAY 1232
27-17. SPI 5-Pin Mode Demonstrating C2TDELAY and C2EDELAY 1233
27-18. SPI Global Control Register 0 (SPIGCR0) 1234
27-19. SPI Global Control Register 1 (SPIGCR1) 1235
27-20. SPI Interrupt Register (SPIINT0) 1237
27-21. SPI Interrupt Level Register (SPILVL) 1239
27-22. SPI Flag Register (SPIFLG) 1240
27-23. SPI Pin Control Register 0 (SPIPC0) 1242
27-24. SPI Pin Control Register 1 (SPIPC1) 1243
27-25. SPI Pin Control Register 2 (SPIPC2) 1244
27-26. SPI Pin Control Register 3 (SPIPC3) 1245

Copyright © 2013–2016, Texas Instruments Incorporated

27-27. SPI Pin Control Register 4 (SPIPC4) 1246
27-28. SPI Pin Control Register 5 (SPIPC5) 1247
27-29. SPI Data Register 0 (SPIDAT0) 1248
27-30. SPI Data Register 1 (SPIDAT1) 1249
27-31. SPI Buffer Register (SPIBUF) 1250
27-32. SPI Emulation Register (SPIEMU) 1252
27-33. SPI Delay Register (SPIDELAY) 1253
27-34. Example: $t_{C2TDELAY}$ = 8 SPI Module Clock Cycles 1254
27-35. Example: $t_{T2CDELAY}$ = 4 SPI Module Clock Cycles 1255
27-36. Transmit-Data-Finished-to-$\overline{\text{SPIx_ENA}}$-Inactive-Timeout 1255
27-37. Chip-Select-Active-to-$\overline{\text{SPIx_ENA}}$-Signal-Active-Timeout 1255
27-38. SPI Default Chip Select Register (SPIDEF) 1256
27-39. SPI Data Format Register (SPIFMT*n*) 1257
27-40. SPI Interrupt Vector Register 1 (INTVEC1) 1259
28-1. Timer Block Diagram 1262
28-2. Timer Clock Source Block Diagram 1263
28-3. 64-Bit Timer Mode Block Diagram 1264
28-4. Dual 32-Bit Timers Chained Mode Block Diagram 1267
28-5. Dual 32-Bit Timers Chained Mode Example 1267
28-6. Dual 32-Bit Timers Unchained Mode Block Diagram 1269
28-7. Dual 32-Bit Timers Unchained Mode Example 1270
28-8. 32-Bit Timer Counter Overflow Example 1273
28-9. Watchdog Timer Mode Block Diagram 1275
28-10. Watchdog Timer Operation State Diagram 1275
28-11. Timer Operation in Pulse Mode (CP*n* = 0) 1277
28-12. Timer Operation in Clock Mode (CP*n* = 1) 1277
28-13. Revision ID Register (REVID) 1281
28-14. Emulation Management Register (EMUMGT) 1281
28-15. GPIO Interrupt Control and Enable Register (GPINTGPEN) 1282
28-16. GPIO Data and Direction Register (GPDATGPDIR) 1283
28-17. Timer Counter Register 12 (TIM12) 1284
28-18. Timer Counter Register 34 (TIM34) 1284
28-19. Timer Period Register 12 (PRD12) 1285
28-20. Timer Period Register 34 (PRD34) 1285
28-21. Timer Control Register (TCR) 1286
28-22. Timer Global Control Register (TGCR) 1288
28-23. Watchdog Timer Control Register (WDTCR) 1289
28-24. Timer Reload Register 12 (REL12) 1290
28-25. Timer Reload Register 34 (REL34) 1290
28-26. Timer Capture Register 12 (CAP12) 1291
28-27. Timer Capture Register 34 (CAP34) 1291
28-28. Timer Interrupt Control and Status Register (INTCTLSTAT) 1292
28-29. Timer Compare Register (CMP*n*) 1293
29-1. UART Block Diagram 1296
29-2. UART Clock Generation Diagram 1297
29-3. Relationships Between Data Bit, BCLK, and UART Input Clock 1298
29-4. UART Protocol Formats 1300
29-5. UART Interface Using Autoflow Diagram 1303
29-6. Autoflow Functional Timing Waveforms for $\overline{\text{UARTn_RTS}}$ 1304

Copyright © 2013–2016, Texas Instruments Incorporated

29-7. Autoflow Functional Timing Waveforms for UARTn_CTS 1304
29-8. UART Interrupt Request Enable Paths 1306
29-9. Receiver Buffer Register (RBR) 1309
29-10. Transmitter Holding Register (THR) 1310
29-11. Interrupt Enable Register (IER) 1311
29-12. Interrupt Identification Register (IIR) 1312
29-13. FIFO Control Register (FCR) 1314
29-14. Line Control Register (LCR) 1315
29-15. Modem Control Register (MCR) 1317
29-16. Line Status Register (LSR) 1318
29-17. Modem Status Register (MSR) 1321
29-18. Scratch Pad Register (SCR) 1322
29-19. Divisor LSB Latch (DLL) 1323
29-20. Divisor MSB Latch (DLH) 1323
29-21. Revision Identification Register 1 (REVID1) 1324
29-22. Revision Identification Register 2 (REVID2) 1324
29-23. Power and Emulation Management Register (PWREMU_MGMT) 1325
29-24. Mode Definition Register (MDR) 1326
30-1. uPP Functional Block Diagram 1329
30-2. Data Flow for Single-Channel Receive Mode 1329
30-3. Data Flow for Single-Channel Transmit Mode 1329
30-4. Data Flow for Digital Loopback (DLB) Mode (Duplex Mode 0) 1330
30-5. Data Flow for Single-Channel Transmit with Data Interleave 1330
30-6. Clock Generation for a Channel Configured in Transmit Mode 1331
30-7. Clock Generation for a Channel Configured in Receive Mode 1331
30-8. Structure of DMA Window and Lines in Memory 1334
30-9. Signal Timing for uPP Channel in Receive Mode with Single Data Rate 1337
30-10. Signal Timing for uPP Channel in Transmit Mode with Single Data Rate 1338
30-11. Signal Timing for uPP Channel in Receive Mode with Double Data Rate 1338
30-12. Signal Timing for uPP Channel in Transmit Mode with Double Data Rate 1338
30-13. Signal Timing for uPP Channel in Receive Mode with Double Data Rate and Data Interleave Enabled (via UPCTL.DDRDEMUX) 1339
30-14. Signal Timing for uPP Channel in Transmit Mode with Double Data Rate and Data Interleave Enabled (via UPCTL.DDRDEMUX) 1339
30-15. Signal Timing for uPP Channel in Transmit Mode with Single Data Rate and Data Interleave Enabled (via UPCTL.SDRTXIL) 1339
30-16. uPP Peripheral Identification Register (UPPID) 1348
30-17. uPP Peripheral Control Register (UPPCR) 1349
30-18. uPP Digital Loopback Register (UPDLB) 1350
30-19. uPP Channel Control Register (UPCTL) 1351
30-20. uPP Interface Configuration Register (UPICR) 1353
30-21. uPP Interface Idle Value Register (UPIVR) 1355
30-22. uPP Threshold Configuration Register (UPTCR) 1356
30-23. uPP Interrupt Raw Status Register (UPISR) 1357
30-24. uPP Interrupt Enabled Status Register (UPIER) 1359
30-25. uPP Interrupt Enable Set Register (UPIES) 1361
30-26. uPP Interrupt Enable Clear Register (UPIEC) 1363
30-27. uPP End of Interrupt Register (UPEOI) 1365
30-28. uPP DMA Channel I Descriptor 0 Register (UPID0) 1365
30-29. uPP DMA Channel I Descriptor 1 Register (UPID1) 1366

Copyright © 2013–2016, Texas Instruments Incorporated

30-30. uPP DMA Channel I Descriptor 2 Register (UPID2) 1366
30-31. uPP DMA Channel I Status 0 Register (UPIS0) 1367
30-32. uPP DMA Channel I Status 1 Register (UPIS1) 1367
30-33. uPP DMA Channel I Status 2 Register (UPIS2) 1368
30-34. uPP DMA Channel Q Descriptor 0 Register (UPQD0) 1369
30-35. uPP DMA Channel Q Descriptor 1 Register (UPQD1) 1369
30-36. uPP DMA Channel Q Descriptor 2 Register (UPID2) 1370
30-37. uPP DMA Channel Q Status 0 Register (UPQS0) 1371
30-38. uPP DMA Channel Q Status 1 Register (UPQS1) 1371
30-39. uPP DMA Channel Q Status 2 Register (UPQS2) 1372
31-1. Functional Block Diagram 1374
31-2. USB Clocking Diagram 1375
31-3. Interrupt Service Routine Flow Chart 1379
31-4. CPU Actions at Transfer Phases 1384
31-5. Sequence of Transfer 1384
31-6. Service Endpoint 0 Flow Chart 1386
31-7. IDLE Mode Flow Chart 1387
31-8. TX Mode Flow Chart 1388
31-9. RX Mode Flow Chart 1389
31-10. Setup Phase of a Control Transaction Flow Chart 1399
31-11. IN Data Phase Flow Chart 1401
31-12. OUT Data Phase Flow Chart 1403
31-13. Completion of SETUP or OUT Data Phase Flow Chart 1405
31-14. Completion of IN Data Phase Flow Chart 1407
31-15. USB Controller Block Diagram 1414
31-16. Host Packet Descriptor Layout 1417
31-17. Host Buffer Descriptor Layout 1420
31-18. Teardown Descriptor Layout 1422
31-19. Relationship Between Memory Regions and Linking RAM 1425
31-20. High-Level Transmit and Receive Data Transfer Example 1431
31-21. Transmit Descriptors and Queue Status Configuration 1432
31-22. Transmit USB Data Flow Example (Initialization) 1433
31-23. Transmit USB Data Flow Example (Completion) 1434
31-24. Receive Descriptors and Queue Status Configuration 1435
31-25. Receive USB Data Flow Example (Initialization) 1435
31-26. Receive USB Data Flow Example (Completion) 1436
31-27. Revision Identification Register (REVID) 1460
31-28. Control Register (CTRLR) 1460
31-29. Status Register (STATR) 1461
31-30. Emulation Register (EMUR) 1461
31-31. Mode Register (MODE) 1462
31-32. Auto Request Register (AUTOREQ) 1464
31-33. SRP Fix Time Register (SRPFIXTIME) 1465
31-34. Teardown Register (TEARDOWN) 1465
31-35. USB Interrupt Source Register (INTSRCR) 1466
31-36. USB Interrupt Source Set Register (INTSETR) 1467
31-37. USB Interrupt Source Clear Register (INTCLRR) 1468
31-38. USB Interrupt Mask Register (INTMSKR) 1469
31-39. USB Interrupt Mask Set Register (INTMSKSETR) 1470

Copyright © 2013–2016, Texas Instruments Incorporated

31-40. USB Interrupt Mask Clear Register (INTMSKCLRR) 1471
31-41. USB Interrupt Source Masked Register (INTMASKEDR) 1472
31-42. USB End of Interrupt Register (EOIR) 1473
31-43. Generic RNDIS EP1 Size Register (GENRNDISSZ1) 1473
31-44. Generic RNDIS EP2 Size Register (GENRNDISSZ2) 1474
31-45. Generic RNDIS EP3 Size Register (GENRNDISSZ3) 1474
31-46. Generic RNDIS EP4 Size Register (GENRNDISSZ4) 1475
31-47. Function Address Register (FADDR) 1475
31-48. Power Management Register (POWER) 1476
31-49. Interrupt Register for Endpoint 0 Plus Tx Endpoints 1 to 4 (INTRTX) 1477
31-50. Interrupt Register for Receive Endpoints 1 to 4 (INTRRX) 1478
31-51. Interrupt Enable Register for INTRTX (INTRTXE) 1479
31-52. Interrupt Enable Register for INTRRX (INTRRXE) 1479
31-53. Interrupt Register for Common USB Interrupts (INTRUSB) 1480
31-54. Interrupt Enable Register for INTRUSB (INTRUSBE) 1481
31-55. Frame Number Register (FRAME) 1481
31-56. Index Register for Selecting the Endpoint Status and Control Registers (INDEX) 1482
31-57. Register to Enable the USB 2.0 Test Modes (TESTMODE) 1482
31-58. Maximum Packet Size for Peripheral/Host Transmit Endpoint (TXMAXP) 1483
31-59. Control Status Register for Endpoint 0 in Peripheral Mode (PERI_CSR0) 1484
31-60. Control Status Register for Endpoint 0 in Host Mode (HOST_CSR0) 1485
31-61. Control Status Register for Peripheral Transmit Endpoint (PERI_TXCSR) 1486
31-62. Control Status Register for Host Transmit Endpoint (HOST_TXCSR) 1487
31-63. Maximum Packet Size for Peripheral Host Receive Endpoint (RXMAXP) 1488
31-64. Control Status Register for Peripheral Receive Endpoint (PERI_RXCSR) 1489
31-65. Control Status Register for Host Receive Endpoint (HOST_RXCSR) 1490
31-66. Count 0 Register (COUNT0) 1491
31-67. Receive Count Register (RXCOUNT) 1491
31-68. Type Register (Host mode only) (HOST_TYPE0) 1492
31-69. Transmit Type Register (Host mode only) (HOST_TXTYPE) 1492
31-70. NAKLimit0 Register (Host mode only) (HOST_NAKLIMIT0) 1493
31-71. Transmit Interval Register (Host mode only) (HOST_TXINTERVAL) 1493
31-72. Receive Type Register (Host mode only) (HOST_RXTYPE) 1494
31-73. Receive Interval Register (Host mode only) (HOST_RXINTERVAL) 1495
31-74. Configuration Data Register (CONFIGDATA) 1496
31-75. Transmit and Receive FIFO Register for Endpoint 0 (FIFO0) 1497
31-76. Transmit and Receive FIFO Register for Endpoint 1 (FIFO1) 1497
31-77. Transmit and Receive FIFO Register for Endpoint 2 (FIFO2) 1498
31-78. Transmit and Receive FIFO Register for Endpoint 3 (FIFO3) 1498
31-79. Transmit and Receive FIFO Register for Endpoint 4 (FIFO4) 1499
31-80. Device Control Register (DEVCTL) 1499
31-81. Transmit Endpoint FIFO Size (TXFIFOSZ) 1500
31-82. Receive Endpoint FIFO Size (RXFIFOSZ) 1500
31-83. Transmit Endpoint FIFO Address (TXFIFOADDR) 1501
31-84. Receive Endpoint FIFO Address (RXFIFOADDR) 1501
31-85. Hardware Version Register (HWVERS) 1502
31-86. Transmit Function Address (TXFUNCADDR) 1503
31-87. Transmit Hub Address (TXHUBADDR) 1503
31-88. Transmit Hub Port (TXHUBPORT) 1503

Copyright © 2013–2016, Texas Instruments Incorporated

31-89. Receive Function Address (RXFUNCADDR) 1504
31-90. Receive Hub Address (RXHUBADDR) 1504
31-91. Receive Hub Port (RXHUBPORT) 1504
31-92. CDMA Revision Identification Register (DMAREVID) 1505
31-93. CDMA Teardown Free Descriptor Queue Control Register (TDFDQ) 1505
31-94. CDMA Emulation Control Register (DMAEMU) 1506
31-95. CDMA Transmit Channel *n* Global Configuration Registers (TXGCR[*n*]) 1506
31-96. CDMA Receive Channel *n* Global Configuration Registers (RXGCR[*n*]) 1507
31-97. Receive Channel *n* Host Packet Configuration Registers A (RXHPCRA[*n*]) 1508
31-98. Receive Channel *n* Host Packet Configuration Registers B (RXHPCRB[*n*]) 1509
31-99. CDMA Scheduler Control Register (DMA_SCHED_CTRL) 1510
31-100. CDMA Scheduler Table Word *n* Registers (WORD[*n*]) 1510
31-101. Queue Manager Revision Identification Register (QMGRREVID) 1512
31-102. Queue Manager Queue Diversion Register (DIVERSION) 1512
31-103. Queue Manager Free Descriptor/Buffer Starvation Count Register 0 (FDBSC0) 1513
31-104. Queue Manager Free Descriptor/Buffer Starvation Count Register 1 (FDBSC1) 1514
31-105. Queue Manager Free Descriptor/Buffer Starvation Count Register 2 (FDBSC2) 1515
31-106. Queue Manager Free Descriptor/Buffer Starvation Count Register 3 (FDBSC3) 1516
31-107. Queue Manager Linking RAM Region 0 Base Address Register (LRAM0BASE) 1516
31-108. Queue Manager Linking RAM Region 0 Size Register (LRAM0SIZE) 1517
31-109. Queue Manager Linking RAM Region 1 Base Address Register (LRAM1BASE) 1517
31-110. Queue Manager Queue Pending Register 0 (PEND0) 1518
31-111. Queue Manager Queue Pending Register 1 (PEND1) 1518
31-112. Queue Manager Memory Region *R* Base Address Registers (QMEMRBASE[*R*]) 1519
31-113. Queue Manager Memory Region *R* Control Registers (QMEMRCTRL[*R*]) 1520
31-114. Queue Manager Queue *N* Control Register D (CTRLD[*N*]) 1521
31-115. Queue Manager Queue *N* Status Register A (QSTATA[*N*]) 1522
31-116. Queue Manager Queue *N* Status Register B (QSTATB[*N*]) 1522
31-117. Queue Manager Queue *N* Status Register C (QSTATC[*N*]) 1523
32-1. Input and Output Channels of VPIF 1525
32-2. Video Port Interface (VPIF) Block Diagram 1526
32-3. VPIF Architecture Block Diagram 1527
32-4. Interlaced Video 1530
32-5. Progressive Video 1531
32-6. Memory Storage Modes for Interlaced Video 1532
32-7. Functional Image of Raw Data Capturing Mode 1535
32-8. Raw Capture Progressive Mode 1536
32-9. Raw Capture Interlaced Mode 1537
32-10. Stuffing Manner in Storage Memory 1537
32-11. VBI Result Data Transmit Image for Interlaced Image 1538
32-12. Module Performance with Emulation Suspend Signal 1543
32-13. Emulation Suspend Function on Channels 2 and 3 (Transmit) 1544
32-14. Method for Turning off Module Channel 1545
32-15. Clock Control on Video Input and Output with SDTV Encoding 1546
32-16. Clock Control on Video Input and Output with HDTV Encoding 1548
32-17. Clock Control on Video Input and Output with HDTV Encoding 1549
32-18. VPIF Revision ID Register (REVID) 1553
32-19. Channel 0 Control Register (C0CTRL) 1553
32-20. Channel 1 Control Register (C1CTRL) 1555

Copyright © 2013–2016, Texas Instruments Incorporated

32-21. Channel 2 Control Register (C2CTRL) 1556
32-22. Channel 3 Control Register (C3CTRL) 1558
32-23. Interrupt Enable Register (INTEN) 1560
32-24. Interrupt Enable Set Register (INTSET) 1561
32-25. Interrupt Enable Clear Register (INTCLR) 1562
32-26. Interrupt Status Register (INTSTAT) 1563
32-27. Interrupt Status Clear Register (INTSTATCLR) 1564
32-28. Emulation Suspend Control Register (EMUCTRL) 1565
32-29. DMA Size Control Register (REQSIZE) 1565
32-30. Channel *n* Top Field Luminance Address Register (C*n*TLUMA) 1566
32-31. Channel *n* Bottom Field Luminance Address Register (C*n*BLUMA) 1566
32-32. Channel *n* Top Field Chrominance Address Register (C*n*TCHROMA) 1567
32-33. Channel *n* Bottom Field Chrominance Address Register (C*n*BCHROMA) 1567
32-34. Channel *n* Top Field Horizontal Ancillary Address Register (C*n*THANC) 1568
32-35. Channel *n* Bottom Field Horizontal Ancillary Address Register (C*n*BHANC) 1568
32-36. Channel *n* Top Field Vertical Ancillary Address Register (C*n*TVANC) 1569
32-37. Channel *n* Bottom Field Vertical Ancillary Data Buffer Start Address Register (C*n*BVANC) 1569
32-38. Channel *n* Image Address Offset Register (C*n*IMGOFFSET) 1570
32-39. Channel *n* Horizontal Ancillary Address Offset Register (C*n*HANCOFFSET) 1570
32-40. Channel *n* Horizontal Size Configuration Register (C*n*HCFG) 1571
32-41. Channel *n* Vertical Size Configuration 0 Register (C*n*VCFG0) 1572
32-42. Channel *n* Vertical Data Size Configuration 1 Register (C*n*VCFG1) 1572
32-43. Channel *n* Vertical Size Configuration 2 Register (C*n*VCFG2) 1573
32-44. Channel *n* Vertical Image Size Register (C*n*VSIZE) 1573
32-45. Channel *n* Horizontal Size Configuration Register (C*n*HCFG) 1574
32-46. Channel *n* Vertical Size Configuration 0 Register (C*n*VCFG0) 1575
32-47. Channel *n* Vertical Size Configuration 1 Register (C*n*VCFG1) 1575
32-48. Channel *n* Vertical Size Configuration 2 Register (C*n*VCFG2) 1576
32-49. Channel *n* Vertical Image Size Register (C*n*VSIZE) 1576
32-50. Channel *n* Top Field Horizontal Ancillary Position Register (C*n*THANCPOS) 1577
32-51. Channel *n* Top Field Horizontal Ancillary Size Register (C*n*THANCSIZE) 1578
32-52. Channel *n* Bottom Field Horizontal Ancillary Position Register (C*n*BHANCPOS) 1579
32-53. Channel *n* Bottom Field Horizontal Ancillary Size Register (C*n*BHANCSIZE) 1580
32-54. Channel *n* Top Field Vertical Ancillary Position Register (C*n*TVANCPOS) 1581
32-55. Channel *n* Top Field Vertical Ancillary Size Register (C*n*TVANCSIZE) 1582
32-56. Channel *n* Bottom Field Vertical Ancillary Position Register (C*n*BVANCPOS) 1583
32-57. Channel *n* Bottom Field Vertical Ancillary Size Register (C*n*BVANCSIZE) 1584

Copyright © 2013–2016, Texas Instruments Incorporated

List of Tables

2-1. Exception Vector Table for ARM 78
2-2. Different Address Types in ARM System 80
3-1. AM1806 ARM Microprocessor System Interconnect Matrix 83
5-1. MPU Memory Regions 89
5-2. MPU Default Configuration 89
5-3. Device Master Settings 90
5-4. Request Type Access Controls 91
5-5. MPU_BOOTCFG_ERR Interrupt Sources 93
5-6. Memory Protection Unit 1 (MPU1) Registers 94
5-7. Memory Protection Unit 2 (MPU2) Registers 94
5-8. Revision ID Register (REVID) Field Descriptions 96
5-9. Configuration Register (CONFIG) Field Descriptions 96
5-10. Interrupt Raw Status/Set Register (IRAWSTAT) Field Descriptions 97
5-11. Interrupt Enable Status/Clear Register (IENSTAT) Field Descriptions 98
5-12. Interrupt Enable Set Register (IENSET) Field Descriptions 99
5-13. Interrupt Enable Clear Register (IENCLR) Field Descriptions 99
5-14. Fixed Range Memory Protection Page Attributes Register (FXD_MPPA) Field Descriptions 101
5-15. MPU1 Programmable Range *n* Start Address Register (PROG*n*_MPSAR) Field Descriptions 102
5-16. MPU2 Programmable Range *n* Start Address Register (PROG*n*_MPSAR) Field Descriptions 102
5-17. MPU1 Programmable Range *n* End Address Register (PROG*n*_MPEAR) Field Descriptions 103
5-18. MPU2 Programmable Range *n* End Address Register (PROG*n*_MPEAR) Field Descriptions 103
5-19. Programmable Range Memory Protection Page Attributes Register (PROG*n*_MPPA) Field Descriptions 104
5-20. Fault Address Register (FLTADDRR) Field Descriptions 105
5-21. Fault Status Register (FLTSTAT) Field Descriptions 106
5-22. Fault Clear Register (FLTCLR) Field Descriptions 107
6-1. Device Clock Inputs 109
6-2. System Clock Domains 109
6-3. Example PLL Frequencies 112
6-4. USB Clock Multiplexing Options 112
6-5. DDR2/mDDR Memory Controller MCLK Frequencies 114
6-6. EMIFA Frequencies 115
6-7. uPP Transmit Clock Selection 116
6-8. Peripherals 118
7-1. System PLLC Output Clocks 122
7-2. PLL Controller 0 (PLLC0) Registers 125
7-3. PLL Controller 1 (PLLC1) Registers 126
7-4. PLLC0 Revision Identification Register (REVID) Field Descriptions 126
7-5. PLLC1 Revision Identification Register (REVID) Field Descriptions 127
7-6. Reset Type Status Register (RSTYPE) Field Descriptions 127
7-7. Reset Control Register (RSCTRL) Field Descriptions 128
7-8. PLLC0 Control Register (PLLCTL) Field Descriptions 129
7-9. PLLC1 Control Register (PLLCTL) Field Descriptions 130
7-10. PLLC0 OBSCLK Select Register (OCSEL) Field Descriptions 131
7-11. PLLC1 OBSCLK Select Register (OCSEL) Field Descriptions 132
7-12. PLL Multiplier Control Register (PLLM) Field Descriptions 133
7-13. PLLC0 Pre-Divider Control Register (PREDIV) Field Descriptions 133
7-14. PLLC0 Divider 1 Register (PLLDIV1) Field Descriptions 134

Copyright © 2013–2016, Texas Instruments Incorporated

7-15. PLLC1 Divider 1 Register (PLLDIV1) Field Descriptions 134
7-16. PLLC0 Divider 2 Register (PLLDIV2) Field Descriptions 135
7-17. PLLC1 Divider 2 Register (PLLDIV2) Field Descriptions 135
7-18. PLLC0 Divider 3 Register (PLLDIV3) Field Descriptions 136
7-19. PLLC1 Divider 3 Register (PLLDIV3) Field Descriptions 136
7-20. PLLC0 Divider 4 Register (PLLDIV4) Field Descriptions 137
7-21. PLLC0 Divider 5 Register (PLLDIV5) Field Descriptions 137
7-22. PLLC0 Divider 6 Register (PLLDIV6) Field Descriptions 138
7-23. PLLC0 Divider 7 Register (PLLDIV7) Field Descriptions 138
7-24. PLLC0 Oscillator Divider 1 Register (OSCDIV) Field Descriptions 139
7-25. PLLC1 Oscillator Divider 1 Register (OSCDIV) Field Descriptions 139
7-26. PLL Post-Divider Control Register (POSTDIV) Field Descriptions 140
7-27. PLL Controller Command Register (PLLCMD) Field Descriptions 140
7-28. PLL Controller Status Register (PLLSTAT) Field Descriptions 141
7-29. PLLC0 Clock Align Control Register (ALNCTL) Field Descriptions 142
7-30. PLLC1 Clock Align Control Register (ALNCTL) Field Descriptions 143
7-31. PLLC0 PLLDIV Ratio Change Status Register (DCHANGE) Field Descriptions 144
7-32. PLLC1 PLLDIV Ratio Change Status Register (DCHANGE) Field Descriptions 145
7-33. PLLC0 Clock Enable Control Register (CKEN) Field Descriptions 146
7-34. PLLC1 Clock Enable Control Register (CKEN) Field Descriptions 146
7-35. PLLC0 Clock Status Register (CKSTAT) Field Descriptions 147
7-36. PLLC1 Clock Status Register (CKSTAT) Field Descriptions 148
7-37. PLLC0 SYSCLK Status Register (SYSTAT) Field Descriptions 149
7-38. PLLC1 SYSCLK Status Register (SYSTAT) Field Descriptions 150
7-39. Emulation Performance Counter 0 Register (EMUCNT0) Field Descriptions 151
7-40. Emulation Performance Counter 1 Register (EMUCNT1) Field Descriptions 151
8-1. PSC0 Default Module Configuration 154
8-2. PSC1 Default Module Configuration 154
8-3. Module States 157
8-4. IcePick Emulation Commands 159
8-5. PSC Interrupt Events 159
8-6. Power and Sleep Controller 0 (PSC0) Registers 162
8-7. Power and Sleep Controller 1 (PSC1) Registers 162
8-8. Revision Identification Register (REVID) Field Descriptions 163
8-9. Interrupt Evaluation Register (INTEVAL) Field Descriptions 163
8-10. PSC0 Module Error Pending Register 0 (MERRPR0) Field Descriptions 164
8-11. PSC0 Module Error Clear Register 0 (MERRCR0) Field Descriptions 165
8-12. Power Error Pending Register (PERRPR) Field Descriptions 166
8-13. Power Error Clear Register (PERRCR) Field Descriptions 166
8-14. Power Domain Transition Command Register (PTCMD) Field Descriptions 167
8-15. Power Domain Transition Status Register (PTSTAT) Field Descriptions 168
8-16. Power Domain 0 Status Register (PDSTAT0) Field Descriptions 169
8-17. Power Domain 1 Status Register (PDSTAT1) Field Descriptions 170
8-18. Power Domain 0 Control Register (PDCTL0) Field Descriptions 171
8-19. Power Domain 1 Control Register (PDCTL1) Field Descriptions 172
8-20. Power Domain 0 Configuration Register (PDCFG0) Field Descriptions 173
8-21. Power Domain 1 Configuration Register (PDCFG1) Field Descriptions 174
8-22. Module Status *n* Register (MDSTAT*n*) Field Descriptions 175
8-23. PSC0 Module Control *n* Register (MDCTL*n*) Field Descriptions 176

Copyright © 2013–2016, Texas Instruments Incorporated

8-24. PSC1 Module Control *n* Register (MDCTL*n*) Field Descriptions 177
9-1. Power Management Features 180
10-1. Master IDs 193
10-2. Default Master Priority 194
10-3. System Configuration Module 0 (SYSCFG0) Registers 195
10-4. System Configuration Module 1 (SYSCFG1) Registers 196
10-5. Revision Identification Register (REVID) Field Descriptions 196
10-6. Device Identification Register 0 (DEVIDR0) Field Descriptions 197
10-7. Boot Configuration Register (BOOTCFG) Field Descriptions 197
10-8. Chip Revision Identification Register (CHIPREVIDR) Field Descriptions 198
10-9. Kick 0 Register (KICK0R) Field Descriptions 199
10-10. Kick 1 Register (KICK1R) Field Descriptions 199
10-11. Host 0 Configuration Register (HOST0CFG) Field Descriptions 200
10-12. Interrupt Raw Status/Set Register (IRAWSTAT) Field Descriptions 201
10-13. Interrupt Enable Status/Clear Register (IENSTAT) Field Descriptions 202
10-14. Interrupt Enable Register (IENSET) Field Descriptions 203
10-15. Interrupt Enable Clear Register (IENCLR) Field Descriptions 203
10-16. End of Interrupt Register (EOI) Field Descriptions 204
10-17. Fault Address Register (FLTADDRR) Field Descriptions 204
10-18. Fault Status Register (FLTSTAT) Field Descriptions 205
10-19. Master Priority 0 Register (MSTPRI0) Field Descriptions 206
10-20. Master Priority 1 Register (MSTPRI1) Field Descriptions 207
10-21. Master Priority 2 Register (MSTPRI2) Field Descriptions 208
10-22. Pin Multiplexing Control 0 Register (PINMUX0) Field Descriptions 209
10-23. Pin Multiplexing Control 1 Register (PINMUX1) Field Descriptions 211
10-24. Pin Multiplexing Control 2 Register (PINMUX2) Field Descriptions 213
10-25. Pin Multiplexing Control 3 Register (PINMUX3) Field Descriptions 215
10-26. Pin Multiplexing Control 4 Register (PINMUX4) Field Descriptions 217
10-27. Pin Multiplexing Control 5 Register (PINMUX5) Field Descriptions 219
10-28. Pin Multiplexing Control 6 Register (PINMUX6) Field Descriptions 221
10-29. Pin Multiplexing Control 7 Register (PINMUX7) Field Descriptions 223
10-30. Pin Multiplexing Control 8 Register (PINMUX8) Field Descriptions 225
10-31. Pin Multiplexing Control 9 Register (PINMUX9) Field Descriptions 227
10-32. Pin Multiplexing Control 10 Register (PINMUX10) Field Descriptions 229
10-33. Pin Multiplexing Control 11 Register (PINMUX11) Field Descriptions 231
10-34. Pin Multiplexing Control 12 Register (PINMUX12) Field Descriptions 233
10-35. Pin Multiplexing Control 13 Register (PINMUX13) Field Descriptions 235
10-36. Pin Multiplexing Control 14 Register (PINMUX14) Field Descriptions 237
10-37. Pin Multiplexing Control 15 Register (PINMUX15) Field Descriptions 239
10-38. Pin Multiplexing Control 16 Register (PINMUX16) Field Descriptions 241
10-39. Pin Multiplexing Control 17 Register (PINMUX17) Field Descriptions 243
10-40. Pin Multiplexing Control 18 Register (PINMUX18) Field Descriptions 245
10-41. Pin Multiplexing Control 19 Register (PINMUX19) Field Descriptions 247
10-42. Suspend Source Register (SUSPSRC) Field Descriptions 249
10-43. Chip Signal Register (CHIPSIG) Field Descriptions 252
10-44. Chip Signal Clear Register (CHIPSIG_CLR) Field Descriptions 253
10-45. Chip Configuration 0 Register (CFGCHIP0) Field Descriptions 254
10-46. Chip Configuration 1 Register (CFGCHIP1) Field Descriptions 255
10-47. Chip Configuration 2 Register (CFGCHIP2) Field Descriptions 257

Copyright © 2013–2016, Texas Instruments Incorporated

10-48. Chip Configuration 3 Register (CFGCHIP3) Field Descriptions 259
10-49. Chip Configuration 4 Register (CFGCHIP4) Field Descriptions 260
10-50. VTP I/O Control Register (VTPIO_CTL) Field Descriptions 261
10-51. DDR Slew Register (DDR_SLEW) Field Descriptions 263
10-52. Deep Sleep Register (DEEPSLEEP) Field Descriptions 264
10-53. Pullup/Pulldown Enable Register (PUPD_ENA) Field Descriptions 265
10-54. Pullup/Pulldown Select Register (PUPD_SEL) Field Descriptions 265
10-55. Pullup/Pulldown Select Register (PUPD_SEL) Default Values 266
10-56. RXACTIVE Control Register (RXACTIVE) Field Descriptions 267
11-1. AINTC System Interrupt Assignments 270
11-2. ARM Interrupt Controller (AINTC) Registers 276
11-3. Revision Identification Register (REVID) Field Descriptions 277
11-4. Control Register (CR) Field Descriptions 278
11-5. Global Enable Register (GER) Field Descriptions 279
11-6. Global Nesting Level Register (GNLR) Field Descriptions 279
11-7. System Interrupt Status Indexed Set Register (SISR) Field Descriptions 280
11-8. System Interrupt Status Indexed Clear Register (SICR) Field Descriptions 280
11-9. System Interrupt Enable Indexed Set Register (EISR) Field Descriptions 281
11-10. System Interrupt Enable Indexed Clear Register (EICR) Field Descriptions 281
11-11. Host Interrupt Enable Indexed Set Register (HEISR) Field Descriptions 282
11-12. Host Interrupt Enable Indexed Clear Register (HIEICR) Field Descriptions 282
11-13. Vector Base Register (VBR) Field Descriptions 283
11-14. Vector Size Register (VSR) Field Descriptions 283
11-15. Vector Null Register (VNR) Field Descriptions 284
11-16. Global Prioritized Index Register (GPIR) Field Descriptions 284
11-17. Global Prioritized Vector Register (GPVR) Field Descriptions 285
11-18. System Interrupt Status Raw/Set Register 1 (SRSR1) Field Descriptions 285
11-19. System Interrupt Status Raw/Set Register 2 (SRSR2) Field Descriptions 286
11-20. System Interrupt Status Raw/Set Register 3 (SRSR3) Field Descriptions 286
11-21. System Interrupt Status Raw/Set Register 4 (SRSR4) Field Descriptions 287
11-22. System Interrupt Status Enabled/Clear Register 1 (SECR1) Field Descriptions 287
11-23. System Interrupt Status Enabled/Clear Register 2 (SECR2) Field Descriptions 288
11-24. System Interrupt Status Enabled/Clear Register 3 (SECR3) Field Descriptions 288
11-25. System Interrupt Status Enabled/Clear Register 4 (SECR4) Field Descriptions 289
11-26. System Interrupt Enable Set Register 1 (ESR1) Field Descriptions 289
11-27. System Interrupt Enable Set Register 2 (ESR2) Field Descriptions 290
11-28. System Interrupt Enable Set Register 3 (ESR3) Field Descriptions 290
11-29. System Interrupt Enable Set Register 4 (ESR4) Field Descriptions 291
11-30. System Interrupt Enable Clear Register 1 (ECR1) Field Descriptions 291
11-31. System Interrupt Enable Clear Register 2 (ECR2) Field Descriptions 292
11-32. System Interrupt Enable Clear Register 3 (ECR3) Field Descriptions 292
11-33. System Interrupt Enable Clear Register 4 (ECR4) Field Descriptions 293
11-34. Channel Map Registers (CMR*n*) Field Descriptions 293
11-35. Host Interrupt Prioritized Index Register 1 (HIPIR1) Field Descriptions 294
11-36. Host Interrupt Prioritized Index Register 2 (HIPIR2) Field Descriptions 294
11-37. Host Interrupt Nesting Level Register 1 (HINLR1) Field Descriptions 295
11-38. Host Interrupt Nesting Level Register 2 (HINLR2) Field Descriptions 295
11-39. Host Interrupt Enable Register (HIER) Field Descriptions 296
11-40. Host Interrupt Prioritized Vector Register 1 (HIPVR1) Field Descriptions 297

Copyright © 2013–2016, Texas Instruments Incorporated

11-41. Host Interrupt Prioritized Vector Register 2 (HIPVR2) Field Descriptions 297
13-1. Constants Table 304
13-2. Abbreviations for Instruction Descriptions 306
13-3. Load/Store Instructions 306
13-4. Arithmetic Instructions 306
13-5. Logical Instructions 307
13-6. Program Flow Control Instructions 308
13-7. Format 1a: (All Arithmetic and Logical Functions – Register Op2) 309
13-8. Format 1b: (All Arithmetic and Logical Functions – Immediate Op2) 311
13-9. Format 2 312
13-10. Format 2a: (JMP,JAL – Register Op2) 313
13-11. Format 2b: (JMP, JAL – Immediate Op2) 314
13-12. Format 2c: (LDI) 315
13-13. Format 2d: (LMBD - Leftmost Bit Detect - Register Op2) 316
13-14. Format 2e: (LMBD - Immediate Op2) 317
13-15. Format 2f: (SCAN - Register Op2) 318
13-16. Format 2g: (SCAN - Immediate Op2) 319
13-17. Format 2h: (HALT) 319
13-18. Format 2i: (SLP) 319
13-19. Format 4a: (Quick Arithmetic Test and Branch – Register Op2) 320
13-20. Format 4b: (Quick Arithmetic Test and Branch – Immediate Op2) 321
13-21. Format 5a: (Quick Bit Test and Branch – Register Op2) 322
13-22. Format 5b: (Quick Bit Test and Branch – Immediate Op2) 323
13-23. Format 6a: (LBBO/SBBO - Register Offset) 324
13-24. Format 6b: (LBBO/SBBO - Immediate Offset) 325
13-25. Format 6c: (LBCO/SBCO - Register Offset) 326
13-26. Format 6d: (LBCO/SBCO - Immediate Offset) 327
13-27. PRUSS System Events [0:31] Assignments 329
13-28. ARM Interrupt Controller Mapping 330
13-29. DSP Interrupt Controller Mapping 330
13-30. Local Instruction Space Memory Map 335
13-31. Local Data Space Memory Map 335
13-32. Subsystem Global Memory Map 335
13-33. PRU Control/Status Register Memory Map 336
13-34. CONTROL Register Field Descriptions 336
13-35. STATUS Register Field Descriptions 338
13-36. WAKEUP Register Field Descriptions 338
13-37. CYCLECNT Register Field Descriptions 339
13-38. STALLCNT Register Field Descriptions 339
13-39. CONTABBLKIDX0 Register Field Descriptions 340
13-40. CONTABPROPTR0 Register Field Descriptions 340
13-41. CONTABPROPTR1 Register Field Descriptions 340
13-42. INTGPR0 to INTGPR31 Register Field Descriptions 341
13-43. INTCTER0 to INTCTER31 Register Field Descriptions 341
13-44. Instruction RAM Memory Region 342
13-45. PRUSS Interrupt Controller (INTC) Registers 342
13-46. REVID Register 343
13-47. REVID Register Field Descriptions 343
13-48. CONTROL Register 343

Copyright © 2013–2016, Texas Instruments Incorporated

13-49. CONTROL Register Field Descriptions 343
13-50. GLBLEN Register 343
13-51. GLBLEN Register Field Descriptions 344
13-52. GLBLNSTLVL Register 344
13-53. GLBLNSTLVL Register Field Descriptions 344
13-54. STATIDXSET Register 344
13-55. STATIDXSET Register Field Descriptions 344
13-56. STATIDXCLR Register 345
13-57. STATIDXCLR Register Field Descriptions 345
13-58. ENIDXSET Register 345
13-59. ENIDXSET Register Field Descriptions 345
13-60. ENIDXCLR Register 345
13-61. ENIDXCLR Register Field Descriptions 345
13-62. HSTINTENIDXSET Register 346
13-63. HSTINTENIDXSET Register Field Descriptions 346
13-64. HSTINTENIDXCLR Register 346
13-65. HSTINTENIDXCLR Register Field Descriptions 346
13-66. GLBLPRIIDX Register 346
13-67. GLBLPRIIDX Register Field Descriptions 346
13-68. STATESETINT0 Register 347
13-69. STATESETINT0 Register Field Descriptions 347
13-70. STATESETINT1 Register 347
13-71. STATESETINT1 Register Field Descriptions 347
13-72. STATCLRINT0 Register 347
13-73. STATCLRINT0 Register Field Descriptions 347
13-74. STATCLRINT1 Register 348
13-75. STATCLRINT1 Register Field Descriptions 348
13-76. ENABLESET0 Register 348
13-77. ENABLESET0 Register Field Descriptions 348
13-78. ENABLESET1 Register 348
13-79. ENABLESET1 Register Field Descriptions 348
13-80. ENABLECLR0 Register 349
13-81. ENABLECLR0 Register Field Descriptions 349
13-82. ENABLECLR1 Register 349
13-83. ENABLECLR1 Register Field Descriptions 349
13-84. CHANMAP0 to CHANMAP15 Register 349
13-85. CHANMAP0 to CHANMAP15 Register Field Descriptions 349
13-86. HOSTMAP0 to HOSTMAP2 Register 350
13-87. HOSTMAP0 to HOSTMAP2 Register Field Descriptions 350
13-88. HOSTINTPRIIDX0 to HOSTINTPRIIDX9 Register 350
13-89. HOSTINTPRIIDX0 to HOSTINTPRIIDX9 Register Field Descriptions 350
13-90. POLARITY0 Register 350
13-91. POLARITY0 Register Field Descriptions 351
13-92. POLARITY1 Register 351
13-93. POLARITY1 Register Field Descriptions 351
13-94. TYPE0 Register 351
13-95. TYPE0 Register Field Descriptions 351
13-96. TYPE1 Register 351
13-97. TYPE1 Register Field Descriptions 352

Copyright © 2013–2016, Texas Instruments Incorporated

13-98. HOSTINTNSTLVL0 to HOSTINTNSTLVL9 Register 352
13-99. HOSTINTNSTLVL0 to HOSTINTNSTLVL9 Register Field Descriptions 352
13-100. HOSTINTEN Register 352
13-101. HOSTINTEN Register Field Descriptions 352
14-1. DDR2/mDDR SDRAM Commands 358
14-2. Truth Table for DDR2/mDDR SDRAM Commands 359
14-3. Addressable Memory Ranges 366
14-4. Configuration Register Fields for Address Mapping 367
14-5. Logical Address-to-DDR2/mDDR SDRAM Address Map for 16-bit SDRAM 368
14-6. Address Mapping Diagram for 16-Bit SDRAM (IBANKPOS = 1) 370
14-7. DDR2/mDDR Memory Controller FIFO Description 372
14-8. Refresh Urgency Levels 375
14-9. Configuration Bit Field for Partial Array Self-refresh 376
14-10. Reset Sources 377
14-11. DDR2 SDRAM Configuration by MRS Command 379
14-12. DDR2 SDRAM Configuration by EMRS(1) Command 379
14-13. Mobile DDR SDRAM Configuration by MRS Command 379
14-14. Mobile DDR SDRAM Configuration by EMRS(1) Command 380
14-15. SDCR Configuration 385
14-16. DDR2 Memory Refresh Specification 386
14-17. SDRCR Configuration 386
14-18. SDTIMR1 Configuration 387
14-19. SDTIMR2 Configuration 387
14-20. DRPYC1R Configuration 388
14-21. DDR2/mDDR Memory Controller Registers 389
14-22. Revision ID Register (REVID) Field Descriptions 389
14-23. SDRAM Status Register (SDRSTAT) Field Descriptions 390
14-24. SDRAM Configuration Register (SDCR) Field Descriptions 391
14-25. SDRAM Refresh Control Register (SDRCR) Field Descriptions 394
14-26. SDRAM Timing Register 1 (SDTIMR1) Field Descriptions 395
14-27. SDRAM Timing Register 2 (SDTIMR2) Field Descriptions 396
14-28. SDRAM Configuration Register 2 (SDCR2) Field Descriptions 397
14-29. Peripheral Bus Burst Priority Register (PBBPR) Field Descriptions 398
14-30. Performance Counter 1 Register (PC1) Field Descriptions 399
14-31. Performance Counter 2 Register (PC2) Field Descriptions 399
14-32. Performance Counter Configuration Register (PCC) Field Descriptions 400
14-33. Performance Counter Filter Configuration 401
14-34. Performance Counter Master Region Select Register (PCMRS) Field Descriptions 402
14-35. Performance Counter Time Register (PCT) Field Description 403
14-36. DDR PHY Reset Control Register (DRPYRCR) 403
14-37. Interrupt Raw Register (IRR) Field Descriptions 404
14-38. Interrupt Masked Register (IMR) Field Descriptions 404
14-39. Interrupt Mask Set Register (IMSR) Field Descriptions 405
14-40. Interrupt Mask Clear Register (IMCR) Field Descriptions 406
14-41. DDR PHY Control Register 1 (DRPYC1R) Field Descriptions 407
15-1. ECAP Initialization for CAP Mode Absolute Time, Rising Edge Trigger 421
15-2. ECAP Initialization for CAP Mode Absolute Time, Rising and Falling Edge Trigger 423
15-3. ECAP Initialization for CAP Mode Delta Time, Rising Edge Trigger 425
15-4. ECAP Initialization for CAP Mode Delta Time, Rising and Falling Edge Triggers 427

Copyright © 2013–2016, Texas Instruments Incorporated

15-5. ECAP Initialization for APWM Mode 429
15-6. ECAP1 Initialization for Multichannel PWM Generation with Synchronization 431
15-7. ECAP2 Initialization for Multichannel PWM Generation with Synchronization 431
15-8. ECAP3 Initialization for Multichannel PWM Generation with Synchronization 431
15-9. ECAP4 Initialization for Multichannel PWM Generation with Synchronization 431
15-10. ECAP1 Initialization for Multichannel PWM Generation with Phase Control 434
15-11. ECAP2 Initialization for Multichannel PWM Generation with Phase Control 434
15-12. ECAP3 Initialization for Multichannel PWM Generation with Phase Control 434
15-13. Control and Status Register Set 435
15-14. Time-Stamp Counter Register (TSCTR) Field Descriptions 435
15-15. Counter Phase Control Register (CTRPHS) Field Descriptions 436
15-16. Capture 1 Register (CAP1) Field Descriptions 436
15-17. Capture 2 Register (CAP2) Field Descriptions 437
15-18. Capture 3 Register (CAP3) Field Descriptions 437
15-19. Capture 4 Register (CAP4) Field Descriptions 438
15-20. ECAP Control Register 1 (ECCTL1) Field Descriptions 438
15-21. ECAP Control Register 2 (ECCTL2) Field Descriptions 440
15-22. ECAP Interrupt Enable Register (ECEINT) Field Descriptions 442
15-23. ECAP Interrupt Flag Register (ECFLG) Field Descriptions 443
15-24. ECAP Interrupt Clear Register (ECCLR) Field Descriptions 444
15-25. ECAP Interrupt Forcing Register (ECFRC) Field Descriptions 445
15-26. Revision ID Register (REVID) Field Descriptions 446
16-1. ePWM Module Control and Status Registers Grouped by Submodule 452
16-2. Submodule Configuration Parameters 453
16-3. Time-Base Submodule Registers 458
16-4. Key Time-Base Signals 459
16-5. Counter-Compare Submodule Registers 467
16-6. Counter-Compare Submodule Key Signals 467
16-7. Action-Qualifier Submodule Registers 471
16-8. Action-Qualifier Submodule Possible Input Events 472
16-9. Action-Qualifier Event Priority for Up-Down-Count Mode 474
16-10. Action-Qualifier Event Priority for Up-Count Mode 474
16-11. Action-Qualifier Event Priority for Down-Count Mode 474
16-12. Behavior if CMPA/CMPB is Greater than the Period 475
16-13. EPWMx Initialization for 478
16-14. EPWMx Run Time Changes for 478
16-15. EPWMx Initialization for 480
16-16. EPWMx Run Time Changes for 480
16-17. EPWMx Initialization for 482
16-18. EPWMx Run Time Changes for 482
16-19. EPWMx Initialization for 484
16-20. EPWMx Run Time Changes for 484
16-21. EPWMx Initialization for 486
16-22. EPWMx Run Time Changes for 486
16-23. EPWMx Initialization for 488
16-24. EPWMx Run Time Changes for 488
16-25. Dead-Band Generator Submodule Registers 489
16-26. Classical Dead-Band Operating Modes 491
16-27. PWM-Chopper Submodule Registers 493

Copyright © 2013–2016, Texas Instruments Incorporated

16-28. Trip-Zone Submodule Registers 498
16-29. Possible Actions On a Trip Event 499
16-30. Event-Trigger Submodule Registers 501
16-31. Resolution for PWM and HRPWM 506
16-32. HRPWM Submodule Registers 507
16-33. Relationship Between MEP Steps, PWM Frequency and Resolution 508
16-34. CMPA vs Duty (left), and [CMPA:CMPAHR] vs Duty (right) 509
16-35. EPWM1 Initialization for 516
16-36. EPWM2 Initialization for 516
16-37. EPWM3 Initialization for 516
16-38. EPWM1 Initialization for 519
16-39. EPWM2 Initialization for 519
16-40. EPWM1 Initialization for 522
16-41. EPWM2 Initialization for 522
16-42. EPWM1 Initialization for 525
16-43. EPWM2 Initialization for 525
16-44. EPWM3 Initialization for 526
16-45. EPWM1 Initialization for 531
16-46. EPWM2 Initialization for 531
16-47. EPWM3 Initialization for 532
16-48. EPWM1 Initialization for 535
16-49. EPWM2 Initialization for 535
16-50. Submodule Registers 536
16-51. Time-Base Submodule Registers 536
16-52. Time-Base Control Register (TBCTL) Field Descriptions 537
16-53. Time-Base Status Register (TBSTS) Field Descriptions 538
16-54. Time-Base Phase Register (TBPHS) Field Descriptions 539
16-55. Time-Base Counter Register (TBCNT) Field Descriptions 539
16-56. Time-Base Period Register (TBPRD) Field Descriptions 540
16-57. Counter-Compare Submodule Registers 540
16-58. Counter-Compare Control Register (CMPCTL) Field Descriptions 541
16-59. Counter-Compare A Register (CMPA) Field Descriptions 542
16-60. Counter-Compare B Register (CMPB) Field Descriptions 543
16-61. Action-Qualifier Submodule Registers 543
16-62. Action-Qualifier Output A Control Register (AQCTLA) Field Descriptions 544
16-63. Action-Qualifier Output B Control Register (AQCTLB) Field Descriptions 545
16-64. Action-Qualifier Software Force Register (AQSFRC) Field Descriptions 546
16-65. Action-Qualifier Continuous Software Force Register (AQCSFRC) Field Descriptions 547
16-66. Dead-Band Generator Submodule Registers 547
16-67. Dead-Band Generator Control Register (DBCTL) Field Descriptions 548
16-68. Dead-Band Generator Rising Edge Delay Register (DBRED) Field Descriptions 549
16-69. Dead-Band Generator Falling Edge Delay Register (DBFED) Field Descriptions 549
16-70. PWM-Chopper Control Register (PCCTL) Bit Descriptions 550
16-71. Trip-Zone Submodule Registers 551
16-72. Trip-Zone Submodule Select Register (TZSEL) Field Descriptions 551
16-73. Trip-Zone Control Register (TZCTL) Field Descriptions 552
16-74. Trip-Zone Enable Interrupt Register (TZEINT) Field Descriptions 552
16-75. Trip-Zone Flag Register (TZFLG) Field Descriptions 553
16-76. Trip-Zone Clear Register (TZCLR) Field Descriptions 554

Copyright © 2013–2016, Texas Instruments Incorporated

16-77. Trip-Zone Force Register (TZFRC) Field Descriptions 554
16-78. Event-Trigger Submodule Registers 555
16-79. Event-Trigger Selection Register (ETSEL) Field Descriptions 555
16-80. Event-Trigger Prescale Register (ETPS) Field Descriptions 556
16-81. Event-Trigger Flag Register (ETFLG) Field Descriptions 557
16-82. Event-Trigger Clear Register (ETCLR) Field Descriptions 557
16-83. Event-Trigger Force Register (ETFRC) Field Descriptions 558
16-84. High-Resolution PWM Submodule Registers 558
16-85. Time-Base Phase High-Resolution Register (TBPHSHR) Field Descriptions 559
16-86. Counter-Compare A High-Resolution Register (CMPAHR) Field Descriptions 559
16-87. HRPWM Configuration Register (HRCNFG) Field Descriptions 560
17-1. EDMA3 Channel Parameter Description 574
17-2. Dummy and Null Transfer Request 577
17-3. Parameter Updates in EDMA3CC (for Non-Null, Non-Dummy PaRAM Set) 578
17-4. Expected Number of Transfers for Non-Null Transfer 586
17-5. EDMA3 DMA Channel to PaRAM Mapping 588
17-6. Shadow Region Registers 590
17-7. Chain Event Triggers 592
17-8. EDMA3 Transfer Completion Interrupts 593
17-9. EDMA3 Error Interrupts 593
17-10. Transfer Complete Code (TCC) to EDMA3CC Interrupt Mapping 594
17-11. Number of Interrupts 594
17-12. EDMA3 Transfer Controller Configurations 601
17-13. Read/Write Command Optimization Rules 607
17-14. EDMA3 Channel Controller (EDMA3CC) Parameter RAM (PaRAM) Entries 626
17-15. Channel Options Parameters (OPT) Field Descriptions 627
17-16. Channel Source Address Parameter (SRC) Field Descriptions 629
17-17. A Count/B Count Parameter (A_B_CNT) Field Descriptions 629
17-18. Channel Destination Address Parameter (DST) Field Descriptions 630
17-19. Source B Index/Destination B Index Parameter (SRC_DST_BIDX) Field Descriptions 630
17-20. Link Address/B Count Reload Parameter (LINK_BCNTRLD) Field Descriptions 631
17-21. Source C Index/Destination C Index Parameter (SRC_DST_CIDX) Field Descriptions 632
17-22. C Count Parameter (CCNT) Field Descriptions 632
17-23. EDMA3 Channel Controller (EDMA3CC) Registers 633
17-24. Revision ID Register (REVID) Field Descriptions 636
17-25. EDMA3CC Configuration Register (CCCFG) Field Descriptions 637
17-26. QDMA Channel *n* Mapping Register (QCHMAP*n*) Field Descriptions 638
17-27. DMA Channel Queue Number Register *n* (DMAQNUM*n*) Field Descriptions 639
17-28. Bits in DMAQNUM*n* 639
17-29. QDMA Channel Queue Number Register (QDMAQNUM) Field Descriptions 640
17-30. Event Missed Register (EMR) Field Descriptions 641
17-31. Event Missed Clear Register (EMCR) Field Descriptions 642
17-32. QDMA Event Missed Register (QEMR) Field Descriptions 643
17-33. QDMA Event Missed Clear Register (QEMCR) Field Descriptions 644
17-34. EDMA3CC Error Register (CCERR) Field Descriptions 645
17-35. EDMA3CC Error Clear Register (CCERRCLR) Field Descriptions 646
17-36. Error Evaluate Register (EEVAL) Field Descriptions 647
17-37. DMA Region Access Enable Register for Region *m* (DRAE*m*) Field Descriptions 648
17-38. QDMA Region Access Enable for Region *m* (QRAE*m*) Field Descriptions 649

Copyright © 2013–2016, Texas Instruments Incorporated

17-39. Event Queue Entry Registers (Q*x*E*y*) Field Descriptions 650
17-40. Queue *n* Status Register (QSTAT*n*) Field Descriptions 651
17-41. Queue Watermark Threshold A Register (QWMTHRA) Field Descriptions 652
17-42. EDMA3CC Status Register (CCSTAT) Field Descriptions 653
17-43. Event Register (ER) Field Descriptions 655
17-44. Event Clear Register (ECR) Field Descriptions 656
17-45. Event Set Register (ESR) Field Descriptions 657
17-46. Chained Event Register (CER) Field Descriptions 658
17-47. Event Enable Register (EER) Field Descriptions 659
17-48. Event Enable Clear Register (EECR) Field Descriptions 660
17-49. Event Enable Set Register (EESR) Field Descriptions 660
17-50. Secondary Event Register (SER) Field Descriptions 661
17-51. Secondary Event Clear Register (SECR) Field Descriptions 661
17-52. Interrupt Enable Register (IER) Field Descriptions 662
17-53. Interrupt Enable Clear Register (IECR) Field Descriptions 663
17-54. Interrupt Enable Set Register (IESR) Field Descriptions 663
17-55. Interrupt Pending Register (IPR) Field Descriptions 664
17-56. Interrupt Clear Register (ICR) Field Descriptions 665
17-57. Interrupt Evaluate Register (IEVAL) Field Descriptions 666
17-58. QDMA Event Register (QER) Field Descriptions 667
17-59. QDMA Event Enable Register (QEER) Field Descriptions 668
17-60. QDMA Event Enable Clear Register (QEECR) Field Descriptions 669
17-61. QDMA Event Enable Set Register (QEESR) Field Descriptions 669
17-62. QDMA Secondary Event Register (QSER) Field Descriptions 670
17-63. QDMA Secondary Event Clear Register (QSECR) Field Descriptions 671
17-64. EDMA3 Transfer Controller (EDMA3TC) Registers 672
17-65. Revision ID Register (REVID) Field Descriptions 673
17-66. EDMA3TC Configuration Register (TCCFG) Field Descriptions 674
17-67. EDMA3TC Channel Status Register (TCSTAT) Field Descriptions 675
17-68. Error Status Register (ERRSTAT) Field Descriptions 676
17-69. Error Enable Register (ERREN) Field Descriptions 677
17-70. Error Clear Register (ERRCLR) Field Descriptions 678
17-71. Error Details Register (ERRDET) Field Descriptions 679
17-72. Error Interrupt Command Register (ERRCMD) Field Descriptions 680
17-73. Read Command Rate Register (RDRATE) Field Descriptions 681
17-74. Source Active Options Register (SAOPT) Field Descriptions 682
17-75. Source Active Source Address Register (SASRC) Field Descriptions 683
17-76. Source Active Count Register (SACNT) Field Descriptions 683
17-77. Source Active Destination Address Register (SADST) Field Descriptions 684
17-78. Source Active B-Index Register (SABIDX) Field Descriptions 684
17-79. Source Active Memory Protection Proxy Register (SAMPPRXY) Field Descriptions 685
17-80. Source Active Count Reload Register (SACNTRLD) Field Descriptions 686
17-81. Source Active Source Address B-Reference Register (SASRCBREF) Field Descriptions 686
17-82. Source Active Destination Address B-Reference Register (SADSTBREF) Field Descriptions 687
17-83. Destination FIFO Set Count Reload Register (DFCNTRLD) Field Descriptions 687
17-84. Destination FIFO Set Source Address B-Reference Register (DFSRCBREF) Field Descriptions 688
17-85. Destination FIFO Set Destination Address B-Reference Register (DFDSTBREF) Field Descriptions 688
17-86. Destination FIFO Options Register *n* (DFOPT*n*) Field Descriptions 689
17-87. Destination FIFO Source Address Register *n* (DFSRC*n*) Field Descriptions 690

Copyright © 2013–2016, Texas Instruments Incorporated

17-88. Destination FIFO Count Register *n* (DFCNT*n*) Field Descriptions 690
17-89. Destination FIFO Destination Address Register *n* (DFDST*n*) Field Descriptions 691
17-90. Destination FIFO B-Index Register *n* (DFBIDX*n*) Field Descriptions 691
17-91. Destination FIFO Memory Protection Proxy Register *n* (DFMPPRXY*n*) Field Descriptions 692
17-92. Debug List 693
18-1. EMIFA Pins Used to Access Both SDRAM and Asynchronous Memories 698
18-2. EMIFA Pins Specific to SDRAM 699
18-3. EMIFA Pins Specific to Asynchronous Memory 699
18-4. EMIFA SDRAM Commands 700
18-5. Truth Table for SDRAM Commands 700
18-6. 16-bit EMIFA Address Pin Connections 702
18-7. Description of the SDRAM Configuration Register (SDCR) 703
18-8. Description of the SDRAM Refresh Control Register (SDRCR) 703
18-9. Description of the SDRAM Timing Register (SDTIMR) 704
18-10. Description of the SDRAM Self Refresh Exit Timing Register (SDSRETR) 704
18-11. SDRAM LOAD MODE REGISTER Command 705
18-12. Refresh Urgency Levels 706
18-13. Mapping from Logical Address to EMIFA Pins for 16-bit SDRAM 711
18-14. Normal Mode vs. Select Strobe Mode 712
18-15. Description of the Asynchronous *m* Configuration Register (CE*n*CFG) 714
18-16. Description of the Asynchronous Wait Cycle Configuration Register (AWCC) 715
18-17. Description of the EMIFA Interrupt Mask Set Register (INTMSKSET) 717
18-18. Description of the EMIFA Interrupt Mast Clear Register (INTMSKCLR) 717
18-19. Asynchronous Read Operation in Normal Mode 717
18-20. Asynchronous Write Operation in Normal Mode 719
18-21. Asynchronous Read Operation in Select Strobe Mode 721
18-22. Asynchronous Write Operation in Select Strobe Mode 723
18-23. Description of the NAND Flash Control Register (NANDFCR) 725
18-24. Reset Sources 731
18-25. Interrupt Monitor and Control Bit Fields 733
18-26. SR Field Value For the EMIFA to K4S641632H-TC(L)70 Interface 738
18-27. SDTIMR Field Calculations for the EMIFA to K4S641632H-TC(L)70 Interface 740
18-28. RR Calculation for the EMIFA to K4S641632H-TC(L)70 Interface 741
18-29. RR Calculation for the EMIFA to K4S641632H-TC(L)70 Interface 741
18-30. SDCR Field Values For the EMIFA to K4S641632H-TC(L)70 Interface 742
18-31. EMIFA Input Timing Requirements 743
18-32. ASRAM Output Timing Characteristics 743
18-33. ASRAM Input Timing Requirement for a Read 743
18-34. ASRAM Input Timing Requirements for a Write 744
18-35. ASRAM Timing Requirements With PCB Delays 746
18-36. EMIFA Timing Requirements for TC5516100FT-12 Example 749
18-37. ASRAM Timing Requirements for TC5516100FT-12 Example 749
18-38. Measured PCB Delays for TC5516100FT-12 Example 749
18-39. Configuring CE3CFG for TC5516100FT-12 Example 751
18-40. Recommended Margins 751
18-41. EMIFA Read Timing Requirements 752
18-42. NAND Flash Read Timing Requirements 752
18-43. NAND Flash Write Timing Requirements 754
18-44. EMIFA Timing Requirements for HY27UA081G1M Example 757

Copyright © 2013–2016, Texas Instruments Incorporated

18-45. NAND Flash Timing Requirements for HY27UA081G1M Example 757
18-46. Configuring CE2CFG for HY27UA081G1M Example 759
18-47. Configuring NANDFCR for HY27UA081G1M Example 759
18-48. External Memory Interface (EMIFA) Registers 760
18-49. Module ID Register (MIDR) Field Descriptions 761
18-50. Asynchronous Wait Cycle Configuration Register (AWCCR) Field Descriptions 762
18-51. SDRAM Configuration Register (SDCR) Field Descriptions 763
18-52. SDRAM Refresh Control Register (SDRCR) Field Descriptions 765
18-53. Asynchronous *n* Configuration Register (CE*n*CFG) Field Descriptions 766
18-54. SDRAM Timing Register (SDTIMR) Field Descriptions 768
18-55. SDRAM Self Refresh Exit Timing Register (SDSRETR) Field Descriptions 769
18-56. EMIFA Interrupt Raw Register (INTRAW) Field Descriptions 770
18-57. EMIFA Interrupt Mask Register (INTMSK) Field Descriptions 771
18-58. EMIFA Interrupt Mask Set Register (INTMSKSET) Field Descriptions 772
18-59. EMIFA Interrupt Mask Clear Register (INTMSKCLR) Field Descriptions 773
18-60. NAND Flash Control Register (NANDFCR) Field Descriptions 774
18-61. NAND Flash Status Register (NANDFSR) Field Descriptions 776
18-62. NAND Flash *n* ECC Register (NANDF*n*ECC) Field Descriptions 777
18-63. NAND Flash 4-Bit ECC LOAD Register (NAND4BITECCLOAD) Field Descriptions 778
18-64. NAND Flash 4-Bit ECC Register 1 (NAND4BITECC1) Field Descriptions 779
18-65. NAND Flash 4-Bit ECC Register 2 (NAND4BITECC2) Field Descriptions 779
18-66. NAND Flash 4-Bit ECC Register 3 (NAND4BITECC3) Field Descriptions 780
18-67. NAND Flash 4-Bit ECC Register 4 (NAND4BITECC4) Field Descriptions 780
18-68. NAND Flash 4-Bit ECC Error Address Register 1 (NANDERRADD1) Field Descriptions 781
18-69. NAND Flash 4-Bit ECC Error Address Register 2 (NANDERRADD2) Field Descriptions 781
18-70. NAND Flash 4-Bit ECC Error Value Register 1 (NANDERRVAL1) Field Descriptions 782
18-71. NAND Flash 4-Bit ECC Error Value Register 2 (NANDERRVAL2) Field Descriptions 782
19-1. GPIO Register Bits and Banks Associated With GPIO Signals 786
19-2. GPIO Registers 793
19-3. Revision ID Register (REVID) Field Descriptions 794
19-4. GPIO Interrupt Per-Bank Enable Register (BINTEN) Field Descriptions 795
19-5. GPIO Direction Register (DIR*n*) Field Descriptions 797
19-6. GPIO Output Data Register (OUT_DATA*n*) Field Descriptions 799
19-7. GPIO Set Data Register (SET_DATA*n*) Field Descriptions 801
19-8. GPIO Clear Data Register (CLR_DATA*n*) Field Descriptions 803
19-9. GPIO Input Data Register (IN_DATA*n*) Field Descriptions 805
19-10. GPIO Set Rising Edge Trigger Interrupt Register (SET_RIS_TRIG*n*) Field Descriptions 807
19-11. GPIO Clear Rising Edge Interrupt Register (CLR_RIS_TRIG*n*) Field Descriptions 809
19-12. GPIO Set Falling Edge Trigger Interrupt Register (SET_FAL_TRIG*n*) Field Descriptions 811
19-13. GPIO Clear Falling Edge Interrupt Register (CLR_FAL_TRIG*n*) Field Descriptions 813
19-14. GPIO Interrupt Status Register (INTSTAT*n*) Field Descriptions 815
20-1. HPI Pins 820
20-2. Value on Optional Pins when Configured as General-Purpose I/O 821
20-3. Options for Connecting Host and HPI Data Strobe Pins 825
20-4. Access Types Selectable With the UHPI_HCNTL Signals 826
20-5. Cycle Types Selectable With the UHPI_HCNTL and UHPI_HR/$\overline{W}$ Signals 826
20-6. HPI Registers 840
20-7. Revision Identification Register (REVID) Field Descriptions 841
20-8. Power and Emulation Management Register (PWREMU_MGMT) Field Descriptions 841

Copyright © 2013–2016, Texas Instruments Incorporated

20-9. GPIO Enable Register (GPIO_EN) Field Descriptions 842
20-10. GPIO Direction 1 Register (GPIO_DIR1) Field Descriptions 843
20-11. GPIO Data 1 Register (GPIO_DAT1) Field Descriptions 843
20-12. GPIO Direction 2 Register (GPIO_DIR2) Field Descriptions 844
20-13. GPIO Data 2 Register (GPIO_DAT2) Field Descriptions 845
20-14. Host Port Interface Control Register (HPIC) Field Descriptions 847
20-15. Host Port Interface Write Address Register (HPIAW) Field Descriptions 848
20-16. Host Port Interface Read Address Register (HPIAR) Field Descriptions 848
21-1. Operating Modes of the I2C Peripheral 858
21-2. Ways to Generate a NACK Bit 859
21-3. Descriptions of the I2C Interrupt Events 863
21-4. Inter-Integrated Circuit (I2C) Registers 864
21-5. I2C Own Address Register (ICOAR) Field Descriptions 865
21-6. I2C Interrupt Mask Register (ICIMR) Field Descriptions 866
21-7. I2C Interrupt Status Register (ICSTR) Field Descriptions 867
21-8. I2C Clock Low-Time Divider Register (ICCLKL) Field Descriptions 870
21-9. I2C Clock High-Time Divider Register (ICCLKH) Field Descriptions 870
21-10. I2C Data Count Register (ICCNT) Field Descriptions 871
21-11. I2C Data Receive Register (ICDRR) Field Descriptions 872
21-12. I2C Slave Address Register (ICSAR) Field Descriptions 873
21-13. I2C Data Transmit Register (ICDXR) Field Descriptions 874
21-14. I2C Mode Register (ICMDR) Field Descriptions 875
21-15. Master-Transmitter/Receiver Bus Activity Defined by RM, STT, and STP Bits 877
21-16. How the MST and FDF Bits Affect the Role of TRX Bit 877
21-17. I2C Interrupt Vector Register (ICIVR) Field Descriptions 879
21-18. I2C Extended Mode Register (ICEMDR) Field Descriptions 880
21-19. I2C Prescaler Register (ICPSC) Field Descriptions 881
21-20. I2C Revision Identification Register 1 (REVID1) Field Descriptions 882
21-21. I2C Revision Identification Register 2 (REVID2) Field Descriptions 882
21-22. I2C DMA Control Register (ICDMAC) Field Descriptions 883
21-23. I2C Pin Function Register (ICPFUNC) Field Descriptions 884
21-24. I2C Pin Direction Register (ICPDIR) Field Descriptions 885
21-25. I2C Pin Data In Register (ICPDIN) Field Descriptions 886
21-26. I2C Pin Data Out Register (ICPDOUT) Field Descriptions 887
21-27. I2C Pin Data Set Register (ICPDSET) Field Descriptions 888
21-28. I2C Pin Data Clear Register (ICPDCLR) Field Descriptions 889
22-1. LCD External I/O Signals 894
22-2. Register Configuration for DMA Engine Programming 895
22-3. LIDD I/O Name Map 897
22-4. Operation Modes Supported by Raster Controller 898
22-5. Bits-Per-Pixel Encoding for Palette Entry 0 Buffer 900
22-6. Frame Buffer Size According to BPP 901
22-7. Color/Grayscale Intensities and Modulation Rates 905
22-8. Number of Colors/Shades of Gray Available on Screen 905
22-9. LCD Controller (LCDC) Registers 908
22-10. LCD Revision Identification Register (REVID) Field Descriptions 908
22-11. LCD Control Register (LCD_CTRL) Field Descriptions 909
22-12. Pixel Clock Frequency Programming Limitations 910
22-13. LCD Status Register (LCD_STAT) Field Descriptions 911

Copyright © 2013–2016, Texas Instruments Incorporated

22-14. LCD LIDD Control Register (LIDD_CTRL) Field Descriptions 914
22-15. LCD LIDD CS*n* Configuration Register (LIDD_CS*n*_CONF) Field Descriptions 916
22-16. LCD LIDD CS*n* Address Read/Write Register (LIDD_CS*n*_ADDR) Field Descriptions 917
22-17. LCD LIDD CS*n* Data Read/Write Register (LIDD_CS*n*_DATA) Field Descriptions 918
22-18. LCD Raster Control Register (RASTER_CTRL) Field Descriptions 919
22-19. LCD Controller Data Pin Utilization for Mono/Color Passive/Active Panels 921
22-20. LCD Raster Timing Register 0 (RASTER_TIMING_0) Field Descriptions 926
22-21. LCD Raster Timing Register 1 (RASTER_TIMING_1) Field Descriptions 928
22-22. LCD Raster Timing Register 2 (RASTER_TIMING_2) Field Descriptions 932
22-23. LCD Raster Subpanel Display Register (RASTER_SUBPANEL) Field Descriptions 936
22-24. LCD DMA Control Register (LCDDMA_CTRL) Field Descriptions 938
22-25. LCD DMA Frame Buffer *n* Base Address Register (LCDDMA_FB*n*_BASE) Field Descriptions 939
22-26. LCD DMA Frame Buffer *n* Ceiling Address Register (LCDDMA_FB*n*_CEILING) Field Descriptions 939
23-1. Biphase-Mark Encoder 948
23-2. Preamble Codes 949
23-3. Channel Status and User Data for Each DIT Block 975
23-4. Transmit Bitstream Data Alignment 983
23-5. Receive Bitstream Data Alignment 985
23-6. EDMA Events - McASP 995
23-7. McASP Registers Accessed by CPU/EDMA Through Peripheral Configuration Port 996
23-8. McASP Registers Accessed by CPU/EDMA Through DMA Port 999
23-9. McASP AFIFO Registers Accessed Through Peripheral Configuration Port 999
23-10. Revision Identification Register (REV) Field Descriptions 1000
23-11. Pin Function Register (PFUNC) Field Descriptions 1002
23-12. Pin Direction Register (PDIR) Field Descriptions 1004
23-13. Pin Data Output Register (PDOUT) Field Descriptions 1006
23-14. Pin Data Input Register (PDIN) Field Descriptions 1008
23-15. Pin Data Set Register (PDSET) Field Descriptions 1010
23-16. Pin Data Clear Register (PDCLR) Field Descriptions 1012
23-17. Global Control Register (GBLCTL) Field Descriptions 1013
23-18. Audio Mute Control Register (AMUTE) Field Descriptions 1015
23-19. Digital Loopback Control Register (DLBCTL) Field Descriptions 1017
23-20. Digital Mode Control Register (DITCTL) Field Descriptions 1018
23-21. Receiver Global Control Register (RGBLCTL) Field Descriptions 1019
23-22. Receive Format Unit Bit Mask Register (RMASK) Field Descriptions 1020
23-23. Receive Bit Stream Format Register (RFMT) Field Descriptions 1021
23-24. Receive Frame Sync Control Register (AFSRCTL) Field Descriptions 1023
23-25. Receive Clock Control Register (ACLKRCTL) Field Descriptions 1024
23-26. Receive High-Frequency Clock Control Register (AHCLKRCTL) Field Descriptions 1025
23-27. Receive TDM Time Slot Register (RTDM) Field Descriptions 1026
23-28. Receiver Interrupt Control Register (RINTCTL) Field Descriptions 1027
23-29. Receiver Status Register (RSTAT) Field Descriptions 1028
23-30. Current Receive TDM Time Slot Registers (RSLOT) Field Descriptions 1029
23-31. Receive Clock Check Control Register (RCLKCHK) Field Descriptions 1030
23-32. Receiver DMA Event Control Register (REVTCTL) Field Descriptions 1031
23-33. Transmitter Global Control Register (XGBLCTL) Field Descriptions 1032
23-34. Transmit Format Unit Bit Mask Register (XMASK) Field Descriptions 1033
23-35. Transmit Bit Stream Format Register (XFMT) Field Descriptions 1034
23-36. Transmit Frame Sync Control Register (AFSXCTL) Field Descriptions 1036

Copyright © 2013–2016, Texas Instruments Incorporated

23-37. Transmit Clock Control Register (ACLKXCTL) Field Descriptions 1037
23-38. Transmit High-Frequency Clock Control Register (AHCLKXCTL) Field Descriptions 1038
23-39. Transmit TDM Time Slot Register (XTDM) Field Descriptions 1039
23-40. Transmitter Interrupt Control Register (XINTCTL) Field Descriptions 1040
23-41. Transmitter Status Register (XSTAT) Field Descriptions 1041
23-42. Current Transmit TDM Time Slot Register (XSLOT) Field Descriptions 1042
23-43. Transmit Clock Check Control Register (XCLKCHK) Field Descriptions 1043
23-44. Transmitter DMA Event Control Register (XEVTCTL) Field Descriptions 1044
23-45. Serializer Control Registers (SRCTL*n*) Field Descriptions 1045
23-46. AFIFO Revision Identification Register (AFIFOREV) Field Descriptions 1049
23-47. Write FIFO Control Register (WFIFOCTL) Field Descriptions 1050
23-48. Write FIFO Status Register (WFIFOSTS) Field Descriptions 1051
23-49. Read FIFO Control Register (RFIFOCTL) Field Descriptions 1052
23-50. Read FIFO Status Register (RFIFOSTS) Field Descriptions 1053
24-1. McBSP Interface Signals 1057
24-2. Choosing an Input Clock for the Sample Rate Generator With the SCLKME and CLKSM Bits 1061
24-3. Receive Clock Selection 1064
24-4. Transmit Clock Selection 1065
24-5. Receive Frame Synchronization Selection 1066
24-6. Transmit Frame Synchronization Selection 1067
24-7. RCR/XCR Fields Controlling Elements per Frame and Bits per Element 1068
24-8. Receive/Transmit Frame Length Configuration 1068
24-9. Receive/Transmit Element Length Configuration 1069
24-10. Effect of RJUST Bit Values With 12-Bit Example Data ABCh 1071
24-11. Effect of RJUST Bit Values With 20-Bit Example Data ABCDEh 1071
24-12. Justification of Expanded Data in DRR 1087
24-13. Receive Channel Assignment and Control When Two Receive Partitions are Used 1090
24-14. Transmit Channel Assignment and Control When Two Transmit Partitions are Used 1090
24-15. Receive Channel Assignment and Control When Eight Receive Partitions are Used 1092
24-16. Transmit Channel Assignment and Control When Eight Transmit Partitions are Used 1092
24-17. Selecting a Transmit Multichannel Selection Mode With the XMCM Bits 1093
24-18. Reset State of McBSP Pins 1096
24-19. Receiver Clock and Frame Configurations 1097
24-20. Transmitter Clock and Frame Configurations 1097
24-21. McBSP Emulation Modes Selectable With the FREE and SOFT Bits of SPCR 1103
24-22. McBSP Registers 1104
24-23. Data Receive Register (DRR) Field Descriptions 1105
24-24. Data Transmit Register (DXR) Field Descriptions 1105
24-25. Serial Port Control Register (SPCR) Field Descriptions 1106
24-26. Receive Control Register (RCR) Field Descriptions 1108
24-27. Transmit Control Register (XCR) Field Descriptions 1110
24-28. Sample Rate Generator Register (SRGR) Field Descriptions 1112
24-29. Multichannel Control Register (MCR) Field Descriptions 1113
24-30. Enhanced Receive Channel Enable Register *n* (RCERE*n*) Field Descriptions 1117
24-31. Use of the Receive Channel Enable Registers 1118
24-32. Enhanced Transmit Channel Enable Register *n* (XCERE*n*) Field Descriptions 1119
24-33. Use of the Transmit Channel Enable Registers 1120
24-34. Pin Control Register (PCR) Field Descriptions 1121
24-35. BFIFO Revision Identification Register (BFIFOREV) Field Descriptions 1123

Copyright © 2013–2016, Texas Instruments Incorporated

24-36. Write FIFO Control Register (WFIFOCTL) Field Descriptions 1124
24-37. Write FIFO Status Register (WFIFOSTS) Field Descriptions 1125
24-38. Read FIFO Control Register (RFIFOCTL) Field Descriptions 1126
24-39. Read FIFO Status Register (RFIFOSTS) Field Descriptions 1127
25-1. MMC/SD Controller Pins Used in Each Mode 1132
25-2. MMC/SD Mode Write Sequence 1133
25-3. MMC/SD Mode Read Sequence 1134
25-4. Description of MMC/SD Interrupt Requests 1144
25-5. Multimedia Card/Secure Digital (MMC/SD) Card Controller Registers 1158
25-6. MMC Control Register (MMCCTL) Field Descriptions 1159
25-7. MMC Memory Clock Control Register (MMCCLK) Field Descriptions 1160
25-8. MMC Status Register 0 (MMCST0) Field Descriptions 1161
25-9. MMC Status Register 1 (MMCST1) Field Descriptions 1163
25-10. MMC Interrupt Mask Register (MMCIM) Field Descriptions 1164
25-11. MMC Response Time-Out Register (MMCTOR) Field Descriptions 1166
25-12. MMC Data Read Time-Out Register (MMCTOD) Field Descriptions 1167
25-13. MMC Block Length Register (MMCBLEN) Field Descriptions 1168
25-14. MMC Number of Blocks Register (MMCNBLK) Field Descriptions 1169
25-15. MMC Number of Blocks Counter Register (MMCNBLC) Field Descriptions 1169
25-16. MMC Data Receive Register (MMCDRR) Field Descriptions 1170
25-17. MMC Data Transmit Register (MMCDXR) Field Descriptions 1170
25-18. MMC Command Register (MMCCMD) Field Descriptions 1171
25-19. Command Format 1172
25-20. MMC Argument Register (MMCARGHL) Field Descriptions 1173
25-21. R1, R3, R4, R5, or R6 Response (48 Bits) 1175
25-22. R2 Response (136 Bits) 1175
25-23. MMC Data Response Register (MMCDRSP) Field Descriptions 1176
25-24. MMC Command Index Register (MMCCIDX) Field Descriptions 1176
25-25. SDIO Control Register (SDIOCTL) Field Descriptions 1177
25-26. SDIO Status Register 0 (SDIOST0) Field Descriptions 1178
25-27. SDIO Interrupt Enable Register (SDIOIEN) Field Descriptions 1179
25-28. SDIO Interrupt Status Register (SDIOIST) Field Descriptions 1179
25-29. MMC FIFO Control Register (MMCFIFOCTL) Field Descriptions 1180
26-1. Real-Time Clock Signals 1183
26-2. Real-Time Clock (RTC) Registers 1189
26-3. Second Register (SECOND) Field Descriptions 1190
26-4. Minute Register (MINUTE) Field Descriptions 1190
26-5. Hour Register (HOUR) Field Descriptions 1191
26-6. Day Register (DAY) Field Descriptions 1192
26-7. Month Register (MONTH) Field Descriptions 1192
26-8. Year Register (YEAR) Field Descriptions 1193
26-9. Day of the Week Register (DOTW) Field Descriptions 1193
26-10. Alarm Second Register (ALARMSECOND) Field Descriptions 1194
26-11. Alarm Minute Register (ALARMMINUTE) Field Descriptions 1194
26-12. Alarm Hour Register (ALARMHOUR) Field Descriptions 1195
26-13. Alarm Day Register (ALARMDAY) Field Descriptions 1196
26-14. Alarm Month Register (ALARMMONTH) Field Descriptions 1197
26-15. Alarm Years Register (ALARMYEARS) Field Descriptions 1197
26-16. Control Register (CTRL) Field Descriptions 1198

Copyright © 2013–2016, Texas Instruments Incorporated

26-17. Status Register (STATUS) Field Descriptions 1199
26-18. Interrupt Register (INTERRUPT) Field Descriptions 1200
26-19. Compensations Register (COMPLSB) Field Descriptions 1201
26-20. Compensations Register (COMPMSB) Field Descriptions 1202
26-21. Oscillator Register (OSC) Field Descriptions 1203
26-22. Scratch Registers (SCRATCH*n*) Field Descriptions 1204
26-23. Kick Registers (KICK*n*R) Field Descriptions 1204
27-1. SPI Pins 1208
27-2. SPI Registers 1209
27-3. SPI Register Settings Defining Master Modes 1210
27-4. Allowed SPI Register Settings in Master Modes 1210
27-5. SPI Register Settings Defining Slave Modes 1212
27-6. Allowed SPI Register Settings in Slave Modes 1212
27-7. Clocking Modes 1221
27-8. SPI Registers 1234
27-9. SPI Global Control Register 0 (SPIGCR0) Field Descriptions 1234
27-10. SPI Global Control Register 1 (SPIGCR1) Field Descriptions 1235
27-11. SPI Interrupt Register (SPIINT0) Field Descriptions 1237
27-12. SPI Interrupt Level Register (SPILVL) Field Descriptions 1239
27-13. SPI Flag Register (SPIFLG) Field Descriptions 1240
27-14. SPI Pin Control Register 0 (SPIPC0) Field Descriptions 1242
27-15. SPI Pin Control Register 1 (SPIPC1) Field Descriptions 1243
27-16. SPI Pin Control Register 2 (SPIPC2) Field Descriptions 1244
27-17. SPI Pin Control Register 3 (SPIPC3) Field Descriptions 1245
27-18. SPI Pin Control Register 4 (SPIPC4) Field Descriptions 1246
27-19. SPI Pin Control Register 5 (SPIPC5) Field Descriptions 1247
27-20. SPI Data Register 0 (SPIDAT0) Field Descriptions 1248
27-21. SPI Data Register 1 (SPIDAT1) Field Descriptions 1249
27-22. SPI Buffer Register (SPIBUF) Field Descriptions 1250
27-23. SPI Emulation Register (SPIEMU) Field Descriptions 1252
27-24. SPI Delay Register (SPIDELAY) Field Descriptions 1253
27-25. SPI Default Chip Select Register (SPIDEF) Field Descriptions 1256
27-26. SPI Data Format Register (SPIFMT*n*) Field Descriptions 1257
27-27. SPI Interrupt Vector Register 1 (INTVEC1) Field Descriptions 1259
28-1. Timer Clock Source Selection 1263
28-2. 64-Bit Timer Configurations 1265
28-3. 32-Bit Timer Chained Mode Configurations 1268
28-4. 32-Bit Timer Unchained Mode Configurations 1271
28-5. Counter and Period Registers Used in GP Timer Modes 1273
28-6. TSTAT Parameters in Pulse and Clock Modes 1277
28-7. Timer Emulation Modes Selection 1279
28-8. Timer Registers 1279
28-9. Revision ID Register (REVID) Field Descriptions 1281
28-10. Emulation Management Register (EMUMGT) Field Descriptions 1281
28-11. GPIO Interrupt Control and Enable Register (GPINTGPEN) Field Descriptions 1282
28-12. GPIO Data and Direction Register (GPDATGPDIR) Field Descriptions 1283
28-13. Timer Counter Register 12 (TIM12) Field Descriptions 1284
28-14. Timer Counter Register 34 (TIM34) Field Descriptions 1284
28-15. Timer Period Register (PRD12) Field Descriptions 1285

Copyright © 2013–2016, Texas Instruments Incorporated

28-16. Timer Period Register (PRD34) Field Descriptions 1285
28-17. Timer Control Register (TCR) Field Descriptions 1286
28-18. Timer Global Control Register (TGCR) Field Descriptions 1288
28-19. Watchdog Timer Control Register (WDTCR) Field Descriptions 1289
28-20. Timer Reload Register 12 (REL12) Field Descriptions 1290
28-21. Timer Reload Register 34 (REL34) Field Descriptions 1290
28-22. Timer Capture Register 12 (CAP12) Field Descriptions 1291
28-23. Timer Capture Register 34 (CAP34) Field Descriptions 1291
28-24. Timer Interrupt Control and Status Register (INTCTLSTAT) Field Descriptions 1292
28-25. Timer Compare Register (CMP*n*) Field Descriptions 1293
29-1. Baud Rate Examples for 150-MHZ UART Input Clock and 16× Over-sampling Mode 1298
29-2. Baud Rate Examples for 150-MHZ UART Input Clock and 13× Over-sampling Mode 1298
29-3. UART Signal Descriptions 1299
29-4. Character Time for Word Lengths 1302
29-5. UART Interrupt Requests Descriptions 1306
29-6. UART Registers 1308
29-7. Receiver Buffer Register (RBR) Field Descriptions 1309
29-8. Transmitter Holding Register (THR) Field Descriptions 1310
29-9. Interrupt Enable Register (IER) Field Descriptions 1311
29-10. Interrupt Identification Register (IIR) Field Descriptions 1312
29-11. Interrupt Identification and Interrupt Clearing Information 1313
29-12. FIFO Control Register (FCR) Field Descriptions 1314
29-13. Line Control Register (LCR) Field Descriptions 1315
29-14. Relationship Between ST, EPS, and PEN Bits in LCR 1316
29-15. Number of STOP Bits Generated 1316
29-16. Modem Control Register (MCR) Field Descriptions 1317
29-17. Line Status Register (LSR) Field Descriptions 1318
29-18. Modem Status Register (MSR) Field Descriptions 1321
29-19. Scratch Pad Register (MSR) Field Descriptions 1322
29-20. Divisor LSB Latch (DLL) Field Descriptions 1323
29-21. Divisor MSB Latch (DLH) Field Descriptions 1323
29-22. Revision Identification Register 1 (REVID1) Field Descriptions 1324
29-23. Revision Identification Register 2 (REVID2) Field Descriptions 1324
29-24. Power and Emulation Management Register (PWREMU_MGMT) Field Descriptions 1325
29-25. Mode Definition Register (MDR) Field Descriptions 1326
30-1. I/O Clock Speeds for Channel in Transmit Mode Given 150 MHz Transmit Clock 1332
30-2. uPP Signal Descriptions 1332
30-3. DATA and XDATA Pin Assignments to Channels A and B According to Operating Mode 1333
30-4. Interface and DMA Channel Mapping for Various Operating Modes 1335
30-5. Required Signals for Various Modes 1336
30-6. Data Packing Examples for 12-Bit Data Words 1340
30-7. Basic Operating Mode Selection 1342
30-8. Sample uPP Parameters for Duplex Mode 0 1343
30-9. uPP Parameters Useful for System Tuning 1344
30-10. uPP Registers 1348
30-11. uPP Peripheral Identification Register (UPPID) Field Descriptions 1348
30-12. uPP Peripheral Control Register (UPPCR) Field Descriptions 1349
30-13. uPP Digital Loopback Register (UPDLB) Field Descriptions 1350
30-14. uPP Channel Control Register (UPCTL) Field Descriptions 1351

Copyright © 2013–2016, Texas Instruments Incorporated

30-15. uPP Interface Configuration Register (UPICR) Field Descriptions 1353
30-16. uPP Interface Idle Value Register (UPIVR) Field Descriptions 1355
30-17. uPP Threshold Configuration Register (UPTCR) Field Descriptions 1356
30-18. uPP Interrupt Raw Status Register (UPISR) Field Descriptions 1357
30-19. uPP Interrupt Enabled Status Register (UPIER) Field Descriptions 1359
30-20. uPP Interrupt Enable Set Register (UPIES) Field Descriptions 1361
30-21. uPP Interrupt Enable Clear Register (UPIEC) Field Descriptions 1363
30-22. uPP End of Interrupt Register (UPEOI) Field Descriptions 1365
30-23. uPP DMA Channel I Descriptor 0 Register (UPID0) Field Descriptions 1365
30-24. uPP DMA Channel I Descriptor 1 Register (UPID1) Field Descriptions 1366
30-25. uPP DMA Channel I Descriptor 2 Register (UPID2) Field Descriptions 1366
30-26. uPP DMA Channel I Status 0 Register (UPIS0) Field Descriptions 1367
30-27. uPP DMA Channel I Status 1 Register (UPIS1) Field Descriptions 1367
30-28. uPP DMA Channel I Status 2 Register (UPIS2) Field Descriptions 1368
30-29. uPP DMA Channel Q Descriptor 0 Register (UPQD0) Field Descriptions 1369
30-30. uPP DMA Channel Q Descriptor 1 Register (UPQD1) Field Descriptions 1369
30-31. uPP DMA Channel Q Descriptor 2 Register (UPID2) Field Descriptions 1370
30-32. uPP DMA Channel Q Status 0 Register (UPQS0) Field Descriptions 1371
30-33. uPP DMA Channel Q Status 1 Register (UPQS1) Field Descriptions 1371
30-34. uPP DMA Channel Q Status 2 Register (UPQS2) Field Descriptions 1372
31-1. USB Clock Multiplexing Options 1375
31-2. PHY PLL Clock Frequencies Supported 1376
31-3. USB Terminal Functions 1376
31-4. PERI_TXCSR Register Bit Configuration for Bulk IN Transactions 1391
31-5. PERI_RXCSR Register Bit Configuration for Bulk OUT Transactions 1392
31-6. PERI_TXCSR Register Bit Configuration for Isochronous IN Transactions 1394
31-7. PERI_RXCSR Register Bit Configuration for Isochronous OUT Transactions 1396
31-8. Host Packet Descriptor Word 0 (HPD Word 0) 1417
31-9. Host Packet Descriptor Word 1 (HPD Word 1) 1417
31-10. Host Packet Descriptor Word 2 (HPD Word 2) 1418
31-11. Host Packet Descriptor Word 3 (HPD Word 3) 1418
31-12. Host Packet Descriptor Word 4 (HPD Word 4) 1418
31-13. Host Packet Descriptor Word 5 (HPD Word 5) 1418
31-14. Host Packet Descriptor Word 6 (HPD Word 6) 1419
31-15. Host Packet Descriptor Word 7 (HPD Word 7) 1419
31-16. Host Buffer Descriptor Word 0 (HBD Word 0) 1420
31-17. Host Buffer Descriptor Word 1 (HBD Word 1) 1420
31-18. Host Buffer Descriptor Word 2 (HBD Word 2) 1420
31-19. Host Buffer Descriptor Word 3 (HBD Word 3) 1420
31-20. Host Buffer Descriptor Word 4 (HBD Word 4) 1421
31-21. Host Buffer Descriptor Word 5 (HBD Word 5) 1421
31-22. Host Buffer Descriptor Word 6 (HBD Word 6) 1421
31-23. Host Buffer Descriptor Word 7 (HBD Word 7) 1421
31-24. Teardown Descriptor Word 0 1422
31-25. Teardown Descriptor Words 1-7 1422
31-26. Allocation of Queues 1423
31-27. Interrupts Generated by the USB Controller 1437
31-28. USB Interrupt Conditions 1437
31-29. USB Interrupts 1440

Copyright © 2013–2016, Texas Instruments Incorporated

31-30. Universal Serial Bus OTG (USB0) Registers 1453
31-31. Revision Identification Register (REVID) Field Descriptions 1460
31-32. Control Register (CTRLR) Field Descriptions 1460
31-33. Status Register (STATR) Field Descriptions 1461
31-34. Emulation Register (EMUR) Field Descriptions 1461
31-35. Mode Register (MODE) Field Descriptions 1462
31-36. Auto Request Register (AUTOREQ) Field Descriptions 1464
31-37. SRP Fix Time Register (SRPFIXTIME) Field Descriptions 1465
31-38. Teardown Register (TEARDOWN) Field Descriptions 1465
31-39. USB Interrupt Source Register (INTSRCR) Field Descriptions 1466
31-40. USB Interrupt Source Set Register (INTSETR) Field Descriptions 1467
31-41. USB Interrupt Source Clear Register (INTCLRR) Field Descriptions 1468
31-42. USB Interrupt Mask Register (INTMSKR) Field Descriptions 1469
31-43. USB Interrupt Mask Set Register (INTMSKSETR) Field Descriptions 1470
31-44. USB Interrupt Mask Clear Register (INTMSKCLRR) Field Descriptions 1471
31-45. USB Interrupt Source Masked Register (INTMASKEDR) Field Descriptions 1472
31-46. USB End of Interrupt Register (EOIR) Field Descriptions 1473
31-47. Generic RNDIS EP1 Size Register (GENRNDISSZ1) Field Descriptions 1473
31-48. Generic RNDIS EP2 Size Register (GENRNDISSZ2) Field Descriptions 1474
31-49. Generic RNDIS EP3 Size Register (GENRNDISSZ3) Field Descriptions 1474
31-50. Generic RNDIS EP4 Size Register (GENRNDISSZ4) Field Descriptions 1475
31-51. Function Address Register (FADDR) Field Descriptions 1475
31-52. Power Management Register (POWER) Field Descriptions 1476
31-53. Interrupt Register for Endpoint 0 Plus Transmit Endpoints 1 to 4 (INTRTX)Field Descriptions 1477
31-54. Interrupt Register for Receive Endpoints 1 to 4 (INTRRX) Field Descriptions 1478
31-55. Interrupt Enable Register for INTRTX (INTRTXE) Field Descriptions 1479
31-56. Interrupt Enable Register for INTRRX (INTRRXE) Field Descriptions 1479
31-57. Interrupt Register for Common USB Interrupts (INTRUSB) Field Descriptions 1480
31-58. Interrupt Enable Register for INTRUSB (INTRUSBE) Field Descriptions 1481
31-59. Frame Number Register (FRAME) Field Descriptions 1481
31-60. Index Register for Selecting the Endpoint Status and Control Registers (INDEX)Field Descriptions 1482
31-61. Register to Enable the USB 2.0 Test Modes (TESTMODE) Field Descriptions 1482
31-62. Maximum Packet Size for Peripheral/Host Transmit Endpoint (TXMAXP) Field Descriptions 1483
31-63. Control Status Register for Endpoint 0 in Peripheral Mode (PERI_CSR0) Field Descriptions 1484
31-64. Control Status Register for Endpoint 0 in Host Mode (HOST_CSR0) Field Descriptions 1485
31-65. Control Status Register for Peripheral Transmit Endpoint (PERI_TXCSR) Field Descriptions 1486
31-66. Control Status Register for Host Transmit Endpoint (HOST_TXCSR) Field Descriptions 1487
31-67. Maximum Packet Size for Peripheral Host Receive Endpoint (RXMAXP) Field Descriptions 1488
31-68. Control Status Register for Peripheral Receive Endpoint (PERI_RXCSR) Field Descriptions 1489
31-69. Control Status Register for Host Receive Endpoint (HOST_RXCSR) Field Descriptions 1490
31-70. Count 0 Register (COUNT0) Field Descriptions 1491
31-71. Receive Count Register (RXCOUNT) Field Descriptions 1491
31-72. Type Register (Host mode only) (HOST_TYPE0) Field Descriptions 1492
31-73. Transmit Type Register (Host mode only) (HOST_TXTYPE) Field Descriptions 1492
31-74. NAKLimit0 Register (Host mode only) (HOST_NAKLIMIT0) Field Descriptions 1493
31-75. Transmit Interval Register (Host mode only) (HOST_TXINTERVAL) Field Descriptions 1493
31-76. Receive Type Register (Host mode only) (HOST_RXTYPE) Field Descriptions 1494
31-77. Receive Interval Register (Host mode only) (HOST_RXINTERVAL) Field Descriptions 1495
31-78. Configuration Data Register (CONFIGDATA) Field Descriptions 1496

Copyright © 2013–2016, Texas Instruments Incorporated

31-79. Transmit and Receive FIFO Register for Endpoint 0 (FIFO0) Field Descriptions 1497
31-80. Transmit and Receive FIFO Register for Endpoint 1 (FIFO1) Field Descriptions 1497
31-81. Transmit and Receive FIFO Register for Endpoint 2 (FIFO2) Field Descriptions 1498
31-82. Transmit and Receive FIFO Register for Endpoint 3 (FIFO3) Field Descriptions 1498
31-83. Transmit and Receive FIFO Register for Endpoint 4 (FIFO4) Field Descriptions 1499
31-84. Device Control Register (DEVCTL) Field Descriptions 1499
31-85. Transmit Endpoint FIFO Size (TXFIFOSZ) Field Descriptions 1500
31-86. Receive Endpoint FIFO Size (RXFIFOSZ) Field Descriptions 1500
31-87. Transmit Endpoint FIFO Address (TXFIFOADDR) Field Descriptions 1501
31-88. Receive Endpoint FIFO Address (RXFIFOADDR) Field Descriptions 1501
31-89. Hardware Version Register (HWVERS) Field Descriptions 1502
31-90. Transmit Function Address (TXFUNCADDR) Field Descriptions 1503
31-91. Transmit Hub Address (TXHUBADDR) Field Descriptions 1503
31-92. Transmit Hub Port (TXHUBPORT) Field Descriptions 1503
31-93. Receive Function Address (RXFUNCADDR) Field Descriptions 1504
31-94. Receive Hub Address (RXHUBADDR) Field Descriptions 1504
31-95. Receive Hub Port (RXHUBPORT) Field Descriptions 1504
31-96. CDMA Revision Identification Register (DMAREVID) Field Descriptions 1505
31-97. CDMA Teardown Free Descriptor Queue Control Register (TDFDQ) Field Descriptions 1505
31-98. CDMA Emulation Control Register (DMAEMU) Field Descriptions 1506
31-99. CDMA Transmit Channel *n* Global Configuration Registers (TXGCR[*n*]) Field Descriptions 1506
31-100. CDMA Receive Channel *n* Global Configuration Registers (RXGCR[*n*]) Field Descriptions 1507
31-101. Receive Channel *n* Host Packet Configuration Registers A (RXHPCRA[*n*]) Field Descriptions 1508
31-102. Receive Channel *n* Host Packet Configuration Registers B (RXHPCRB[*n*]) Field Descriptions 1509
31-103. CDMA Scheduler Control Register (DMA_SCHED_CTRL) Field Descriptions 1510
31-104. CDMA Scheduler Table Word *n* Registers (WORD[*n*]) Field Descriptions 1510
31-105. Queue Manager Revision Identification Register (QMGRREVID) Field Descriptions 1512
31-106. Queue Manager Queue Diversion Register (DIVERSION) Field Descriptions 1512
31-107. Queue Manager Free Descriptor/Buffer Starvation Count Register 0 (FDBSC0) Field Descriptions 1513
31-108. Queue Manager Free Descriptor/Buffer Starvation Count Register 1 (FDBSC1) Field Descriptions 1514
31-109. Queue Manager Free Descriptor/Buffer Starvation Count Register 2 (FDBSC2) Field Descriptions 1515
31-110. Queue Manager Free Descriptor/Buffer Starvation Count Register 3 (FDBSC3) Field Descriptions 1516
31-111. Queue Manager Linking RAM Region 0 Base Address Register (LRAM0BASE) Field Descriptions 1516
31-112. Queue Manager Linking RAM Region 0 Size Register (LRAM0SIZE) Field Descriptions 1517
31-113. Queue Manager Linking RAM Region 1 Base Address Register (LRAM1BASE) Field Descriptions 1517
31-114. Queue Manager Queue Pending Register 0 (PEND0) Field Descriptions 1518
31-115. Queue Manager Queue Pending Register 1 (PEND1) Field Descriptions 1518
31-116. Queue Manager Memory Region *R* Base Address Registers (QMEMRBASE[*R*]) Field Descriptions 1519
31-117. Queue Manager Memory Region *R* Control Registers (QMEMRCTRL[*R*]) Field Descriptions 1520
31-118. Queue Manager Queue *N* Control Register D (CTRLD[*N*]) Field Descriptions 1521
31-119. Queue Manager Queue *N* Status Register A (QSTATA[*N*]) Field Descriptions 1522
31-120. Queue Manager Queue *N* Status Register B (QSTATB[*N*]) Field Descriptions 1522
31-121. Queue Manager Queue *N* Status Register C (QSTATC[*N*]) Field Descriptions 1523
32-1. Supported Formats on VPIF 1528
32-2. Input and Output Usage Combinations on VPIF 1528
32-3. Receive Pin Multiplexing Control 1529
32-4. Transmit Pin Multiplexing Control 1529
32-5. Video Port Interface (VPIF) Registers 1550
32-6. VPIF Revision ID Register (REVID) Field Descriptions 1553

Copyright © 2013–2016, Texas Instruments Incorporated

32-7. Channel 0 Control Register (C0CTRL) Field Descriptions 1553
32-8. Channel 1 Control Register (C1CTRL) Field Descriptions 1555
32-9. Channel 2 Control Register (C2CTRL) Field Descriptions 1556
32-10. Channel 3 Control Register (C3CTRL) Field Descriptions 1558
32-11. Interrupt Enable Register (INTEN) Field Descriptions 1560
32-12. Interrupt Enable Set Register (INTSET) Field Descriptions 1561
32-13. Interrupt Enable Clear Register (INTCLR) Field Descriptions 1562
32-14. Interrupt Status Register (INTSTAT) Field Descriptions 1563
32-15. Interrupt Status Clear Register (INTSTATCLR) Field Descriptions 1564
32-16. Emulation Suspend Control Register (EMUCTRL) Field Descriptions 1565
32-17. DMA Size Control Register (REQSIZE) Field Descriptions 1565
32-18. Channel *n* Top Field Luminance Address Register (C*n*TLUMA) Field Descriptions 1566
32-19. Channel *n* Bottom Field Luminance Address Register (C*n*BLUMA) Field Descriptions 1566
32-20. Channel *n* Top Field Chrominance Address Register (C*n*TCHROMA) Field Descriptions 1567
32-21. Channel *n* Bottom Field Chrominance Address Register (C*n*BCHROMA) Field Descriptions 1567
32-22. Channel *n* Top Field Horizontal Ancillary Address Register (C*n*THANC) Field Descriptions 1568
32-23. Channel *n* Bottom Field Horizontal Ancillary Address Register (C*n*BHANC) Field Descriptions 1568
32-24. Channel *n* Top Field Vertical Ancillary Address Register (C*n*TVANC) Field Descriptions 1569
32-25. Channel *n* Bottom Field Vertical Ancillary Data Buffer Start Address Register (C*n*BVANC) Field Descriptions 1569
32-26. Channel *n* Image Address Offset Register (C*n*IMGOFFSET) Field Descriptions 1570
32-27. Channel *n* Horizontal Ancillary Address Offset Register (C*n*HANCOFFSET) Field Descriptions 1570
32-28. Channel *n* Horizontal Size Configuration Register (C*n*HCFG) Field Descriptions 1571
32-29. Channel *n* Vertical Size Configuration 0 Register (C*n*VCFG0) Field Descriptions 1572
32-30. Channel *n* Vertical Size Configuration 1 Register (C*n*VCFG1) Field Descriptions 1572
32-31. Channel *n* Vertical Size Configuration 2 Register (C*n*VCFG2) Field Descriptions 1573
32-32. Channel *n* Vertical Image Size Register (C*n*VSIZE) Field Descriptions 1573
32-33. Channel *n* Horizontal Size Configuration Register (C*n*HCFG) Field Descriptions 1574
32-34. Channel *n* Vertical Size Configuration 0 Register (C*n*VCFG0) Field Descriptions 1575
32-35. Channel *n* Vertical Size Configuration 1 Register (C*n*VCFG1) Field Descriptions 1575
32-36. Channel *n* Vertical Size Configuration 2 Register (C*n*VCFG2) Field Descriptions 1576
32-37. Channel *n* Vertical Image Size Register (C*n*VSIZE) Field Descriptions 1576
32-38. Channel *n* Top Field Horizontal Ancillary Position Register (C*n*THANCPOS) Field Descriptions 1577
32-39. Channel *n* Top Field Horizontal Ancillary Size Register (C*n*THANCSIZE) Field Descriptions 1578
32-40. Channel *n* Bottom Field Horizontal Ancillary Position Register (C*n*BHANCPOS) Field Descriptions 1579
32-41. Channel *n* Bottom Field Horizontal Ancillary Size Register (C*n*BHANCSIZE) Field Descriptions 1580
32-42. Channel *n* Top Field Vertical Ancillary Position Register (C*n*TVANCPOS) Field Descriptions 1581
32-43. Channel *n* Top Field Vertical Ancillary Size Register (C*n*TVANCSIZE) Field Descriptions 1582
32-44. Channel *n* Bottom Field Vertical Ancillary Position Register (C*n*BVANCPOS) Field Descriptions 1583
32-45. Channel *n* Bottom Field Vertical Ancillary Size Register (C*n*BVANCSIZE) Field Descriptions 1584

Copyright © 2013–2016, Texas Instruments Incorporated

Read This First

About This Manual

This Technical Reference Manual (TRM) describes the System-on-Chip (SoC) and each peripheral in the device. The SoC consists of the following primary components

- ARM subsystem and associated memories
- A set of I/O peripherals

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.
- 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 below. A legend explains the notation used for the properties.
  - Reserved bits in a register figure designate a bit that is used for future device expansion.

Code Composer Studio is a trademark of Texas Instruments.
ARM926EJ-S, Jazelle are trademarks of ARM Limited.
SPI is a trademark of Motorola, Inc..
SD is a trademark of SanDisk Corporation.

Copyright © 2013–2016, Texas Instruments Incorporated

Overview

Topic Page

1.1 Introduction 74
1.2 ARM Subsystem 74

Copyright © 2013–2016, Texas Instruments Incorporated

1.1 Introduction

The AM1806 ARM microprocessor contains an ARM RISC CPU for general-purpose processing and systems control. The AM1806 ARM microprocessor consists of the following primary components:

- ARM subsystem and associated memories
- A set of I/O peripherals
- A powerful DMA subsystem and SDRAM EMIF interface

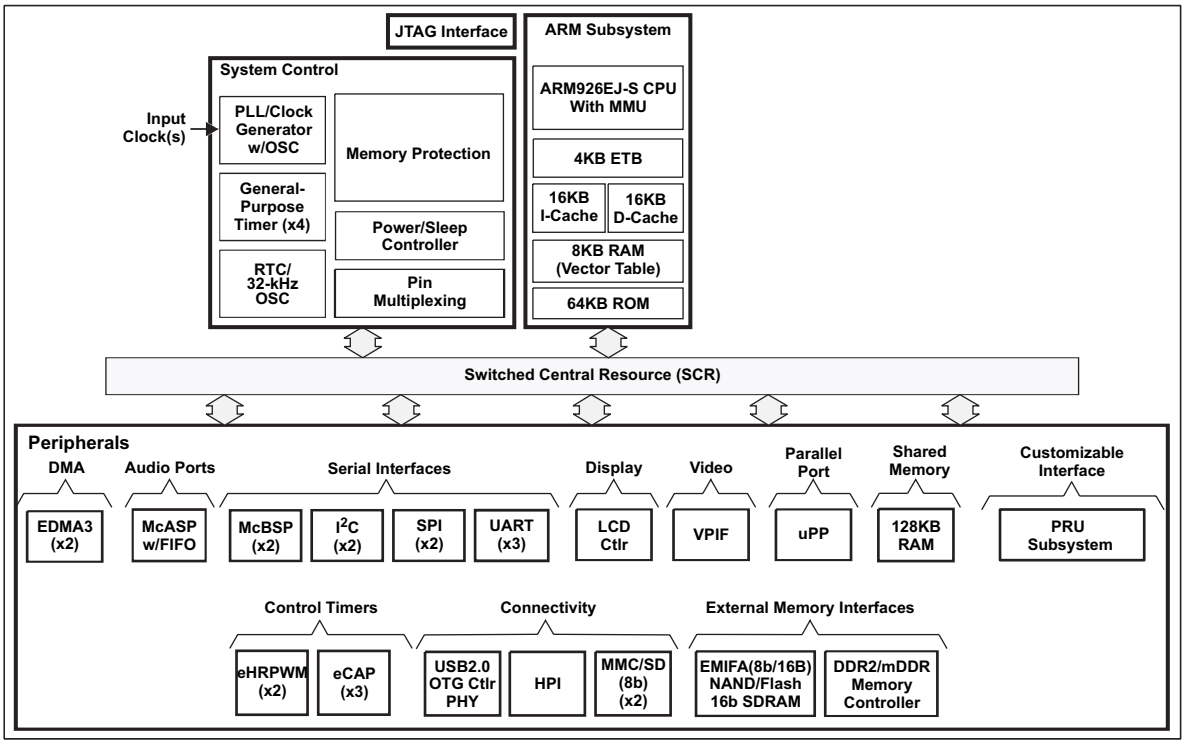
Block Diagram

A block diagram for the AM1806 ARM Microprocessor is shown in Figure 1-1.

1.2 ARM Subsystem

The ARM926EJ-S™ 32-bit RISC CPU in the ARM subsystem (ARMSS) acts as the overall system controller. The ARM CPU performs general system control tasks, such as system initialization, configuration, power management, user interface, and user command implementation. The *ARM Subsystem* chapter describes the ARMSS components and system control functions that the ARM core performs.

Figure 1-1. AM1806 ARM Microprocessor Block Diagram



Note: Not all peripherals are available at the same time due to multiplexing.

DMA Subsystem

The DMA subsystem includes two instances of the enhanced DMA controller (EDMA3). For more information, see the *Enhanced Direct Memory Access (EDMA3) Controller* chapter.

Copyright © 2013–2016, Texas Instruments Incorporated

ARM Subsystem

Topic **Page**

2.1	Introduction	76
2.2	Operating States/Modes	77
2.3	Processor Status Registers	77
2.4	Exceptions and Exception Vectors	78
2.5	The 16-BIS/32-BIS Concept	79
2.6	16-BIS/32-BIS Advantages	79
2.7	Co-Processor 15 (CP15)	80

Copyright © 2013–2016, Texas Instruments Incorporated

2.1 Introduction

This chapter describes the ARM subsystem and its associated memories. The ARM subsystem consists of the following components:

- ARM926EJ-S™ 32-bit RISC CPU
- 16-KB Instruction cache
- 16-KB Data cache
- Memory Management Unit (MMU)
- Co-Processor 15 (CP15) to control MMU, cache, etc.
- Jazelle™ Java Accelerator
- ARM Internal Memory
  - 8 KB RAM
  - 64 KB built-in ROM
- Embedded Trace Module and Embedded Trace Buffer (ETM/ETB)
- Features:
  - The main write buffer has a 16-word data buffer and a 4-address buffer
  - Support for 32-bit ARM/16-bit THUMB instruction sets
  - Fixed little-endian memory format

The ARM926EJ-S processor is a member of the ARM9 family of general-purpose microprocessors. The ARM926EJ-S processor targets multi-tasking applications where full memory management, high performance, low die size, and low power are all important.

The ARM926EJ-S processor supports the 32-bit ARM and the 16-bit THUMB instruction sets, enabling you to trade off between high performance and high code density. This includes features for efficient execution of Java byte codes and providing Java performance similar to Just in Time (JIT) Java interpreter without associated code overhead.

The ARM926EJ-S processor supports the ARM debug architecture and includes logic to assist in both hardware and software debugging. The ARM926EJ-S processor has a Harvard architecture and provides a complete high performance subsystem, including the following:

- An ARM926EJ-S integer core
- A Memory Management Unit (MMU)
- Separate instruction and data Advanced Microcontroller Bus Architecture (AHBA) Advanced High Performance Bus (AHB) bus interfaces

NOTE: There is no TCM memory and interface on this device.

The ARM926EJ-S processor implements ARM architecture version 5TEJ.

The ARM core also has 8 KB RAM (typically used for vector table) and 64 KB ROM (for boot images) associated with it. The RAM/ROM locations are not accessible by any other master peripherals. Furthermore, the ARM has DMA and CFG bus master ports via the AHB interface.

Copyright © 2013–2016, Texas Instruments Incorporated

2.2 Operating States/Modes

The ARM can operate in two states: ARM (32-bit) mode and THUMB (16-bit) mode. You can switch the ARM926EJ-S processor between ARM mode and THUMB mode using the BX instruction.

The ARM can operate in the following modes:

- User mode (USR): Non-privileged mode, usually for the execution of most application programs.
- Fast interrupt mode (FIQ): Fast interrupt processing
- Interrupt mode (IRQ): Normal interrupt processing
- Supervisor mode (SVC): Protected mode of execution for operating systems
- Abort mode (ABT): Mode of execution after a data abort or a pre-fetch abort
- System mode (SYS): Privileged mode of execution for operating systems
- Undefined mode (UND): Executing an undefined instruction causes the ARM to enter undefined mode.

You can only enter privileged modes (system or supervisor) from other privileged modes.

To enter supervisor mode from user mode, generate a software interrupt (SWI). An IRQ interrupt causes the processor to enter the IRQ mode. An FIQ interrupt causes the processor to enter the FIQ mode.

Different stacks must be set up for different modes. The stack pointer (SP) automatically changes to the SP of the mode that was entered.

2.3 Processor Status Registers

The processor status register (PSR) controls the enabling and disabling of interrupts and setting the mode of operation of the processor. The 8 least-significant bits PSR[7:0] are the control bits of the processor. PSR[27:8] are reserved bits and PSR[31:28] are status registers. The details of the control bits are:

- Bit 7 - I bit: Disable IRQ (I =1) or enable IRQ (I = 0)
- Bit 6 - F bit: Disable FIQ (F = 1) or enable FIQ (F = 0)
- Bit 5 - T bit: Controls whether the processor is in thumb mode (T = 1) or ARM mode (T = 0)
- Bits 4:0 Mode: Controls the mode of operation of the processor
  - PSR [4:0] = 10000 : User mode
  - PSR [4:0] = 10001 : FIQ mode
  - PSR [4:0] = 10010 : IRQ mode
  - PSR [4:0] = 10011 : Supervisor mode
  - PSR [4:0] = 10111 : Abort mode
  - PSR [4:0] = 11011 : Undefined mode
  - PSR [4:0] = 11111 : System mode

Status bits show the result of the most recent ALU operation. The details of status bits are:

- Bit 31 - N bit: Negative or less than
- Bit 30 - Z bit: Zero
- Bit 29 - C bit: Carry or borrow
- Bit 28 - V bit: Overflow or underflow

NOTE: See the Programmer's Model of the ARM926EJ-S Technical Reference Manual (TRM), downloadable from http://infocenter.arm.com/help/index.jsp for more detailed information.

Copyright © 2013–2016, Texas Instruments Incorporated

2.4 Exceptions and Exception Vectors

Exceptions arise when the normal flow of the program must be temporarily halted. The exceptions that occur in an ARM system are given below:

- Reset exception: processor reset
- FIQ interrupt: fast interrupt
- IRQ interrupt: normal interrupt
- Abort exception: abort indicates that the current memory access could not be completed. The abort could be a pre-fetch abort or a data abort.
- SWI interrupt: use software interrupt to enter supervisor mode.
- Undefined exception: occurs when the processor executes an undefined instruction

The exceptions in the order of highest priority to lowest priority are: reset, data abort, FIQ, IRQ, pre-fetch abort, undefined instruction, and SWI. SWI and undefined instruction have the same priority. The ARM is configured with the VINITHI signal set high (VINITHI = 1), such that the vector table is located at address FFFF 0000h. This address maps to the beginning of the ARM local RAM (8 KB).

NOTE: The VINITHI signal is configurable by way of the register setting in CP15. However, it is not recommended to set VINITHI = 0, as the device has no physical memory in the 0000 0000h address region.

The default vector table is shown in Table 2-1.

Table 2-1. Exception Vector Table for ARM

Vector Offset Address	Exception	Mode on entry	I Bit State on Entry	F Bit State on Entry
0h	Reset	Supervisor	Set	Set
4h	Undefined instruction	Undefined	Set	Unchanged
8h	Software interrupt	Supervisor	Set	Unchanged
Ch	Pre-fetch abort	Abort	Set	Unchanged
10h	Data abort	Abort	Set	Unchanged
14h	Reserved	—	—	—
18h	IRQ	IRQ	Set	Unchanged
1Ch	FIQ	FIQ	Set	Set

Copyright © 2013–2016, Texas Instruments Incorporated

2.5 The 16-BIS/32-BIS Concept

The key idea behind 16-BIS is that of a super-reduced instruction set. Essentially, the ARM926EJ processor has two instruction sets:

- ARM mode or 32-BIS: the standard 32-bit instruction set
- THUMB mode or 16-BIS: a 16-bit instruction set

The 16-bit instruction length (16-BIS) allows the 16-BIS to approach twice the density of standard 32-BIS code while retaining most of the 32-BIS's performance advantage over a traditional 16-bit processor using 16-bit registers. This is possible because 16-BIS code operates on the same 32-bit register set as 32-BIS code. 16-bit code can provide up to 65% of the code size of the 32-bit code and 160% of the performance of an equivalent 32-BIS processor connected to a 16-bit memory system.

2.6 16-BIS/32-BIS Advantages

16-bit instructions operate with the standard 32-bit register configuration, allowing excellent inter-operability between 32-BIS and 16-BIS states. Each 16-bit instruction has a corresponding 32-bit instruction with the same effect on the processor model. The major advantage of a 32-bit architecture over a 16-bit architecture is its ability to manipulate 32-bit integers with single instructions, and to address a large address space efficiently. When processing 32-bit data, a 16-bit architecture takes at least two instructions to perform the same task as a single 32-bit instruction. However, not all of the code in a program processes 32-bit data (for example, code that performs character string handling), and some instructions (like branches) do not process any data at all. If a 16-bit architecture only has 16-bit instructions, and a 32-bit architecture only has 32-bit instructions, then the 16-bit architecture has better code density overall, and has better than one half of the performance of the 32-bit architecture. Clearly, 32-bit performance comes at the cost of code density. The 16-bit instruction breaks this constraint by implementing a 16-bit instruction length on a 32-bit architecture, making the processing of 32-bit data efficient with compact instruction coding. This provides far better performance than a 16-bit architecture, with better code density than a 32-bit architecture. The 16-BIS also has a major advantage over other 32-bit architectures with 16-bit instructions. The advantage is the ability to switch back to full 32-bit code and execute at full speed. Thus, critical loops for applications such as fast interrupts can be coded using the full 32-BIS and linked with 16-BIS code. The overhead of switching from 16-bit code to 32-bit code is folded into sub-routine entry time. Various portions of a system can be optimized for speed or for code density by switching between 16-BIS and 32-BIS execution, as appropriate.

Copyright © 2013–2016, Texas Instruments Incorporated

2.7 Co-Processor 15 (CP15)

The system control coprocessor (CP15) is used to configure and control instruction and data caches, Tightly-Coupled Memories (TCMs), Memory Management Units (MMUs), and many system functions. The CP15 registers are only accessible with MRC and MCR instructions by the ARM in a privileged mode like supervisor mode or system mode.

2.7.1 Addresses in an ARM926EJ-S System

Three different types of addresses exist in an ARM926EJ-S system. They are listed in Table 2-2.

Table 2-2. Different Address Types in ARM System

Domain	ARM9EJ-S	Caches and MMU	TCM and AMBA Bus
Address type	Virtual Address (VA)	Modified Virtual Address (MVA)	Physical Address (PA)

An example of the address manipulation that occurs when the ARM9EJ-S core requests an instruction is shown in Example 2-1

Example 2-1. Address Manipulation

The VA of the instruction is issued by the ARM9EJ-S core.

The VA is translated to the MVA. The Instruction Cache (Icache) and Memory Management Unit (MMU) detect the MVA.

If the protection check carried out by the MMU on the MVA does not abort and the MVA tag is in the Icache, the instruction data is returned to the ARM9EJ-S core.

If the protection check carried out by the MMU on the MVA does not abort, and the MVA tag is not in the cache, then the MMU translates the MVA to produce the PA.

NOTE: See the Programmers Model of the ARM926EJ-S Technical Reference Manual (TRM), downloadable from http://infocenter.arm.com/help/index.jsp for more detailed information.

2.7.2 Memory Management Unit (MMU)

The ARM926EJ-S MMU provides virtual memory features required by operating systems such as SymbianOS, WindowsCE, and Linux. A single set of two level page tables stored in main memory controls the address translation, permission checks, and memory region attributes for both data and instruction accesses. The MMU uses a single unified Translation Lookaside Buffer (TLB) to cache the information held in the page tables.

The MMU features are as follows:

- Standard ARM architecture v4 and v5 MMU mapping sizes, domains, and access protection scheme.
- Mapping sizes are 1 MB (sections), 64 KB (large pages), 4 KB (small pages) and 1 KB (tiny pages)
- Access permissions for large pages and small pages can be specified separately for each quarter of the page (subpage permissions)
- Hardware page table walks
- Invalidate entire TLB, using CP15 register 8
- Invalidate TLB entry, selected by MVA, using CP15 register 8
- Lockdown of TLB entries, using CP15 register 10

NOTE: See the Memory Management Unit of the ARM926EJ-S Technical Reference Manual (TRM), downloadable from http://infocenter.arm.com/help/index.jsp for more detailed information.

Copyright © 2013–2016, Texas Instruments Incorporated

2.7.3 Caches and Write Buffer

The ARM926EJ-S processor includes:

- An Instruction cache (Icache)
- A Data cache (Dcache)
- A write buffer

The size of the data cache is 16 KB, instruction cache is 16 KB, and write buffer is 17 bytes.

The caches have the following features:

- Virtual index, virtual tag, addressed using the Modified Virtual Address (MVA)
- Four-way set associative, with a cache line length of eight words per line (32 bytes per line), and two dirty bits in the Dcache
- Dcache supports write-through and write-back (or copy back) cache operation, selected by memory region using the C and B bits in the MMU translation tables
- Perform critical-word first cache refilling
- Cache lockdown registers enable control over which cache ways are used for allocation on a line fill, providing a mechanism for both lockdown and controlling cache pollution.
- Dcache stores the Physical Address TAG (PA TAG) corresponding to each Dcache entry in the TAGRAM for use during the cache line write-backs, in addition to the Virtual Address TAG stored in the TAG RAM. This means that the MMU is not involved in Dcache write-back operations, removing the possibility of TLB misses related to the write-back address.
- Cache maintenance operations to provide efficient invalidation of the following:
  - The entire Dcache or Icache
  - Regions of the Dcache or Icache
  - The entire Dcache
  - Regions of virtual memory
- They also provide operations for efficient cleaning and invalidation of the following:
  - The entire Dcache
  - Regions of the Dcache
  - Regions of virtual memory

The write buffer is used for all writes to a non-cachable bufferable region, write-through region, and write misses to a write-back region. A separate buffer is incorporated in the Dcache for holding write-back for cache line evictions or cleaning of dirty cache lines.

The main write buffer has a 16-word data buffer and a four-address buffer.

The Dcache write-back has eight data word entries and a single address entry.

The MCR drain write buffer enables both write buffers to be drained under software control.

The MCR wait for interrupt causes both write buffers to be drained and the ARM926EJ-S processor to be put into a low power state until an interrupt occurs.

NOTE: See the Caches and Write Buffer of the ARM926EJ-S Technical Reference Manual (TRM), downloadable from http://infocenter.arm.com/help/index.jsp for more detailed information.

Copyright © 2013–2016, Texas Instruments Incorporated

System Interconnect

Topic Page

3.1 Introduction 83
3.2 System Interconnect Block Diagram 84

Copyright © 2013–2016, Texas Instruments Incorporated

3.1 Introduction

The ARM, the Programmable Real-Time Unit (PRU) subsystem, the EDMA3 transfer controllers, and the device peripherals are interconnected through a switch fabric architecture (see Section 3.2). The switch fabric is composed of multiple switched central resources (SCRs) and multiple bridges. The SCRs establish low-latency connectivity between master peripherals and slave peripherals.

Additionally, the SCRs provide priority-based arbitration and facilitate concurrent data movement between master and slave peripherals. Bridges are mainly used to perform bus-width conversion as well as bus operating frequency conversion.

The ARM, the PRU subsystem, the EDMA3 transfer controllers, and the various device peripherals can be classified into two categories: master peripherals and slave peripherals. Master peripherals are typically capable of initiating read and write transfers in the system and do not rely on the EDMA3 or on a CPU to perform transfers to and from them. The system master peripherals include the ARM, the EDMA3 transfer controllers, HPI, LCDC, PRU subsystems, USB2.0, uPP, and VPIF. Not all master peripherals may connect to all slave peripherals. The supported connections are designated by an X in Table 3-1.

Table 3-1. AM1806 ARM Microprocessor System Interconnect Matrix

Masters		Slaves							
Master	Default Priority	ARM ROM, AINTC	ARM RAM	EMIFA	DDR2/ mDDR	128K RAM	EDMA3_0_ TC0/TC1	EDMA3_1_ TC0	Peripheral Group[1]
EDMA3_0_CC0	0						X		
EDMA3_1_CC0	0							X	
EDMA3_0_TC0	0			X	X	X	X	X	X
EDMA3_0_TC1	0			X	X	X	X	X	X
PRU0	0		X	X	X	X	X	X	X
PRU1	0			X	X	X	X	X	X
ARM I	2	X	X	X	X	X			
ARM D	2	X	X	X	X	X	X	X	X
EDMA3_1_TC0	4			X	X	X	X	X	X
uPP	4			X	X	X			
USB2.0	4			X	X	X			
VPIF	4			X	X	X			
LCDC	5				X				
HPI	6			X	X	X			X[2]

(1) Peripheral group: SYSCFG, eCAP0, eCAP1, eCAP2, eHRPWM0, eHRPWM1, GPIO, I2C0, I2C1, LCDC, McASP0, McBSP0, McBSP1, MDIO, MMC/SD0, MMC/SD1, PLLC0, PLLC1, PRU RAM0, PRU RAM1, PRU Config, PSC0, PSC1, RTC, SPI0, SPI1, TIMER64P0, TIMER64P1, TIMER64P2, TIMER64P3, EDMA3_0_CC0, EDMA3_1_CC0, UART0, UART1, UART2, HPI, USB0 (USB2.0), uPP, VPIF.

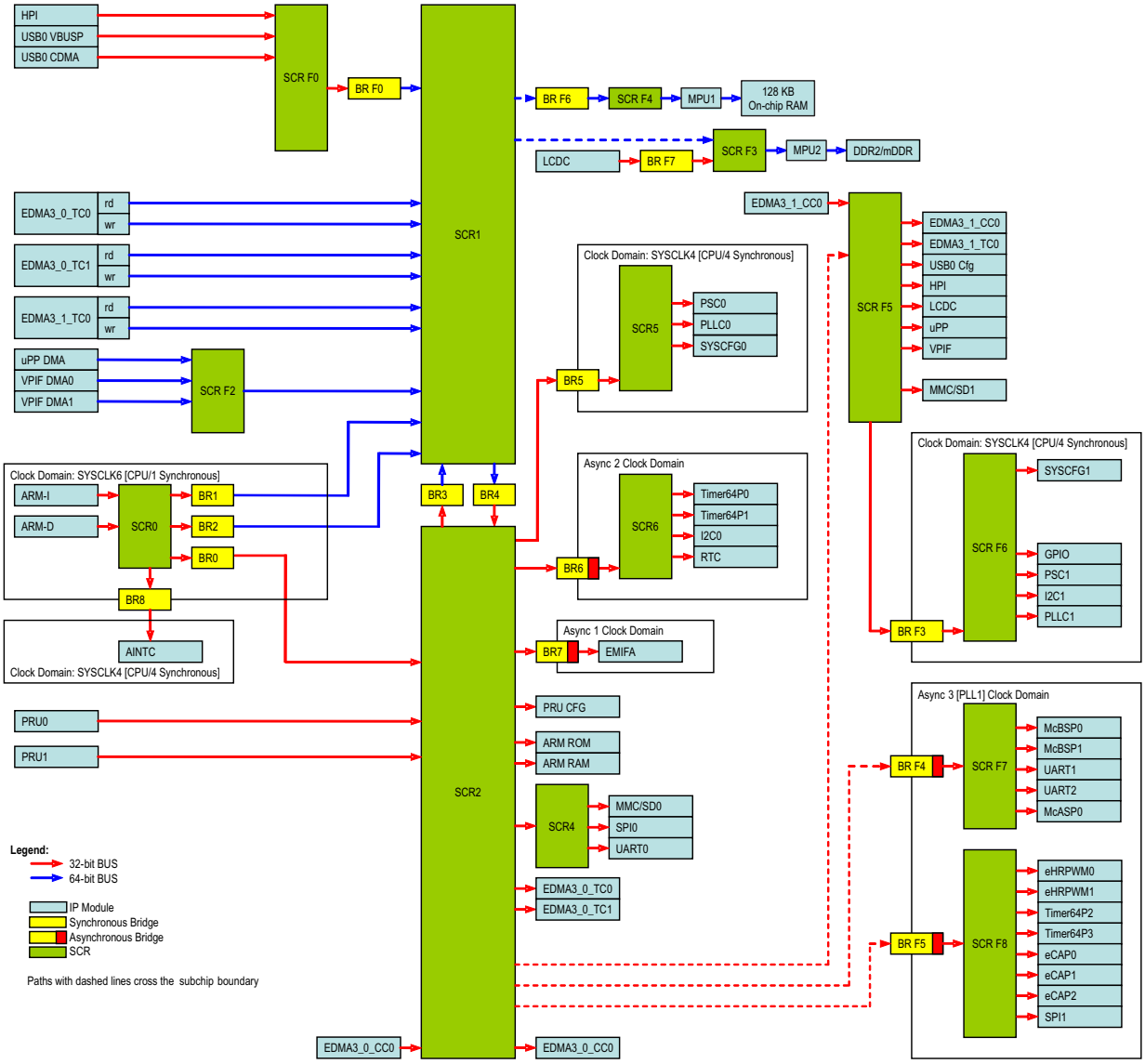
(2) The HPI does not have access to all registers in the SYSCFG module because it operates with the User Privilege Level.

Copyright © 2013–2016, Texas Instruments Incorporated

3.2 System Interconnect Block Diagram

Figure 3-1 shows a system interconnect block diagram.

Figure 3-1. System Interconnect Block Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

System Memory

Topic **Page**

4.1 Introduction **86**
4.2 ARM Memories **86**
4.3 Peripherals **86**

Copyright © 2013–2016, Texas Instruments Incorporated

4.1 Introduction

This device has multiple on-chip/off-chip memories and several external device interfaces associated with the ARM and various subsystems. To help simplify software development, a unified memory-map is used wherever possible to maintain a consistent view of device resources across all masters.

For details on the memory addresses, actual memory supported and accessibility by various bus masters, see the detailed memory-map information in the device-specific data manual.

4.2 ARM Memories

The configuration for the ARM internal memory is:

- 8 KB ARM local RAM
- 64 KB ARM local ROM
- 16 KB Instruction Cache and 16 KB Data cache

The ARM RAM/ROM are only accessible by ARM and PRU0.

On-Chip RAM Memory

This device also offers an on-chip 128-KB single-port RAM, apart from the ARM internal memories. This on-chip RAM is accessible by the ARM, and is also accessible by several master peripherals. Writes to this RAM by all masters is atomic.

External Memories

This device has two external memory interfaces that provide multiple external memory options accessible by the CPU and master peripherals:

- EMIF:
  - 8/16-bit wide asynchronous EMIF module that supports asynchronous devices such as ASRAM, NAND Flash, and NOR Flash (up to 4 devices)
  - 8/16-bit wide NAND Flash with 4-bit ECC (up to 4 devices)
  - 16-bit SDRAM with 128-MB address space
- DDR2/mDDR memory controller:
  - 16-bit DDR2 with up to 256-MB memory address space
  - 16-bit mDDR with up to 256-MB memory address space

Internal Peripherals

The peripheral only accessible by the ARM is the ARM interrupt controller (AINTC). For more information on the AINTC, see the *ARM Interrupt Controller (AINTC)* chapter.

4.3 Peripherals

The ARM has access to all peripherals. This also includes system modules like the PLL controller (PLLC), the power and sleep controller (PSC), and the system configuration module (SYSCFG). See the device-specific data manual for the complete list of peripherals supported on your device.

Copyright © 2013–2016, Texas Instruments Incorporated

Memory Protection Unit (MPU)

Topic **Page**

5.1 Introduction **88**
5.2 Architecture **89**
5.3 MPU Registers **94**

Copyright © 2013–2016, Texas Instruments Incorporated

5.1 Introduction

This device supports two memory protection units (MPU1 and MPU2). MPU1 supports the 128KB on-chip RAM and MPU2 supports the DDR2/mDDR SDRAM.

5.1.1 Purpose of the MPU

The memory protection unit (MPU) is provided to manage access to memory. The MPU allows you to define multiple ranges and limit access to system masters based on their privilege ID. The MPU can record a detected fault, or invalid access, and notify the system through an interrupt.

5.1.2 Features

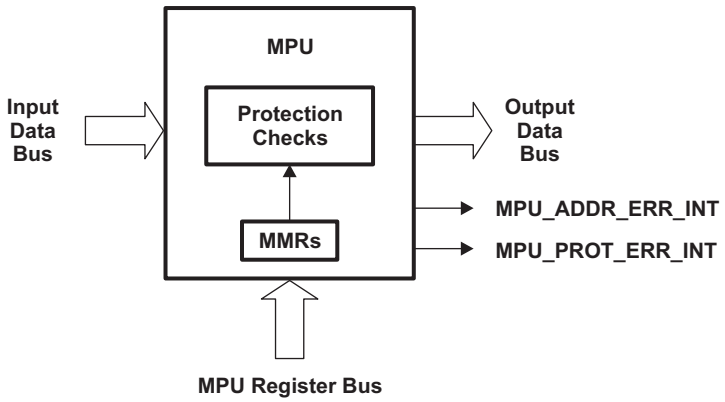
The MPU supports the following features:

- Supports multiple programmable address ranges
- Supports 0 or 1 fixed range
- Supports read, write, and execute access privileges
- Supports privilege ID associations with ranges
- Generates an interrupt when there is a protection violation, and saves violating transfer parameters
- Supports protection of its own registers

5.1.3 Block Diagram

Figure 5-1 shows a block diagram of the MPU. An access to a protected memory must pass through the MPU. During an access, the MPU checks the memory address on the input data bus against fixed and programmable ranges. If allowed, the transfer is passed unmodified to the output data bus. If the transfer fails the protection check then the MPU does not pass the transfer to the output bus but rather services the transfer internally back to the input bus (to prevent a hang) returning the fault status to the requestor as well as generating an interrupt about the fault. The MPU generates two interrupts: an address error interrupt (MPU_ADDR_ERR_INT) and a protection interrupt (MPU_PROT_ERR_INT).

Figure 5-1. MPU Block Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

5.1.4 MPU Default Configuration

Two MPUs are supported on the device, one for the 128KB on-chip RAM and one for the DDR2/mDDR SDRAM. Table 5-1 shows the memory regions protected by each MPU. Table 5-2 shows the configuration of each MPU.

Table 5-1. MPU Memory Regions

Unit	Memory Protection	Memory Region	
		Start Address	End Address
MPU1	128KB On-chip RAM	8000 0000h	8001 FFFFh
MPU2	DDR2/mDDR SDRAM	C000 0000h	DFFF FFFFh

Table 5-2. MPU Default Configuration

Setting	MPU1	MPU2
Default permission	Assume allowed	Assume allowed
Number of allowed IDs supported	12	12
Number of fixed ranges supported	1	0
Number of programmable ranges supported	6	12
Compare width	1 KB granularity	64 KB granularity

5.2 Architecture

5.2.1 Privilege Levels

The privilege level of a memory access determines what level of permissions the originator of the memory access might have. Two privilege levels are supported: supervisor and user.

Supervisor level is generally granted access to peripheral registers and the memory protection configuration. User level is generally confined to the memory spaces that the OS specifically designates for its use.

ARM CPU instruction and data accesses have a privilege level associated with them. See the ARM926EJ-S Technical Reference Manual (TRM), downloadable from http://infocenter.arm.com/help/index.jsp for more details on privilege levels of the ARM CPU.

Although master peripherals like the HPI do not execute code, they still have a privilege level associated with them. Unlike the ARM CPU, the privilege level of this peripheral is fixed.

Table 5-3 shows the privilege ID of the CPU and every mastering peripheral. Table 5-3 also shows the privilege level (supervisor vs. user) and access type (instruction read vs. data/DMA read or write) of each master on the device. In some cases, a particular setting depends on software being executed at the time of the access or the configuration of the master peripheral.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 5-3. Device Master Settings

Master	Privilege ID	Privilege Level	Access Type
EDMA3_0_CC0	Inherited	Inherited	DMA
EDMA3_0_TC0 and EDMA3_0_TC1	Inherited	Inherited	DMA
EDMA3_1_CC0	Inherited	Inherited	DMA
EDMA3_1_TC0	Inherited	Inherited	DMA
ARM (instruction access)	0	Software dependant	Instruction
ARM (data access)	0	Software dependant	Data
PRU0/PRU1	2	Supervisor	DMA
HPI	3	User	DMA
USB2.0	6	Supervisor	DMA
LCD Controller	7	Supervisor	DMA
uPP	8	Supervisor	DMA
VPIF DMA0	10	Supervisor	DMA
VPIF DMA1	11	Supervisor	DMA

5.2.2 Memory Protection Ranges

NOTE: In some cases the amount of physical memory in actual use may be less than the maximum amount of memory supported by the device. For example, the device may support a total of 512 Mbytes of SDRAM memory, but your design may only populate 128 Mbytes. In such cases, the unpopulated memory range must be protected in order to prevent unintended/disallowed aliased access to protected memory. One of the programmable address ranges could be used to detect accesses to this unpopulated memory.

The MPU divides its assigned memory into address ranges. Each MPU can support one fixed address range and multiple programmable address ranges. The fixed address range is configured to an exact address. The programmable address range allows software to program the start and end addresses.

Each address range has the following set of registers:

- Range start and end address registers (MPSAR and MPEAR): Specifies the starting and ending address of the address range.
- Memory protection page attribute register (MPPA): Use to program the permission settings of the address range.

It is allowed to configure ranges such that they overlap each other. In this case, all the overlapped ranges must allow the access, otherwise the access is not allowed. The final permissions given to the access are the lowest of each type of permission from any hit range.

Addresses not covered by a range are either allowed or disallowed based on the configuration of the MPU. The MPU can be configured for assumed allowed or assumed disallowed mode as dictated by the ASSUME_ALLOWED bit in the configuration register (CONFIG).

Copyright © 2013–2016, Texas Instruments Incorporated

5.2.3 Permission Structures

The MPU defines a per-range permission structure with three permission fields in a 32-bit permission entry. Figure 5-2 shows the structure of a permission entry.

Figure 5-2. Permission Fields

31										22	21	20	19	18	17	16
Reserved											Allowed IDs					
											AID11	AID10	AID9	AID8	AID7	AID6
15	14	13	12	11	10	9	8			6	5	4	3	2	1	0
Allowed IDs							Reserved				Access Types					
AID5	AID4	AID3	AID2	AID1	AID0	AIX					SR	SW	SX	UR	UW	UX

5.2.3.1 Requestor-ID Based Access Controls

Each master on the device has an N-bit code associated with it that identifies it for privilege purposes. This privilege ID accompanies all memory accesses made on behalf of that master. That is, when a master triggers a memory access command, the privilege ID will be carried alongside the command.

Each memory protection range has an allowed ID (AID) field associated with it that indicates which requestors may access the given address range. The MPU maps the privilege IDs of all the possible requestors to bits in the allowed IDs field in the memory protection page attribute registers (MPPA).

- AID0 through AID11 are used to specify the allowed privilege IDs.
- An additional allowed ID bit, AIDX, captures access made by all privilege IDs not covered by AID0 through AID11.

When set to 1, the AID bit grants access to the corresponding ID. When cleared to 0, the AID bit denies access to the corresponding requestor.

5.2.3.2 Request-Type Based Permissions

The memory protection model defines three fundamental functional access types: read, write, and execute. Read and write refer to data accesses -- accesses originating via the load/store units on the CPU or via a master peripheral. Execute refers to accesses associated with an instruction fetch.

The memory protection model allows controlling read, write, and execute permissions independently for both user and supervisor mode. This results in six permission bits, listed in Table 5-4. For each bit, a 1 permits the access type and a 0 denies access. For example, UX = 1 means that User Mode may execute from the given page. The memory protection unit allows you to specify all six of these bits separately; 64 different encodings are permitted altogether, although programs might not use all of them.

Table 5-4. Request Type Access Controls

Bit	Field	Description
5	SR	Supervisor may read
4	SW	Supervisor may write
3	SX	Supervisor may execute
2	UR	User may read
1	UW	User may write
0	UX	User may execute

Copyright © 2013–2016, Texas Instruments Incorporated

5.2.4 Protection Check

During a memory access, the MPU checks if the address range of the input transfer overlaps one of the address ranges. When the input transfer address is within a range the transfer parameters are checked against the address range permissions.

The MPU first checks the transfers privilege ID against the AID settings. If the AID bit is 0, then the range will not be checked; if the AID bit is 1, then the transfer parameters are checked against the memory protection page attribute register (MPPA) values to detect an allowed access.

For non-debug accesses, the read, write, and execute permissions are also checked. There is a set of permissions for supervisor mode and a set for user mode. For supervisor mode accesses, the SR, SW, and SX bits are checked. For user mode accesses, the UR, UW, and UX bits are checked.

If the transfer address range does not match any address range then the transfer is either allowed or disallowed based on the configuration of the MPU. The MPU can be configured for assumed allowed or assumed disallowed mode as dictated by the ASSUME_ALLOWED bit in the configuration register (CONFIG).

In the case that a transfer spans multiple address ranges, all the overlapped ranges must allow the access, otherwise the access is not allowed. The final permissions given to the access are the lowest of each type of permission from any hit range. Therefore, if a transfer matches 2 ranges, one that is RW and one that is RX, then the final permission is just R.

5.2.5 MPU Register Protection

Access to the range start and end address registers (MPSAR and MPEAR) and memory protection page attribute registers (MPPA) is also protected. All non-debug writes must be by a supervisor entity. A protection fault can occur from a register write with invalid permissions and this triggers an interrupt just like a memory access.

Faults are not recorded (nor interrupts generated) for debug accesses.

5.2.6 Invalid Accesses and Exceptions

When a transfer fails the protection check, the MPU does not pass the transfer to the output bus. The MPU instead services the transfer locally to prevent a hang and returns a protection error to the requestor. The behavior of the MPU depends on whether the access was a read or a write:

- For a read: The MPU returns 0s, a permission value is 0 (no access allowed), a protection error status.
- For a write: The MPU receives all the write data and returns a protection error status.

The MPU captures system faults due to addressing or protection violations in its registers. The MPU can store the fault information for only one fault, so the first detected fault is recorded into the fault registers and an interrupt is generated. Software must use the fault clear register (FLTCLR) to clear the fault status so that another fault can be recorded. The MPU will not record another fault nor generate another interrupt until the existing fault has been cleared. Also, additional faults will be ignored. Faults are not recorded (no interrupts generated) for debug accesses.

5.2.7 Reset Considerations

After reset, the memory protection page attribute registers (MPPA) default to 0. This disables all protection features.

Copyright © 2013–2016, Texas Instruments Incorporated

5.2.8 Interrupt Support

5.2.8.1 Interrupt Events and Requests

The MPU generates two interrupts: an address error interrupt (MPU_ADDR_ERR_INT) and a protection interrupt (MPU_PROT_ERR_INT). The MPU_ADDR_ERR_INT is generated when there is an addressing violation due to an access to a non-existent location in the MPU register space. The MPU_PROT_ERR_INT interrupt is generated when there is a protection violation of either in the defined ranges or to the MPU registers.

The transfer parameters that caused the violation are saved in the MPU registers.

5.2.8.2 Interrupt Multiplexing

The interrupts from both MPUs are combined with the boot configuration module into a single interrupt called MPU_BOOTCFG_ERR. The combined interrupt is routed to the ARM interrupt controller. Table 5-5 shows the interrupt sources that are combined to make MPU_BOOTCFG_ERR.

Table 5-5. MPU_BOOTCFG_ERR Interrupt Sources

Interrupt	Source
MPU1_ADDR_ERR_INT	MPU1 address error interrupt
MPU1_PROT_ERR_INT	MPU1 protection interrupt
MPU2_ADDR_ERR_INT	MPU2 address error interrupt
MPU2_PROT_ERR_INT	MPU2 protection interrupt
BOOTCFG_ADDR_ERR	Boot configuration address error
BOOTCFG_PROT_ERR	Boot configuration protection error

5.2.9 Emulation Considerations

Memory and MPU registers are not protected against emulation accesses.

Copyright © 2013–2016, Texas Instruments Incorporated

5.3 MPU Registers

There are two MPUs on the device. Each MPU contains a set of memory-mapped registers.

Table 5-7 lists the memory-mapped registers for the MPU2.

Table 5-6. Memory Protection Unit 1 (MPU1) Registers

Address	Acronym	Register Description	Section
01E1 4000h	REVID	Revision identification register	Section 5.3.1
01E1 4004h	CONFIG	Configuration register	Section 5.3.2
01E1 4010h	IRAWSTAT	Interrupt raw status/set register	Section 5.3.3
01E1 4014h	IENSTAT	Interrupt enable status/clear register	Section 5.3.4
01E1 4018h	IENSET	Interrupt enable set register	Section 5.3.5
01E1 401Ch	IENCLR	Interrupt enable clear register	Section 5.3.6
01E1 4200h	PROG1_MPSAR	Programmable range 1 start address register	Section 5.3.10.1
01E1 4204h	PROG1_MPEAR	Programmable range 1 end address register	Section 5.3.11.1
01E1 4208h	PROG1_MPPA	Programmable range 1 memory protection page attributes register	Section 5.3.12
01E1 4210h	PROG2_MPSAR	Programmable range 2 start address register	Section 5.3.10.1
01E1 4214h	PROG2_MPEAR	Programmable range 2 end address register	Section 5.3.11.1
01E1 4218h	PROG2_MPPA	Programmable range 2 memory protection page attributes register	Section 5.3.12
01E1 4220h	PROG3_MPSAR	Programmable range 3 start address register	Section 5.3.10.1
01E1 4224h	PROG3_MPEAR	Programmable range 3 end address register	Section 5.3.11.1
01E1 4228h	PROG3_MPPA	Programmable range 3 memory protection page attributes register	Section 5.3.12
01E1 4230h	PROG4_MPSAR	Programmable range 4 start address register	Section 5.3.10.1
01E1 4234h	PROG4_MPEAR	Programmable range 4 end address register	Section 5.3.11.1
01E1 4238h	PROG4_MPPA	Programmable range 4 memory protection page attributes register	Section 5.3.12
01E1 4240h	PROG5_MPSAR	Programmable range 5 start address register	Section 5.3.10.1
01E1 4244h	PROG5_MPEAR	Programmable range 5 end address register	Section 5.3.11.1
01E1 4248h	PROG5_MPPA	Programmable range 5 memory protection page attributes register	Section 5.3.12
01E1 4250h	PROG6_MPSAR	Programmable range 6 start address register	Section 5.3.10.1
01E1 4254h	PROG6_MPEAR	Programmable range 6 end address register	Section 5.3.11.1
01E1 4258h	PROG6_MPPA	Programmable range 6 memory protection page attributes register	Section 5.3.12
01E1 4300h	FLTADDRR	Fault address register	Section 5.3.13
01E1 4304h	FLTSTAT	Fault status register	Section 5.3.14
01E1 4308h	FLTCLR	Fault clear register	Section 5.3.15

Table 5-7. Memory Protection Unit 2 (MPU2) Registers

Address	Acronym	Register Description	Section
01E1 5000h	REVID	Revision identification register	Section 5.3.1
01E1 5004h	CONFIG	Configuration register	Section 5.3.2
01E1 5010h	IRAWSTAT	Interrupt raw status/set register	Section 5.3.3
01E1 5014h	IENSTAT	Interrupt enable status/clear register	Section 5.3.4
01E1 5018h	IENSET	Interrupt enable set register	Section 5.3.5
01E1 501Ch	IENCLR	Interrupt enable clear register	Section 5.3.6
01E1 5100h	FXD_MPSAR	Fixed range start address register	Section 5.3.7
01E1 5104h	FXD_MPEAR	Fixed range end address register	Section 5.3.8
01E1 5108h	FXD_MPPA	Fixed range memory protection page attributes register	Section 5.3.9
01E1 5200h	PROG1_MPSAR	Programmable range 1 start address register	

Copyright © 2013–2016, Texas Instruments Incorporated

Table 5-7. Memory Protection Unit 2 (MPU2) Registers (continued)

Address	Acronym	Register Description	Section
01E1 5204h	PROG1_MPEAR	Programmable range 1 end address register	
01E1 5208h	PROG1_MPPA	Programmable range 1 memory protection page attributes register	Section 5.3.12
01E1 5210h	PROG2_MPSAR	Programmable range 2 start address register	
01E1 5214h	PROG2_MPEAR	Programmable range 2 end address register	
01E1 5218h	PROG2_MPPA	Programmable range 2 memory protection page attributes register	Section 5.3.12
01E1 5220h	PROG3_MPSAR	Programmable range 3 start address register	
01E1 5224h	PROG3_MPEAR	Programmable range 3 end address register	
01E1 5228h	PROG3_MPPA	Programmable range 3 memory protection page attributes register	Section 5.3.12
01E1 5230h	PROG4_MPSAR	Programmable range 4 start address register	
01E1 5234h	PROG4_MPEAR	Programmable range 4 end address register	
01E1 5238h	PROG4_MPPA	Programmable range 4 memory protection page attributes register	Section 5.3.12
01E1 5240h	PROG5_MPSAR	Programmable range 5 start address register	
01E1 5244h	PROG5_MPEAR	Programmable range 5 end address register	
01E1 5248h	PROG5_MPPA	Programmable range 5 memory protection page attributes register	Section 5.3.12
01E1 5250h	PROG6_MPSAR	Programmable range 6 start address register	
01E1 5254h	PROG6_MPEAR	Programmable range 6 end address register	
01E1 5258h	PROG6_MPPA	Programmable range 6 memory protection page attributes register	Section 5.3.12
01E1 5260h	PROG7_MPSAR	Programmable range 7 start address register	
01E1 5274h	PROG7_MPEAR	Programmable range 7 end address register	
01E1 5268h	PROG7_MPPA	Programmable range 7 memory protection page attributes register	Section 5.3.12
01E1 5270h	PROG8_MPSAR	Programmable range 8 start address register	
01E1 5274h	PROG8_MPEAR	Programmable range 8 end address register	
01E1 5278h	PROG8_MPPA	Programmable range 8 memory protection page attributes register	Section 5.3.12
01E1 5280h	PROG9_MPSAR	Programmable range 9 start address register	
01E1 5284h	PROG9_MPEAR	Programmable range 9 end address register	
01E1 5288h	PROG9_MPPA	Programmable range 9 memory protection page attributes register	Section 5.3.12
01E1 5290h	PROG10_MPSAR	Programmable range 10 start address register	
01E1 5294h	PROG10_MPEAR	Programmable range 10 end address register	
01E1 5298h	PROG10_MPPA	Programmable range 10 memory protection page attributes register	Section 5.3.12
01E1 52A0h	PROG11_MPSAR	Programmable range 11 start address register	
01E1 52A4h	PROG11_MPEAR	Programmable range 11 end address register	
01E1 52A8h	PROG11_MPPA	Programmable range 11 memory protection page attributes register	Section 5.3.12
01E1 52B0h	PROG12_MPSAR	Programmable range 12 start address register	
01E1 52B4h	PROG12_MPEAR	Programmable range 12 end address register	
01E1 52B8h	PROG12_MPPA	Programmable range 12 memory protection page attributes register	Section 5.3.12
01E1 5300h	FLTADDRR	Fault address register	Section 5.3.13
01E1 5304h	FLTSTAT	Fault status register	Section 5.3.14
01E1 5308h	FLTCLR	Fault clear register	Section 5.3.15

Copyright © 2013–2016, Texas Instruments Incorporated

5.3.1 Revision Identification Register (REVID)

The revision ID register (REVID) contains the MPU revision. The REVID is shown in Figure 5-3 and described in Table 5-8.

Figure 5-3. Revision ID Register (REVID)

31	0
REV	
R-4E81 0101h	

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

Table 5-8. Revision ID Register (REVID) Field Descriptions

Bit	Field	Value	Description
31-0	REV	4E81 0101h	Revision ID of the MPU.

5.3.2 Configuration Register (CONFIG)

The configuration register (CONFIG) contains the configuration value of the MPU. The CONFIG is shown in Figure 5-4 and described in Table 5-9.

NOTE: Although the NUM_AIDS bit defaults to 12 (Ch), not all AIDs may be supported on your device. Unsupported AIDs should be cleared to 0 in the memory page protection attributes registers (MPPA). See for a list of AIDs supported on your device.

Figure 5-4. Configuration Register (CONFIG)

31	24	23	20	19	16
ADDR_WIDTH		NUM_FIXED		NUM_PROG	
R-0[1] or 6h[2]		R-0[1] or 1[2]		R-6h[1] or Ch[2]	

15	12	11	1	0
NUM_AIDS		Reserved		ASSUME_ALLOWED
R-Ch		R-0		R-1

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

(1) For MPU1.
(2) For MPU2.

Table 5-9. Configuration Register (CONFIG) Field Descriptions

Bit	Field	Value	Description
31-24	ADDR_WIDTH	0-FFh	Address alignment (2^n KByte alignment) for range checking.
23-20	NUM_FIXED	0-Fh	Number of fixed address ranges.
19-16	NUM_PROG	0-Fh	Number of programmable address ranges.
15-12	NUM_AIDS	0-Fh	Number of supported AIDs.
11-1	Reserved	0	Reserved
0	ASSUME_ALLOWED		Assume allowed. When an address is not covered by any MPU protection range, this bit determines whether the transfer is assumed to be allowed or not allowed.
		0	Assume is disallowed.
		1	Assume is allowed.

Copyright © 2013–2016, Texas Instruments Incorporated

5.3.3 Interrupt Raw Status/Set Register (IRAWSTAT)

Reading the interrupt raw status/set register (IRAWSTAT) returns the status of all interrupts. Software can write to IRAWSTAT to manually set an interrupt; however, an interrupt is generated only if the interrupt is enabled in the interrupt enable set register (IENSET). Writes of 0 have no effect. The IRAWSTAT is shown in Figure 5-5 and described in Table 5-10.

Figure 5-5. Interrupt Raw Status/Set Register (IRAWSTAT)

31	16
Reserved	
R-0	

15	2	1	0
		ADDRERR	PROTERR
R-0		R/W-0	R/W-0

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

Table 5-10. Interrupt Raw Status/Set Register (IRAWSTAT) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	ADDRERR		Address violation error. Reading this bit reflects the status of the interrupt. Writing 1 sets the status; writing 0 has no effect.
		0	Interrupt is not set.
		1	Interrupt is set.
0	PROTERR		Protection violation error. Reading this bit reflects the status of the interrupt. Writing 1 sets the status; writing 0 has no effect.
		0	Interrupt is not set.
		1	Interrupt is set.

Copyright © 2013–2016, Texas Instruments Incorporated

5.3.4 Interrupt Enable Status/Clear Register (IENSTAT)

Reading the interrupt enable status/clear register (IENSTAT) returns the status of only those interrupts that are enabled in the interrupt enable set register (IENSET). Software can write to IENSTAT to clear an interrupt; the interrupt is cleared from both IENSTAT and the interrupt raw status/set register (IRAWSTAT). Writes of 0 have no effect. The IENSTAT is shown in Figure 5-6 and described in Table 5-11.

Figure 5-6. Interrupt Enable Status/Clear Register (IENSTAT)

31	16
Reserved	
R-0	

15	2	1	0
		ADDRERR	PROTERR
R-0		R/W-0	R/W-0

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

Table 5-11. Interrupt Enable Status/Clear Register (IENSTAT) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	ADDRERR		Address violation error. If the interrupt is enabled, reading this bit reflects the status of the interrupt. If the interrupt is disabled, reading this bit returns 0. Writing 1 sets the status; writing 0 has no effect.
		0	Interrupt is not set.
		1	Interrupt is set.
0	PROTERR		Protection violation error. If the interrupt is enabled, reading this bit reflects the status of the interrupt. If the interrupt is disabled, reading this bit returns 0. Writing 1 sets the status; writing 0 has no effect.
		0	Interrupt is not set.
		1	Interrupt is set.

Copyright © 2013–2016, Texas Instruments Incorporated

5.3.5 Interrupt Enable Set Register (IENSET)

Reading the interrupt enable set register (IENSET) returns the interrupts that are enabled. Software can write to IENSET to enable an interrupt. Writes of 0 have no effect. The IENSET is shown in Figure 5-7 and described in Table 5-12.

Figure 5-7. Interrupt Enable Set Register (IENSET)

31	16
Reserved	
R-0	

15	2	1	0
		ADDRERR_EN	PROTERR_EN
R-0		R/W-0	R/W-0

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

Table 5-12. Interrupt Enable Set Register (IENSET) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	ADDRERR_EN		Address violation error enable.
		0	Writing 0 has no effect.
		1	Interrupt is enabled.
0	PROTERR_EN		Protection violation error enable.
		0	Writing 0 has no effect.
		1	Interrupt is enabled.

5.3.6 Interrupt Enable Clear Register (IENCLR)

Reading the interrupt enable clear register (IENCLR) returns the interrupts that are enabled. Software can write to IENCLR to clear/disable an interrupt. Writes of 0 have no effect. The IENCLR is shown in Figure 5-8 and described in Table 5-13.

Figure 5-8. Interrupt Enable Clear Register (IENCLR)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		ADDRERR_CLR	PROTERR_CLR
R-0		R/W-0	R/W-0

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

Table 5-13. Interrupt Enable Clear Register (IENCLR) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	ADDRERR_CLR		Address violation error disable.
		0	Writing 0 has no effect.
		1	Interrupt is cleared/disabled.
0	PROTERR_CLR		Protection violation error disable.
		0	Writing 0 has no effect.
		1	Interrupt is cleared/disabled.

Copyright © 2013–2016, Texas Instruments Incorporated

5.3.7 Fixed Range Start Address Register (FXD_MPSAR)

The fixed range start address register (FXD_MPSAR) holds the start address for the fixed range. The fixed address range manages access to the DDR2/mDDR SDRAM control registers (B000 0000h–B000 7FFFh). However, these addresses are *not* indicated in FXD_MPSAR and the fixed range end address register (FXD_MPEAR), which instead read as 0. The FXD_MPSAR is shown in Figure 5-9.

Figure 5-9. Fixed Range Start Address Register (FXD_MPSAR)

31	0
Reserved	
R-0	

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

5.3.8 Fixed Range End Address Register (FXD_MPEAR)

The fixed range end address register (FXD_MPEAR) holds the end address for the fixed range. The fixed address range manages access to the DDR2/mDDR SDRAM control registers (B000 0000h–B000 7FFFh). However, these addresses are *not* indicated in FXD_MPEAR and the fixed range start address register (FXD_MPSAR), which instead read as 0. The FXD_MPEAR is shown in Figure 5-10.

Figure 5-10. Fixed Range End Address Register (FXD_MPEAR)

31	0
Reserved	
R-0	

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

Copyright © 2013–2016, Texas Instruments Incorporated

5.3.9 Fixed Range Memory Protection Page Attributes Register (FXD_MPPA)

The fixed range memory protection page attributes register (FXD_MPPA) holds the permissions for the fixed region. This register is writeable by a supervisor entity only. The FXD_MPPA is shown in Figure 5-11 and described in Table 5-14.

Figure 5-11. Fixed Range Memory Protection Page Attributes Register (FXD_MPPA)

31										26	25						22	21	20	19	18	17	16
Reserved											Reserved							AID11	AID10	AID9	AID8	AID7	AID6
R-0											R-Fh							R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								
AID5	AID4	AID3	AID2	AID1	AID0	AIDX	Rsvd	Rsvd	Rsvd	SR	SW	SX	UR	UW	UX								
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R-0	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 5-14. Fixed Range Memory Protection Page Attributes Register (FXD_MPPA) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	Reserved
25-22	Reserved	Fh	Reserved
21-10	AID*n*		Controls access from ID = *n*.
		0	Access is denied.
		1	Access is granted.
9	AIDX		Controls access from ID > 11.
		0	Access is denied.
		1	Access is granted.
8	Reserved	0	Reserved
7	Reserved	1	Reserved. This bit must be written as 1.
6	Reserved	1	Reserved. This bit must be written as 1.
5	SR		Supervisor Read permission.
		0	Access is denied.
		1	Access is allowed.
4	SW		Supervisor Write permission.
		0	Access is denied.
		1	Access is allowed.
3	SX		Supervisor Execute permission.
		0	Access is denied.
		1	Access is allowed.
2	UR		User Read permission.
		0	Access is denied.
		1	Access is allowed.
1	UW		User Write permission.
		0	Access is denied.
		1	Access is allowed.
0	UX		User Execute permission.
		0	Access is denied.
		1	Access is allowed.

Copyright © 2013–2016, Texas Instruments Incorporated

5.3.10 Programmable Range *n* Start Address Registers (PROG*n*_MPSAR)

NOTE: In some cases the amount of physical memory in actual use may be less than the maximum amount of memory supported by the device. For example, the device may support a total of 512 Mbytes of SDRAM memory, but your design may only populate 128 Mbytes. In such cases, the unpopulated memory range must be protected in order to prevent unintended/disallowed aliased access to protected memory, especially memory. One of the programmable address ranges could be used to detect accesses to this unpopulated memory.

The programmable range *n* start address register (PROG*n*_MPSAR) holds the start address for the range *n*. The PROG*n*_MPSAR is writeable by a supervisor entity only.

The start address must be aligned on a page boundary. The size of the page depends on the MPU: the page size for MPU1 is 1 KBbyte; the page size for MPU2 is 64 KBytes. The size of the page determines the width of the address field in PROG*n*_MPSAR and the programmable range *n* end address register (PROG*n*_MPEAR). For example, to protect a 64-KB page starting at byte address 8001 0000h, write 8001 0000h to PROG*n*_MPSAR and 8001 FFFFh to PROG*n*_MPEAR.

5.3.10.1 MPU1 Programmable Range *n* Start Address Register (PROG1_MPSAR-PROG6_MPSAR)

The PROG*n*_MPSAR for MPU1 is shown in Figure 5-12 and described in Table 5-15.

Figure 5-12. MPU1 Programmable Range *n* Start Address Register (PROG*n*_MPSAR)

31	10	9	0
START_ADDR		Reserved	
R/W-20 0000h		R-0	

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

Table 5-15. MPU1 Programmable Range *n* Start Address Register (PROG*n*_MPSAR) Field Descriptions

Bit	Field	Value	Description
31-10	START_ADDR	20 0000h–20 007Fh	Start address for range N .
9-0	Reserved	0	Reserved

5.3.10.2 MPU2 Programmable Range *n* Start Address Register (PROG1_MPSAR-PROG12_MPSAR)

The PROG*n*_MPSAR for MPU2 is shown in Figure 5-13 and described in Table 5-16.

Figure 5-13. MPU2 Programmable Range *n* Start Address Register (PROG*n*_MPSAR)

31	16	15	0
START_ADDR		Reserved	
R/W-C000h		R-0	

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

Table 5-16. MPU2 Programmable Range *n* Start Address Register (PROG*n*_MPSAR) Field Descriptions

Bit	Field	Value	Description
31-16	START_ADDR	C000h–DFFFh	Start address for range N.
15-0	Reserved	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

5.3.11 Programmable Range *n* End Address Registers (PROG*n*_MPEAR)

The programmable range *n* end address register (PROG*n*_MPEAR) holds the end address for the range *n*. This register is writeable by a supervisor entity only.

The end address must be aligned on a page boundary. The size of the page depends on the MPU: the page size for MPU1 is 1 KByte; the page size for MPU2 is 64 KBytes. The size of the page determines the width of the address field in the programmable range *n* start address register (PROG*n*_MPSAR) and PROG*n*_MPEAR. For example, to protect a 64-KB page starting at byte address 8001 0000h, write 8001 0000h to PROG*n*_MPSAR and 8001 FFFFh to PROG*n*_MPEAR.

5.3.11.1 MPU1 Programmable Range *n* End Address Register (PROG1_MPEAR-PROG6_MPEAR)

The PROG*n*_MPEAR for MPU1 is shown in Figure 5-14 and described in Table 5-17.

Figure 5-14. MPU1 Programmable Range *n* End Address Register (PROG*n*_MPEAR)

31	10	9	0
END_ADDR		Reserved	
R/W-20 007Fh		R-3FFh	

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

Table 5-17. MPU1 Programmable Range *n* End Address Register (PROG*n*_MPEAR) Field Descriptions

Bit	Field	Value	Description
31-10	END_ADDR	20 0000h–20 007Fh	End address for range N.
9-0	Reserved	3FFh	Reserved

5.3.11.2 MPU2 Programmable Range *n* End Address Register (PROG1_MPEAR-PROG12_MPEAR)

The PROG*n*_MPEAR for MPU2 is shown in Figure 5-15 and described in Table 5-18.

Figure 5-15. MPU2 Programmable Range *n* End Address Register (PROG*n*_MPEAR)

31	16	15	0
END_ADDR		Reserved	
R/W-DFFFh		R-FFFFh	

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

Table 5-18. MPU2 Programmable Range *n* End Address Register (PROG*n*_MPEAR) Field Descriptions

Bit	Field	Value	Description
31-16	END_ADDR	C000h–DFFFh	Start address for range N.
15-0	Reserved	FFFFh	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

5.3.12 Programmable Range *n* Memory Protection Page Attributes Register (PROG*n*_MPPA)

The programmable range *n* memory protection page attributes register (PROG*n*_MPPA) holds the permissions for the region *n*. This register is writeable only by a supervisor entity. The PROG*n*_MPPA is shown in Figure 5-16 and described in Table 5-19.

Figure 5-16. Programmable Range Memory Protection Page Attributes Register (PROG*n*_MPPA)

31						26	25				22	21	20	19	18	17	16
Reserved							Reserved					AID11	AID10	AID9	AID8	AID7	AID6
R-0							R-Fh					R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
AID5	AID4	AID3	AID2	AID1	AID0	AIDX	Rsvd	Rsvd	Rsvd	SR	SW	SX	UR	UW	UX		
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R-0	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 5-19. Programmable Range Memory Protection Page Attributes Register (PROG*n*_MPPA) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	Reserved
25-22	Reserved	Fh	Reserved
21-10	AID*n*		Controls access from ID = *n*.
		0	Access is denied.
		1	Access is granted.
9	AIDX		Controls access from ID > 11.
		0	Access is denied.
		1	Access is granted.
8	Reserved	0	Reserved
7	Reserved	1	Reserved. This bit must be written as 1.
6	Reserved	1	Reserved. This bit must be written as 1.
5	SR		Supervisor Read permission.
		0	Access is denied.
		1	Access is allowed.
4	SW		Supervisor Write permission.
		0	Access is denied.
		1	Access is allowed.
3	SX		Supervisor Execute permission.
		0	Access is denied.
		1	Access is allowed.
2	UR		User Read permission.
		0	Access is denied.
		1	Access is allowed.
1	UW		User Write permission.
		0	Access is denied.
		1	Access is allowed.
0	UX		User Execute permission.
		0	Access is denied.
		1	Access is allowed.

Copyright © 2013–2016, Texas Instruments Incorporated

5.3.13 Fault Address Register (FLTADDRR)

The fault address register (FLTADDRR) holds the address of the first protection fault transfer. The FLTADDRR is shown in Figure 5-17 and described in Table 5-20.

Figure 5-17. Fault Address Register (FLTADDRR)

31	0
FLTADDR	
R-0	

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

Table 5-20. Fault Address Register (FLTADDRR) Field Descriptions

Bit	Field	Value	Description
31-0	FLTADDR	0-FFFF FFFFh	Memory address of fault.

Copyright © 2013–2016, Texas Instruments Incorporated

5.3.14 Fault Status Register (FLTSTAT)

The fault status register (FLTSTAT) holds the status and attributes of the first protection fault transfer. The FLTSTAT is shown in Figure 5-18 and described in Table 5-21.

Figure 5-18. Fault Status Register (FLTSTAT)

31	24	23	16
Reserved		MSTID	
R-0		R-0	

15	13	12	9	8	6	5	0
Reserved		PRIVID		Reserved		TYPE	
R-0		R-0		R-0		R-0	

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

Table 5-21. Fault Status Register (FLTSTAT) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reserved
23-16	MSTID	0-FFh	Master ID of fault transfer.
15-13	Reserved	0	Reserved
12-9	PRIVID	0-Fh	Privilege ID of fault transfer.
8-6	Reserved	0	Reserved
5-0	TYPE	0-3Fh	Fault type. The TYPE bit field is cleared when a 1 is written to the CLEAR bit in the fault clear register (FLTCLR).
		0	No fault.
		1h	User execute fault.
		2h	User write fault.
		3h	Reserved
		4h	User read fault.
		5h-7h	Reserved
		8h	Supervisor execute fault.
		9h-Fh	Reserved
		10h	Supervisor write fault.
		11h	Reserved
		12h	Relaxed cache write back fault.
		13h-1Fh	Reserved
		20h	Supervisor read fault.
		21h-3Eh	Reserved
		3Fh	Relaxed cache line fill fault.

Copyright © 2013–2016, Texas Instruments Incorporated

5.3.15 Fault Clear Register (FLTCLR)

The fault clear register (FLTCLR) allows software to clear the current fault so that another can be captured in the fault status register (FLTSTAT) as well as produce an interrupt. Only the TYPE bit field in FLTSTAT is cleared when a 1 is written to the CLEAR bit. The FLTCLR is shown in Figure 5-19 and described in Table 5-22.

Figure 5-19. Fault Clear Register (FLTCLR)

31	16
Reserved	
R-0	

15	1	0
Reserved		CLEAR
R-0		W-0

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

Table 5-22. Fault Clear Register (FLTCLR) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	CLEAR		Command to clear the current fault. Writing 0 has no effect.
		0	No effect.
		1	Clear the current fault.

Copyright © 2013–2016, Texas Instruments Incorporated

Device Clocking

Topic **Page**

6.1 Overview **109**
6.2 Frequency Flexibility **111**
6.3 Peripheral Clocking **112**

Copyright © 2013–2016, Texas Instruments Incorporated

6.1 Overview

This device requires two primary reference clocks:

- One reference clock is required for the phase-locked loop controllers (PLLCs)
- One reference clock is required for the real-time clock (RTC) module.

These reference clocks may be sourced from either the on-board oscillator via an externally supplied crystal or by a direct external oscillator input. For detailed specifications on clock frequency and voltage requirements, see the electrical specifications in your device-specific data manual.

In addition to the reference clocks required for the PLLCs and RTC module, some peripherals, such as the USB, may also require an input reference clock to be supplied. All possible input clocks are described in Table 6-1. The CPU and the majority of the device peripherals operate at fixed ratios of the primary system/ARM clock frequency, as listed in Table 6-2. However, there are two system clock domains that do not require a fixed ratio to the ARM, these are PLL0_SYSCLK3 and PLL0_SYSCLK7. Figure 6-1 shows the clocking architecture.

Table 6-1. Device Clock Inputs

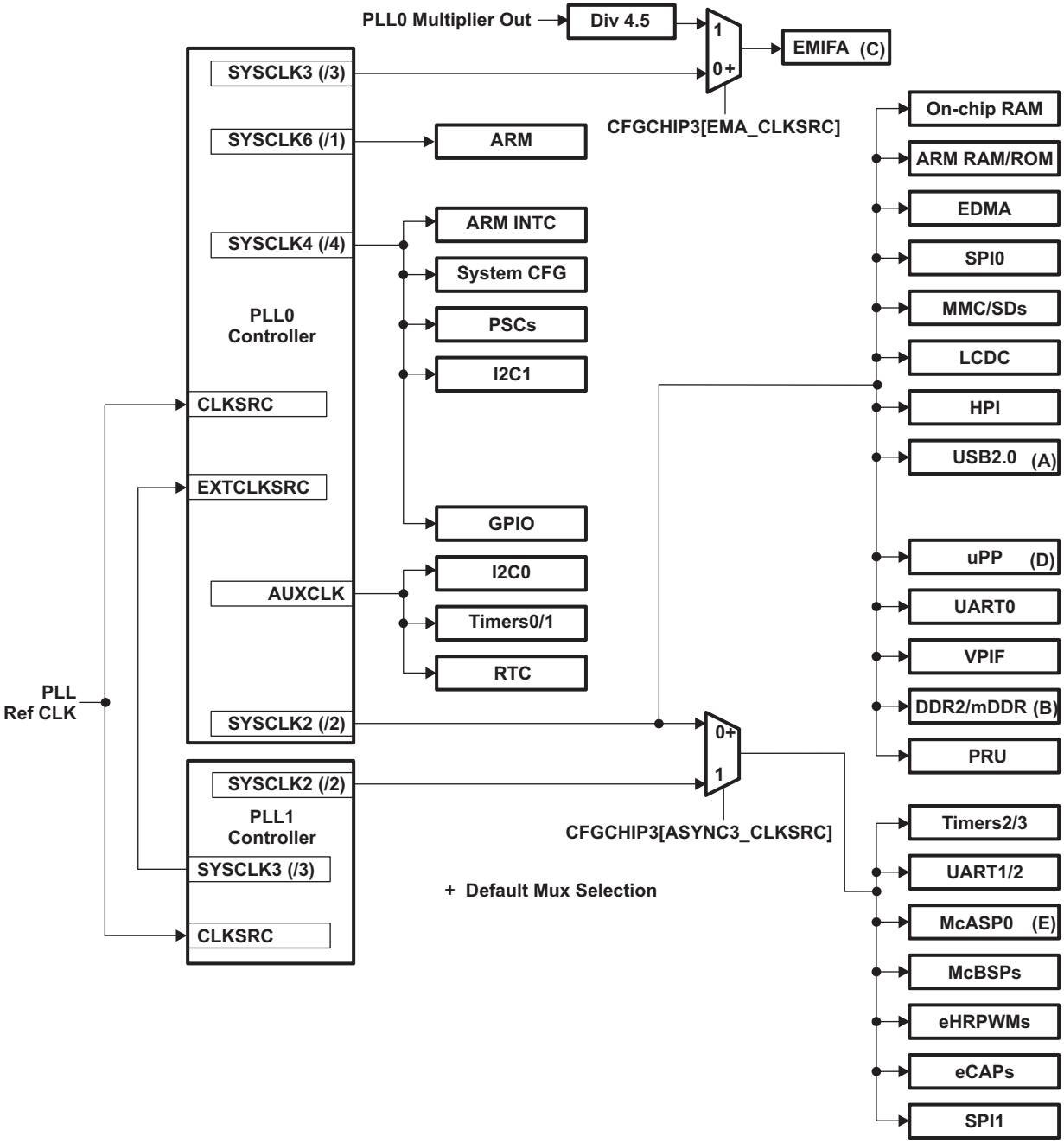
Peripheral	Input Clock Signal Name
Oscillator/PLL	OSCIN
RTC	RTC_XI
JTAG	TCK, RTCK
USB2.0	USB_REFCLKIN
I2Cs	I2Cn_SCL
Timers	TM64Pn_IN12
SPIs	SPIn_CLK
uPP	UPP_CHn_CLK
VPIF	VPIF_CLKINn
McBSPs	CLKSn, CLKRn, CLKXn
McASP0	ACLKR, AHCLKR, ACLKX, AHCLKX

Table 6-2. System Clock Domains

CPU/Device Peripherals	System Clock Domain	Fixed Ratio to ARM Clock Required?	Default Ratio to ARM Clock
ARM RAM/ROM, On-chip RAM, UART0, EDMA, SPI0, MMC/SDs, VPIF, LCDC, uPP, DDR2/mDDR (bus ports), USB2.0, HPI, PRU subsystem	PLL0_SYSCLK2	Yes	1:2
EMIFA	PLL0_SYSCLK3	No	1:3
System configuration (SYSCFG), GPIO, PLLCs, PSCs, I2C1, ARM INTC	PLL0_SYSCLK4	Yes	1:4
ARM	PLL0_SYSCLK6	Yes	1:1
I2C0, Timer64P0/P1, RTC, McASP0 serial clock	PLL0_AUXCLK	Not Applicable	Not Applicable
DDR2/mDDR PHY	PLL1_SYSCLK1	Not Applicable	Not Applicable
PLL0 input reference clock (not configured by default)	PLL1_SYSCLK3	Not Applicable	Not Applicable
ECAPs, UART1/2, Timer64P2/3, eHRPWMs, McBSPs, McASP0, SPI1	ASYNC3	Not Applicable	Not Applicable

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 6-1. Overall Clocking Diagram



A See Section 6.3.1 for USB clocking.

B See Section 6.3.2 for DDR2/mDDR clocking.

C See Section 6.3.3 for EMIFA clocking.

D See Section 6.3.4 for uPP clocking.

E See Section 6.3.5 for McASP clocking.

Copyright © 2013–2016, Texas Instruments Incorporated

6.2 Frequency Flexibility

There are two PLLs on the device with similar architecture and behavior. Each PLL has two clocking modes:

- PLL Bypass
- PLL Active

When the PLL is in Bypass mode, the reference clock supplied on OSCIN serves as the clock source from which all of the system clocks (SYSCLK1 to SYSCLK7) are derived. This means that when the PLL is in Bypass mode, the reference clock supplied on OSCIN passes directly to the system of PLLDIV blocks that creates each of the system clocks. For PLL0 only, the EXTCLKSRC bit in PLLCTL can be configured to use PLL1_SYSCLK3 as the Bypass mode reference clock.

When the PLL operates in Active mode, the PLL is enabled and the PLL multiplier setting is used to multiply the input clock frequency supplied on the OSCIN pin up to the desired frequency. It is this multiplied frequency that all system clocks are derived from in PLL Active mode.

The output of the PLL multiplier passes through a post divider (POSTDIV) block and then is applied to the system of PLLDIV blocks that creates each of the system clock domains (SYSCLK1 to SYSCLK7). Each SYSCLK*n* has a PLLDIV*n* block associated with it. See the *Phase-Locked Loop Controller (PLLC)* chapter for more details on the PLL.

The combination of the PLL multiplier, POSTDIV, and PLLDIV blocks provides flexibility in the frequencies that the system clock domains support. This flexibility does have limitations, as follows:

- OSCIN input frequency is limited to a supported range.
- The output of the PLL Multiplier must be within the range specified in the device-specific data manual.
- The output of each PLLDIV block must be less than or equal to the maximum device frequency specified in the device-specific data manual.

NOTE: The above limitations are provided here as an example and are used to illustrate the recommended configuration of the PLL controller. These limitations may vary based on core voltage and between devices. See the device-specific data manual for more details.

Table 6-3 shows examples of possible PLL multiplier settings, along with the available PLL post-divider modes. The PLL post-divider modes are defined by the value programmed in the RATIO field of the PLL post-divider control register (POSTDIV). For Div1, Div2, Div3, and Div4 modes, the RATIO field would be programmed to 0, 1, 2, and 3, respectively. The Div1, Div2, Div3, and Div4 modes are shown here as an example. Additional post-divider modes are supported and are documented in the *Phase-Locked Loop Controller (PLLC)* chapter.

NOTE: PLL power consumption increases as the output frequency of the PLL multiplier increases. To decrease PLL power consumption, the lowest PLL multiplier (PLLM) setting should be chosen that achieves the desired frequency. For example, if 200 MHz is the desired CPU operating frequency and the OSCIN frequency is 25 MHz; lower power consumption is achieved by choosing a PLLM setting of ×16 and a post-divider (POSTDIV) setting of /2 instead of a PLLM setting of ×24 and a POSTDIV setting of /3, even though both of these modes would result in a CPU frequency of 200 MHz.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 6-3. Example PLL Frequencies

OSCIN Frequency	PLL Multiplier	Multiplier Frequency	Div1	Div2	Div3	Div4
20	30	600 MHz	600	300	200	150
24	25	600 MHz	600	300	200	150
25	24	600 MHz	600	300	200	150
30	20	600 MHz	600	300	200	150
20	25	500 MHz	500	250	167	125
24	20	480 MHz	480	240	160	120
25	18	450 MHz	450	225	150	112.5
30	14	420 MHz	420	210	140	105
25	16	400 MHz	400	200	133	100

6.3 Peripheral Clocking

6.3.1 USB Clocking

Figure 6-2 shows the clock connections for the USB2.0 module. Note that there is no built-in oscillator. The USB2.0 subsystem requires a reference clock for its internal PLL. This reference clock can be sourced from either the USB_REFCLKIN pin or from the AUXCLK of the system PLL. The reference clock input to the USB2.0 subsystem is selected by programming the USB0PHYCLKMUX bit in the chip configuration 2 register (CFGCHIP2) of the System Configuration Module. The USB_REFCLKIN source should be selected when it is not possible (such as when specific audio rates are required) to operate the device at one of the allowed input frequencies to the USB2.0 subsystem. The USB2.0 subsystem peripheral bus clock is sourced from PLL0_SYSCLK2. Table 6-4 determines the source origination as well as the source input frequency to the USB 2.0 PHY. Once the clock source origination (internal/external) and its frequency is determined, the firmware should program the PHY PLL with the correct input frequency via CFGCHIP2.USB0REF_FREQ.

Figure 6-2. USB Clocking Diagram

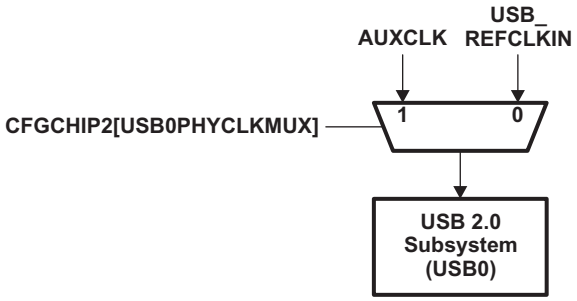


Table 6-4. USB Clock Multiplexing Options

CFGCHIP2. USB0PHYCLKMUX bit	USB2.0 Clock Source	Additional Conditions
0	USB_REFCLKIN	USB_REFCLKIN must be 12, 24, 48, 19.2, 38.4, 13, 26, 20, or 40 MHz. The PLL inside the USB2.0 PHY can be configured to accept any of these input clock frequencies.
1	PLL0_AUXCLK	PLL0_AUXCLK must be 12, 24, 48, 19.2, 38.4, 13, 26, 20, or 40 MHz. The PLL inside the USB2.0 PHY can be configured to accept any of these input clock frequencies.

Copyright © 2013–2016, Texas Instruments Incorporated

6.3.2 DDR2/mDDR Memory Controller Clocking

The DDR2/mDDR memory controller requires two input clocks to source VCLK and 2X_CLK (see Figure 6-3):

- VCLK is sourced from PLL0_SYSCLK2/2 that clocks the command FIFO, write FIFO, and read FIFO of the DDR2/mDDR memory controller. From this, VCLK drives the interface to the peripheral bus.
- 2X_CLK is sourced from PLL1_SYSCLK1.

2X_CLK clock is again divided down by 2 in the DDR PHY controller to generate a clock called MCLK. The MCLK domain consists of the DDR2/mDDR memory controller state machine and memory-mapped registers. This clock domain is clocked at the rate of the external DDR2/mDDR memory, 2X_CLK/2.

Table 6-5 shows example PLL register settings based on the OSCIN reference clock frequency of 25 MHz. From these example configurations, the following observations are made:

- To achieve the maximum frequency (150 MHz) supported by the DDR2/mDDR memory controller and the typical CPU frequency of 300 MHz, the output of the PLL multiplier should be set to be 300 MHz and the DDR_CLK source should be set to PLL1_SYSCLK1.
- The frequency of the PLL1 direct output clock is fixed at the output frequency of the PLL1 multiplier block.
- The PLLDIV1 block that sets the divider ratio for SYSCLK1 can be changed to achieve various clock frequencies.
- For certain PLL1 multiplier and PLL1 post-divider control register (POSTDIV) settings, a higher clock frequency can be achieved by selecting SYSCLK1 as the clock source for 2X_CLK.

If the DDR2/mDDR memory controller is not in use and the DDR_CLK and $\overline{\text{DDR_CLK}}$ are used in the application as a free running clock that could be used by an FPGA or for some other purpose, then 2X_CLK should be used as the source for DDR_CLK and $\overline{\text{DDR_CLK}}$ and VCLK should be gated off. This allows clock gating of the majority of the logic in the DDR2/mDDR memory controller via the LPSC while still providing a clock on the DDR_CLK and $\overline{\text{DDR_CLK}}$.

NOTE: DDR_CLK and $\overline{\text{DDR_CLK}}$ are output clock signals.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 6-3. DDR2/mDDR Memory Controller Clocking Diagram

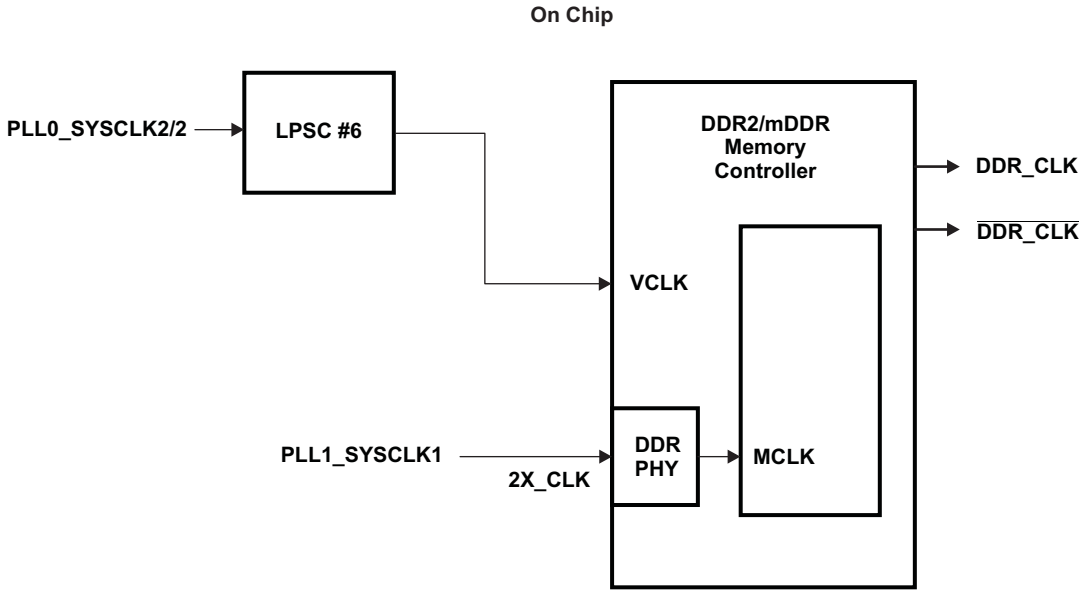


Table 6-5. DDR2/mDDR Memory Controller MCLK Frequencies

OSCIN Frequency	PLL1 Multiplier Register Setting	PLL1 Multiplier Frequency	PLL1 Post Divider Mode[1]	PLL1 POSTDIV Output Frequency	PLL1 PLLDIV1 Register Setting	PLL1_SYSCLK1	MCLK
24	18h	600 MHz	Div2	300 MHz	8000h	300 MHz	150 MHz
24	15h	528 MHz	Div2	264 MHz	8000h	264 MHz	132 MHz
24	14h	504 MHz	Div2	252 MHz	8000h	252 MHz	126 MHz

(1) See Section 6.2 for explanation of POSTDIV divider modes.

Copyright © 2013–2016, Texas Instruments Incorporated

6.3.3 EMIFA Clocking

EMIFA requires a single input clock source. The EMIFA clock can be sourced from either PLL0_SYSCLK3 or DIV4P5 (see Figure 6-4). The EMA_CLKSRC bit in the chip configuration 3 register (CFGCHIP3) of the System Configuration Module controls whether PLL0_SYSCLK3 or DIV4P5 is selected as the clock source for EMIFA.

Selecting the appropriate clock source for EMIFA is determined by the desired clock rate. Table 6-6 shows example PLL register settings and the resulting DIV4P5 and PLL0_SYSCLK3 frequencies based on the OSCIN reference clock frequency of 25 MHz. From these example configurations, the following observations can be made:

- To achieve a typical frequency of 100 MHz supported by EMIFA and the typical CPU frequency of 300 MHz, the output of the PLL multiplier should be set to 600 MHz and the EMA_CLK source should be set to PLL0_SYSCLK3 with the PLLDIV3 register set to 3.
- The frequency of the DIV4P5 clock is fixed at the output frequency of the PLL multiplier block divided by 4.5.
- The PLLDIV3 block that sets the divider ratio for PLL0_SYSCLK3 can be changed to achieve various clock frequencies.

Figure 6-4. EMIFA Clocking Diagram

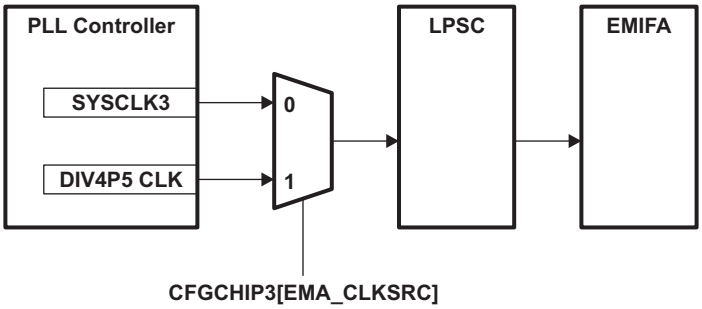


Table 6-6. EMIFA Frequencies

OSCIN Frequency	PLL Multiplier Register Setting	Multiplier Frequency	Post Divider Mode[1]	POSTDIV Output Frequency	DIV4P5	PLLDIV3 Register Setting	PLL0_SYSCLK3
25	24	600 MHz	Div2	300 MHz	133 MHz	2	100 MHz
			Div3	200 MHz	133 MHz	2	66.6 MHz
						1	100 MHz
			Div4	150 MHz	133 MHz	1	75 MHz
25	18	450 MHz	Div2	225 MHz	100 MHz	3	56.3 MHz
						2	75 MHz
			Div3	150 MHz	100 MHz	1	75 MHz
			Div4	112.5 MHz	100 MHz	1	56.3 MHz
						0	112.5 MHz
25	16	400 MHz	Div2	200 MHz	89 MHz	2	66.6 MHz
						1	100 MHz
			Div3	133 MHz	89 MHz	1	66.5 MHz
			Div4	100 MHz	89 MHz	0	100 MHz

(1) See Section 6.2 for explanation of POSTDIV divider modes.

Copyright © 2013–2016, Texas Instruments Incorporated

6.3.4 uPP Clocking

Figure 6-5 displays the clock connections for the uPP module. The uPP subsystem requires a module clock to drive its internal logic and a transmit clock to drive I/O signals in transmit mode. The module clock is always sourced by PLL0_SYSCLK2. The transmit clock is sourced by three different clocks: PLL0_SYSCLK2 (default), PLL1_SYSCLK2, or the externally driven UPP_2xTXCLK pin. The transmit clock source is selected by the UPP_TX_CLKSRC and ASYNC3_CLKSRC bits in the chip configuration 3 register (CFGCHIP3) of the System Configuration Module. Table 6-7 lists the register values that select each of the three possible clock sources.

Regardless of the source, the uPP transmit clock speed cannot exceed the uPP module clock speed. The module clock speed must be greater than or equal to the transmit clock speed.

Figure 6-5. uPP Clocking Diagram

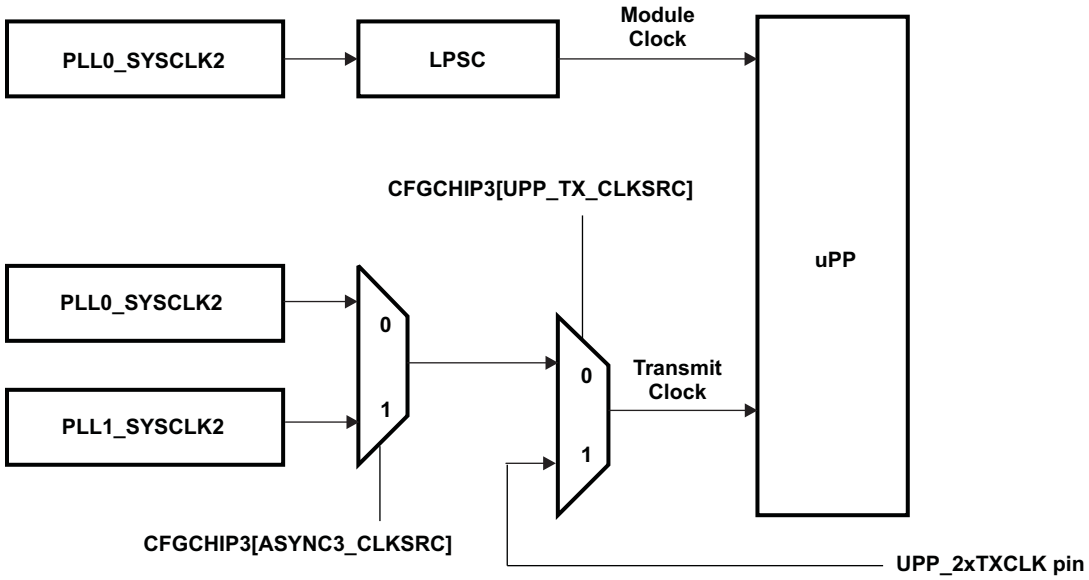


Table 6-7. uPP Transmit Clock Selection

CFGCHIP3.UPP_TX_CLKSRC bit	CFGCHIP3.ASYNC3_CLKSRC bit	uPP Transmit Clock Source
0	0	PLL0_SYSCLK2
0	1	PLL1_SYSCLK2
1	x	UPP_2xTXCLK pin

Copyright © 2013–2016, Texas Instruments Incorporated

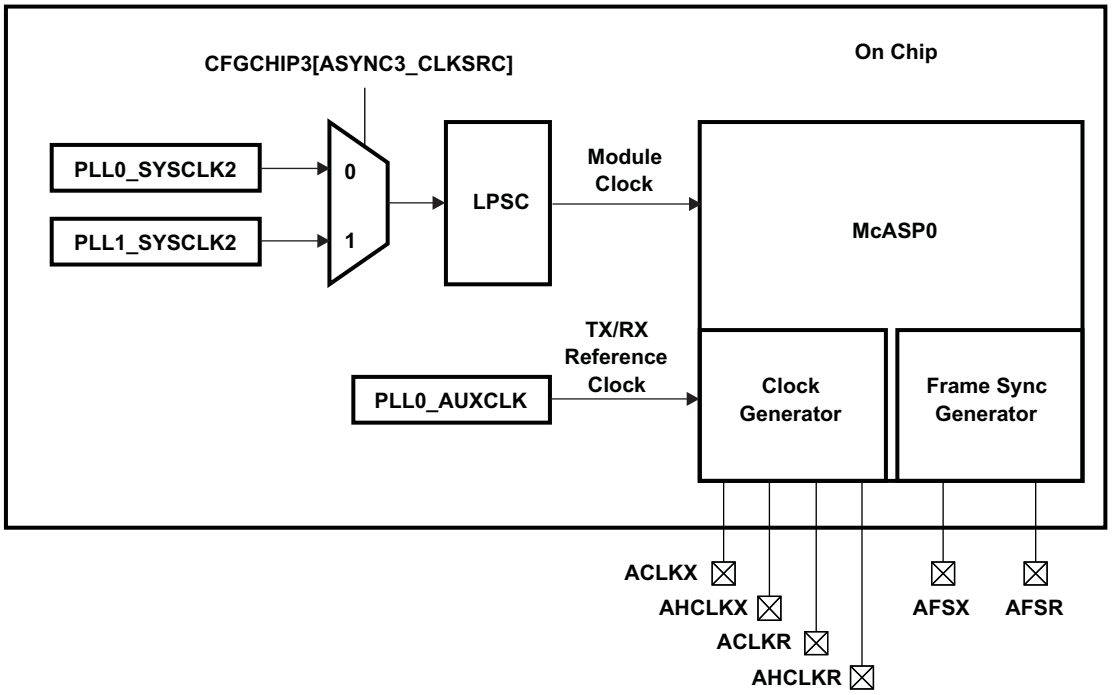
6.3.5 McASP Clocking

As shown in Figure 6-6, the McASP peripheral requires multiple clock sources. Internally, the module clock is selected to be either PLL0_SYSCLK2 or PLL1_SYSCLK2 by configuring the ASYNC3_CLKSRC bit in the chip configuration 3 register (CFGCHIP3) of the System Configuration Module.

The transmit and receive clocks are sourced internally or externally by configuring the McASP clock control registers ACLKRCTL, AHCLKRCTL, ACLKXCTL, and AHCLKXCTL. If an external clock is driven into a high-frequency master clock (AHCLKX or AHCLKR), the McASP module allows for a mixed clock mode where the associated lower frequency clock (ACLKX or ACLKR) can be derived from the high-frequency master clock through a programmable divider.

When the internal clock source option is selected, the transmit and receive clocks are derived from the PLL0_AUXCLK clock through programmable dividers.

Figure 6-6. McASP Clocking Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

6.3.6 I/O Domains

The I/O domains refer to the frequencies of the peripherals that communicate through device pins. In many cases, there are frequency requirements for a peripheral pin interface that are set by an outside standard and must be met. It is not necessarily possible to obtain these frequencies from the on-chip clock generation circuitry, so the frequencies must be obtained from external sources and are asynchronous to the CPU frequency by definition.

The peripherals can be divided into the following groups, depending upon their clock requirements, as shown in Table 6-8.

Table 6-8. Peripherals

Peripheral Group	Peripheral Group Definition	Peripherals Contained within Group	Source of Peripheral Clock
RTC	Operates off of a dedicated 32 kHz crystal oscillator.	RTC	—
Fixed-Frequency Peripherals	As the name suggests, fixed-frequency peripherals have a fixed-frequency. They are fed the AUXCLK directly from the oscillator input.	Timer64P0/P1	—
		I2C0	—
Synchronous Peripherals	Synchronous peripherals have their frequencies derived from the ARM clock frequency. The peripheral system clock frequency changes accordingly, if the PLL0 frequency changes. Most synchronous peripherals have internal dividers so they can generate their required clock frequencies.	MMC/SDs	PLL0_SYSCLK2
		HPI	PLL0_SYSCLK2
		UART0	PLL0_SYSCLK2
		LCDC	PLL0_SYSCLK2
		GPIO	PLL0_SYSCLK4
Asynchronous Peripherals	Asynchronous peripherals are not required to operate at a fixed ratio of the ARM clock.	eCAPs	ASYNC3
		eHRPWMs	ASYNC3
		UART1/2	ASYNC3
		Timer64P2/P3	ASYNC3
		EMIFA	DIV_4P5 or PLL0_SYSCLK3
		DDR2/mDDR	PLL1_SYSCLK1 or PLL1 Direct Output
Synchronous/Asynchronous Peripherals	Synchronous/asynchronous peripherals can be run with either internally generated synchronous clocks, or externally generated asynchronous clocks.	McASP0	ASYNC3 or Peripheral Serial Clock
		McBSPs	ASYNC3 or Peripheral Serial Clock
		SPI0	PLL0_SYSCLK2 or Peripheral Serial Clock
		SPI1	ASYNC3 or Peripheral Serial Clock
		I2C1	PLL0_SYSCLK4 or Peripheral Serial Clock
		uPP	PLL0_SYSCLK2 or Peripheral Serial Clock
		VPIF	PLL0_SYSCLK2 or Peripheral Serial Clock
		USB2.0	USB_REFCLKIN or AUXCLK

Copyright © 2013–2016, Texas Instruments Incorporated

Phase-Locked Loop Controller (PLLC)

Topic **Page**

7.1 Introduction **120**
7.2 PLL Controllers **120**
7.3 PLLC Registers **125**

Copyright © 2013–2016, Texas Instruments Incorporated

7.1 Introduction

This device has two phase-locked loop (PLL) controllers, PLLC0 and PLLC1. These PLL controllers provide clock signals to most of the components of the device through various clock dividers.

Both PLL0 and PLL1 provide the following:

- Glitch-free transitions when clock settings are changed
- Domain clock alignment
- Clock gating
- PLL power-down

The clock outputs generated by the PLL controllers are:

- Domain clocks: PLL0_SYSCLK[1-7] and PLL1_SYSCLK[1-3]
- Auxiliary clock (PLL0_AUXCLK) from the PLLC0 reference clock source

Dividers that can be used for the PLL controllers are:

- Pre-PLL divider: PREDIV
- Post-PLL divider: POSTDIV
- SYSCLK divider: D1, …, Dn

Various other control signals supported are:

- PLL multiplier: PLLM
- Software-programmable PLL bypass: PLLEN

7.2 PLL Controllers

PLL0 and PLL1 share the same internal architecture so they also share the same approach for mode configuration.

PLL0 provides the primary system clock to the device. PLL0 operations are software programmable through the PLL controller 0 (PLLC0) registers.

PLL1 provides the reference clocks to various peripherals (including DDR2/mDDR) and may generate clocks that are asynchronous to the PLL0 clocks. PLL1 operations are software programmable through the PLL controller 1 (PLLC1) registers.

Figure 7-1 shows the PLLC0 and PLLC1 architecture.

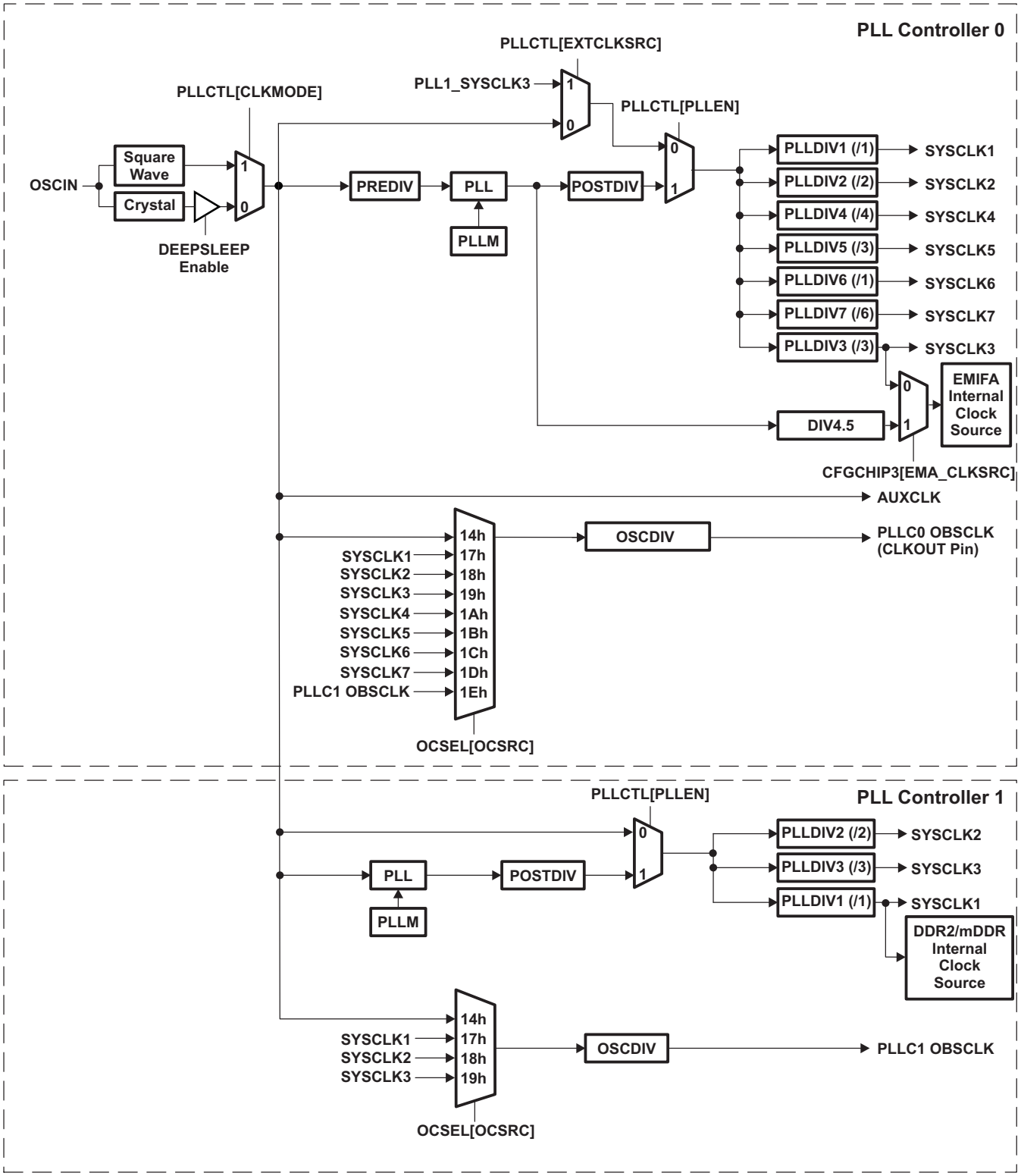
The PLL0 and PLL1 multipliers are controlled by their respective PLL multiplier control register (PLLM). The PLLM defaults to a multiplier value of 13h at power-up, which results in a PLL multiplier of 20×. The PLL0 and PLL1 output clocks may be divided-down for slower device operation using the PLL post-divider control register (POSTDIV). The POSTDIV has a default value of /2, but may be modified through software (using the RATIO field in POSTDIV) to achieve lower device operation frequencies. The default PLLM and POSTDIV settings produce a 300-MHz PLL output clock when given a 30-MHz clock source.

At power-up, PLL0 and PLL1 are powered-down/disabled and must be powered-up by software through the PLLPWRDN bit in their respective PLL control register (PLLCTL). Before each PLL completes the power-up and frequency-lock sequence, the system operates in bypass mode by default and the system clock (OSCIN) is provided directly from an input reference clock (square wave or internal oscillator) selected by the CLKMODE bit in PLLCTL. After the power-up and frequency-lock sequences are complete, software can switch the device to PLL mode operation (set the PLLEN bit in PLLCTL to 1).

The PLL controller registers are listed in Section 7.3.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 7-1. PLLC Structure



Copyright © 2013–2016, Texas Instruments Incorporated

7.2.1 Device Clock Generation

The PLL controllers (PLLC0 and PLLC1) manage the clock ratios, alignment, and gating for the device system clocks. Various PLL mode attributes such as pre-division, multiplier, and post-division are software programmable through the PLL controller registers. Additionally, the reset controller in PLLC0 manages reset propagation through the device, clock alignment, and test points.

The PLLOUT stage in PLLC0 and PLLC1 is capable of providing frequencies greater than what the SYSCLK dividers can handle. The POSTDIV stage should be programmed to keep the input to the SYSCLK dividers within operating limits. See the device datasheet for the maximum operating frequencies.

PLLC0 and PLLC1 generate several clocks for use by the various processors and modules. These reference clocks are summarized in Table 7-1. Some output clock dividers require fixed values so that clock ratios between various device components are maintained regardless of PLL or bypass frequency.

Table 7-1. System PLLC Output Clocks

Output Clock	Used by	Default Ratio (relative to PLL*n*_SYSCLK1)	Fixed Clock Ratio
	PLLC0[1]		
PLL0_SYSCLK1	Not used	/1	Yes
PLL0_SYSCLK2	ARM RAM/ROM, On-chip RAM, UART0, EDMA, SPI0, MMC/SDs, VPIF, LCDC, uPP, DDR2/mDDR (bus ports), USB2.0, HPI, PRU	/2	Yes
PLL0_SYSCLK3[2]	EMIFA	/3	No
PLL0_SYSCLK4	System configuration (SYSCFG), GPIO, PLLCs, PSCs, I2C1, ARM INTC	/4	Yes
PLL0_SYSCLK5	Not used	/3	No
PLL0_SYSCLK6	ARM	/1	Yes
PLL0_AUXCLK	I2C0, Timer64P0/P1, RTC, USB2.0 PHY, McASP0 serial clock	PLL bypass clock	No
PLL0_OBSCLK	Observation clock (OBSCLK) source	Pin configurable	No
	PLLC1		
PLL1_SYSCLK1	DDR2/mDDR PHY	/1 or disabled	No
PLL1_SYSCLK2[3]	ECAPs, UART1/2, Timer64P2/3, eHRPWMs, McBSPs, McASP0, SPI1 (all these modules use PLL0_SYSCLK2 by default)	/2 or disabled	No
PLL1_SYSCLK3[4]	PLL0 input reference clock (not configured by default)	/3 or disabled	No

(1) The divide values in PLLC0 for PLL0_SYSCLK1/PLL0_SYSCLK6, PLL0_SYSCLK2, and PLL0_SYSCLK4 can be changed for power savings, but the device must maintain the 1:2:4 clock ratios between the clock domains.

(2) PLLC0 supports an additional post-divider value of /4.5 that can be used for EMIFA clock generation. When this /4.5 value is used, the resulting clock will not have a 50% duty cycle. Instead, the duty cycle will be 44.4%. The EMIFA uses PLL0_SYSCLK3 by default, but can be configured to use a /4.5 divide-down of PLL0_PLLOUT instead of PLL0_SYSCLK3 by programming the EMA_CLKSRC and DIV45PENA bits in the chip configuration 3 register (CFGCHIP3) of the system configuration (SYSCFG) module.

(3) The ASYNC3 modules use PLL0_SYSCLK2 by default, but all these modules can be configured as a group to use PLL1_SYSCLK2 by programming the ASYNC3_CLKSRC bit in the chip configuration 3 register (CFGCHIP3) of the system configuration (SYSCFG) module.

(4) The PLL0 input clock source can be configured to use PLL1_SYSCLK3 instead of OSCIN by programming the EXTCLKSRC bit in the PLLC0 PLL control register (PLLCTL). The PLL1 input clock source will also be OSCIN.

Copyright © 2013–2016, Texas Instruments Incorporated

7.2.2 Steps for Programming the PLLs

Note that there is a lock mechanism implemented to protect the PLL controller registers. See Section 7.2.2.1 for information on unlocking the PLL controller registers.

Refer to the appropriate subsection on how to program the PLL clocks:

- If the PLL is powered down (PLLPWRDN bit in PLLCTL is set to 1), follow the full PLL initialization procedure in Section 7.2.2.2.
- If the PLL is not powered down (PLLPWRDN bit in PLLCTL is cleared to 0), follow the sequence in Section 7.2.2.3 to change the PLL multiplier.
- If the PLL is already running at a desired multiplier and only the SYSCLK dividers will be updated, follow the sequence in Section 7.2.2.4.

Note that the PLLs are powered down after a Power-on Reset (POR). The PLLs are not powered down after a Warm Reset (RESET), but the PLLEN bit in PLLCTL is cleared to 0 (bypass mode) and the PLLDIVx registers are reset to default values.

7.2.2.1 Locking/Unlocking PLL Register Access

A lock mechanism is implemented on the device to prevent inadvertent writes to the PLL controller registers. This provides protection from stopping modules when the module clocks are disabled. For example, the watchdog timer that runs on the PLL0_AUXCLK will stop if this PLL clock is unintentionally disabled.

The PLL lock bits are located within the system configuration (SYSCFG) module:

- When set, the PLL_MASTER_LOCK bit in the chip configuration 0 register (CFGCHIP0) locks PLLC0.
- When set, the PLL1_MASTER_LOCK bit in the chip configuration 3 register (CFGCHIP3) locks PLLC1.

Because the SYSCFG module has its own lock mechanism, the SYSCFG module must be unlocked first by writing to the KICK0R and KICK1R registers before the PLL lock bits can be cleared. Like the KICK registers, the PLL lock bits can only be modified while in a privileged mode. See the *System Configuration (SYSCFG) Module* chapter for information on privilege type and the KICK0R and KICK1R registers.

NOTE: The PLL_MASTER_LOCK bit in CFGCHIP0 and the PLL1_MASTER_LOCK bit in CFGCHIP3 default to unlocked after reset, so the following procedure is only required if the PLLs have been locked (set to 1).

To modify the PLL controller registers, use the following sequence:

1. Write the correct key values to KICK0R and KICK1R registers.
2. Clear the PLL_MASTER_LOCK bit in CFGCHIP0 and/or the PLL1_MASTER_LOCK bit in CFGCHIP3, as required.
3. Configure the desired PLL controller register values.
4. Set the PLL_MASTER_LOCK bit in CFGCHIP0 and/or the PLL1_MASTER_LOCK bit in CFGCHIP3, as required.
5. Write an incorrect key value to the KICK0R and KICK1R registers.

Copyright © 2013–2016, Texas Instruments Incorporated

7.2.2.2 Initializing PLL Mode from PLL Power Down

If the PLL is powered down (PLLPWRDN bit in PLLCTL is set to 1), perform the following procedure to initialize the PLL:

1. Program the CLKMODE bit in PLLC0 PLLCTL.
2. Switch the PLL to bypass mode:
   (a) Clear the PLLENSRC bit in PLLCTL to 0 (allows PLLEN bit to take effect).
   (b) For PLL0 only, select the clock source by programming the EXTCLKSRC bit in PLLCTL.
   (c) Clear the PLLEN bit in PLLCTL to 0 (PLL in bypass mode).
   (d) Wait for 4 OSCIN cycles to ensure that the PLLC has switched to bypass mode.
3. Clear the PLLRST bit in PLLCTL to 0 (resets PLL).
4. Clear the PLLPWRDN bit in PLLCTL to 0 (brings PLL out of power-down mode).
5. Program the desired multiplier value in PLLM. Program the POSTDIV, as needed.
6. If desired, program PLLDIV*n* registers to change the SYSCLK*n* divide values:
   (a) Wait for the GOSTAT bit in PLLSTAT to clear to 0 (indicates that no operation is currently in progress).
   (b) Program the RATIO field in PLLDIV*n*.
   (c) Set the GOSET bit in PLLCMD to 1 (initiates a new divider transition).
   (d) Wait for the GOSTAT bit in PLLSTAT to clear to 0 (completion of divider change).
7. Set the PLLRST bit in PLLCTL to 1 (brings PLL out of reset).
8. Wait for the PLL to lock. See the device-specific data manual for PLL lock time.
9. Set the PLLEN bit in PLLCTL to 1 (removes PLL from bypass mode).

7.2.2.3 Changing PLL Multiplier

If the PLL is not powered down (PLLPWRDN bit in PLLCTL is cleared to 0), perform the following procedure to change the PLL multiplier:

1. Switch the PLL to bypass mode:
   (a) Clear the PLLENSRC bit in PLLCTL to 0 (allows PLLEN bit to take effect).
   (b) For PLL0 only, select the clock source by programming the EXTCLKSRC bit in PLLCTL.
   (c) Clear the PLLEN bit in PLLCTL to 0 (PLL in bypass mode).
   (d) Wait for 4 OSCIN cycles to ensure that the PLLC has switched to bypass mode.
2. Clear the PLLRST bit in PLLCTL to 0 (resets PLL).
3. Program the desired multiplier value in PLLM. Program the POSTDIV, as needed.
4. If desired, program PLLDIV*n* registers to change the SYSCLK*n* divide values:
   (a) Wait for the GOSTAT bit in PLLSTAT to clear to 0 (indicates that no operation is currently in progress).
   (b) Program the RATIO field in PLLDIV*n*.
   (c) Set the GOSET bit in PLLCMD to 1 (initiates a new divider transition).
   (d) Wait for the GOSTAT bit in PLLSTAT to clear to 0 (completion of divider change).
5. Set the PLLRST bit in PLLCTL to 1 (brings PLL out of reset).
6. Wait for the PLL to lock. See the device-specific data manual for PLL lock time.
7. Set the PLLEN bit in PLLCTL to 1 (removes PLL from bypass mode).

Copyright © 2013–2016, Texas Instruments Incorporated

7.2.2.4 Changing SYSCLK Dividers

If the PLL is already operating at the desired multiplier mode, perform the following procedure to change the SYSCLK divider values:

1. Wait for the GOSTAT bit in PLLSTAT to clear to 0 (indicates that no operation is currently in progress).
2. Program the RATIO field in PLLDIV*n*.
3. Set the GOSET bit in PLLCMD to 1 (initiates a new divider transition).
4. Wait for the GOSTAT bit in PLLSTAT to clear to 0 (completion of divider change).

7.3 PLLC Registers

Table 7-2 lists the memory-mapped registers for the PLLC0 and Table 7-3 lists the memory-mapped registers for the PLLC1.

Table 7-2. PLL Controller 0 (PLLC0) Registers

Address	Acronym	Register Description	Section
01C1 1000h	REVID	PLLC0 Revision Identification Register	Section 7.3.1
01C1 10E4h	RSTYPE	PLLC0 Reset Type Status Register	Section 7.3.3
01C1 10E8h	RSCTRL	PLLC0 Reset Control Register	Section 7.3.4
01C1 1100h	PLLCTL	PLLC0 Control Register	Section 7.3.5
01C1 1104h	OCSEL	PLLC0 OBSCLK Select Register	Section 7.3.7
01C1 1110h	PLLM	PLLC0 PLL Multiplier Control Register	Section 7.3.9
01C1 1114h	PREDIV	PLLC0 Pre-Divider Control Register	Section 7.3.10
01C1 1118h	PLLDIV1	PLLC0 Divider 1 Register	Section 7.3.11
01C1 111Ch	PLLDIV2	PLLC0 Divider 2 Register	Section 7.3.13
01C1 1120h	PLLDIV3	PLLC0 Divider 3 Register	Section 7.3.15
01C1 1124h	OSCDIV	PLLC0 Oscillator Divider 1 Register	Section 7.3.21
01C1 1128h	POSTDIV	PLLC0 PLL Post-Divider Control Register	Section 7.3.23
01C1 1138h	PLLCMD	PLLC0 PLL Controller Command Register	Section 7.3.24
01C1 113Ch	PLLSTAT	PLLC0 PLL Controller Status Register	Section 7.3.25
01C1 1140h	ALNCTL	PLLC0 Clock Align Control Register	Section 7.3.26
01C1 1144h	DCHANGE	PLLC0 PLLDIV Ratio Change Status Register	Section 7.3.28
01C1 1148h	CKEN	PLLC0 Clock Enable Control Register	Section 7.3.30
01C1 114Ch	CKSTAT	PLLC0 Clock Status Register	Section 7.3.32
01C1 1150h	SYSTAT	PLLC0 SYSCLK Status Register	Section 7.3.34
01C1 1160h	PLLDIV4	PLLC0 Divider 4 Register	Section 7.3.17
01C1 1164h	PLLDIV5	PLLC0 Divider 5 Register	Section 7.3.18
01C1 1168h	PLLDIV6	PLLC0 Divider 6 Register	Section 7.3.19
01C1 116Ch	PLLDIV7	PLLC0 Divider 7 Register	Section 7.3.20
01C1 11F0h	EMUCNT0	PLLC0 Emulation Performance Counter 0 Register	Section 7.3.36
01C1 11F4h	EMUCNT1	PLLC0 Emulation Performance Counter 1 Register	Section 7.3.37

Copyright © 2013–2016, Texas Instruments Incorporated

Table 7-3. PLL Controller 1 (PLLC1) Registers

Address	Acronym	Register Description	Section
01E1 A000h	REVID	PLLC1 Revision Identification Register	Section 7.3.2
01E1 A100h	PLLCTL	PLLC1 Control Register	Section 7.3.6
01E1 A104h	OCSEL	PLLC1 OBSCLK Select Register	Section 7.3.8
01E1 A110h	PLLM	PLLC1 PLL Multiplier Control Register	Section 7.3.9
01E1 A118h	PLLDIV1	PLLC1 Divider 1 Register	Section 7.3.12
01E1 A11Ch	PLLDIV2	PLLC1 Divider 2 Register	Section 7.3.14
01E1 A120h	PLLDIV3	PLLC1 Divider 3 Register	Section 7.3.16
01E1 A124h	OSCDIV	PLLC1 Oscillator Divider 1 Register	Section 7.3.22
01E1 A128h	POSTDIV	PLLC1 PLL Post-Divider Control Register	Section 7.3.23
01E1 A138h	PLLCMD	PLLC1 PLL Controller Command Register	Section 7.3.24
01E1 A13Ch	PLLSTAT	PLLC1 PLL Controller Status Register	Section 7.3.25
01E1 A140h	ALNCTL	PLLC1 Clock Align Control Register	Section 7.3.27
01E1 A144h	DCHANGE	PLLC1 PLLDIV Ratio Change Status Register	Section 7.3.29
01E1 A148h	CKEN	PLLC1 Clock Enable Control Register	Section 7.3.31
01E1 A14Ch	CKSTAT	PLLC1 Clock Status Register	Section 7.3.33
01E1 A150h	SYSTAT	PLLC1 SYSCLK Status Register	Section 7.3.35
01E1 A1F0h	EMUCNT0	PLLC1 Emulation Performance Counter 0 Register	Section 7.3.36
01E1 A1F4h	EMUCNT1	PLLC1 Emulation Performance Counter 1 Register	Section 7.3.37

7.3.1 PLLC0 Revision Identification Register (REVID)

The PLLC0 revision identification register (REVID) is shown in Figure 7-2 and described in Table 7-4.

Figure 7-2. PLLC0 Revision Identification Register (REVID)

31	0
REV	
R-4481 3C00h	

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

Table 7-4. PLLC0 Revision Identification Register (REVID) Field Descriptions

Bit	Field	Value	Description
31-0	REV	4481 3C00h	Peripheral revision ID for PLLC0.

Copyright © 2013–2016, Texas Instruments Incorporated

7.3.2 PLLC1 Revision Identification Register (REVID)

The PLLC1 revision identification register (REVID) is shown in Figure 7-3 and described in Table 7-5.

Figure 7-3. PLLC1 Revision Identification Register (REVID)

31	0
REV	
R-4481 4400h	

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

Table 7-5. PLLC1 Revision Identification Register (REVID) Field Descriptions

Bit	Field	Value	Description
31-0	REV	4481 4400h	Peripheral revision ID for PLLC1.

7.3.3 Reset Type Status Register (RSTYPE)

The reset type status register (RSTYPE) latches the cause of the last reset. If multiple reset sources are asserted simultaneously, RSTYPE records the reset source that deasserts last. If multiple reset sources are asserted and deasserted simultaneously, RSTYPE latches the highest priority reset source. RSTYPE is shown in Figure 7-4 and described in Table 7-6.

Figure 7-4. Reset Type Status Register (RSTYPE)

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		PLLSWRST	XWRST	POR
R-0		R-0	R-0	R-0

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

Table 7-6. Reset Type Status Register (RSTYPE) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	PLLSWRST		PLL software reset.
		0	PLL soft reset was not the last reset to occur.
		1	PLL soft was the last reset to occur.
1	XWRST		External warm reset.
		0	External warm reset was not the last reset to occur.
		1	External warm reset was the last reset to occur.
0	POR		Power on reset.
		0	Power On Reset (POR) was not the last reset to occur.
		1	Power On Reset (POR) was the last reset to occur.

Copyright © 2013–2016, Texas Instruments Incorporated

7.3.4 PLLC0 Reset Control Register (RSCTRL)

The reset control register (RSCTRL) allows the device to perform a software-initiated reset. Before writing to the SWRST bit, the register must be unlocked by writing the key value of 5A69h to the KEY bit field. The KEY bit field reads back as Ch when the register is unlocked; any other key value is invalid and indicates that the register is locked. Any write to the register following a successful unlock relocks the register. RSCTRL is shown in Figure 7-5 and described in Table 7-7.

Figure 7-5. Reset Control Register (RSCTRL)

31	17	16
Reserved		SWRST
R-0		R/W-1

15	0
KEY	
R/W-3h	

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

Table 7-7. Reset Control Register (RSCTRL) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16	SWRST		PLL software reset. Register must be unlocked before writing to this bit. Writes are possible only when qualified with a valid key.
		0	In software reset
		1	Not in software reset
15-0	KEY	0-FFFFh	RSCTRL unlock key. Key used to enable writes to RSCTRL.
		3h	Register is locked when read value is 3h.
		Ch	Register is unlocked when read value is Ch.
		5A69h	RSCTRL unlock key

Copyright © 2013–2016, Texas Instruments Incorporated

7.3.5 PLLC0 Control Register (PLLCTL)

The PLLC0 control register (PLLCTL) is shown in Figure 7-6 and described in Table 7-8.

Figure 7-6. PLLC0 Control Register (PLLCTL)

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						EXTCLKSRC	CLKMODE
R-0						R/W-0	R/W-0

7	6	5	4	3	2	1	0
Reserved		PLLENSRC	Reserved	PLLRST	Reserved	PLLPWRDN	PLLEN
R-1		R/W-1	R/W-1	R/W-0	R-0	R/W-1	R/W-0

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

Table 7-8. PLLC0 Control Register (PLLCTL) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9	EXTCLKSRC		External clock source selection.
		0	Use OSCIN for the PLL bypass clock.
		1	Use PLL1_SYSCLK3 for the PLL bypass clock.
8	CLKMODE		Reference clock selection.
		0	Internal oscillator (crystal)
		1	Square wave
7-6	Reserved	1	Reserved
5	PLLENSRC	0	This bit must be cleared before the PLLEN bit will have any effect.
4	Reserved	1	Reserved. Write the default value when modifying this register.
3	PLLRST		PLL0 reset.
		0	PLL0 reset is asserted.
		1	PLL0 reset is not asserted.
2	Reserved	0	Reserved
1	PLLPWRDN		PLL0 power-down.
		0	PLL0 is operating.
		1	PLL0 is powered-down.
0	PLLEN		PLL0 mode enables.
		0	PLL0 is in bypass mode.
		1	PLL0 mode is enabled, not bypassed.

Copyright © 2013–2016, Texas Instruments Incorporated

7.3.6 PLLC1 Control Register (PLLCTL)

The PLLC1 control register (PLLCTL) is shown in Figure 7-7 and described in Table 7-9.

Figure 7-7. PLLC1 Control Register (PLLCTL)

31	16
Reserved	
R-0	

15	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
Reserved		PLLENSRC	Reserved	PLLRST	Reserved	PLLPWRDN	PLLEN
R-1		R/W-1	R/W-1	R/W-0	R-0	R/W-1	R/W-0

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

Table 7-9. PLLC1 Control Register (PLLCTL) Field Descriptions

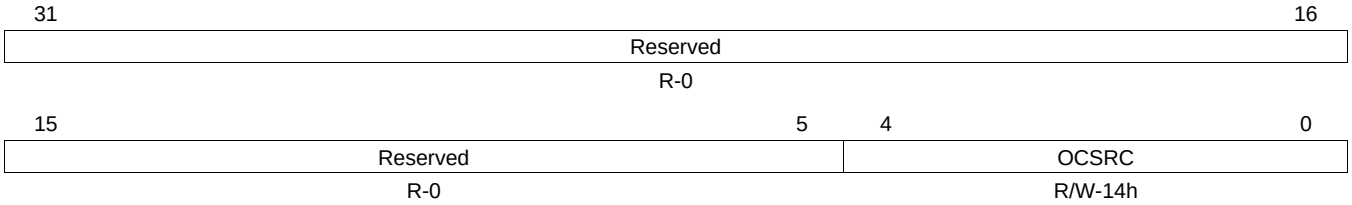
Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-6	Reserved	1	Reserved
5	PLLENSRC	0	This bit must be cleared before the PLLEN bit will have any effect.
4	Reserved	1	Reserved. Write the default value when modifying this register.
3	PLLRST		PLL1 reset.
		0	PLL1 reset is asserted.
		1	PLL1 reset is not asserted.
2	Reserved	0	Reserved
1	PLLPWRDN		PLL1 power-down.
		0	PLL1 is operating.
		1	PLL1 is powered-down.
0	PLLEN		PLL1 mode enables.
		0	PLL1 is in bypass mode.
		1	PLL1 mode is enabled, not bypassed.

Copyright © 2013–2016, Texas Instruments Incorporated

7.3.7 PLLC0 OBSCLK Select Register (OCSEL)

The PLLC0 OBSCLK select register (OCSEL) controls which clock is output on the CLKOUT pin so that it may be used for test and debug purposes (in addition to its normal function of being a direct input clock divider). The OCSEL is shown in Figure 7-8 and described in Table 7-10.

Figure 7-8. PLLC0 OBSCLK Select Register (OCSEL)



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

Table 7-10. PLLC0 OBSCLK Select Register (OCSEL) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4-0	OCSRC	0-1Fh	PLLC0 OBSCLK source. Output on CLKOUT pin.
		0-13h	Reserved
		14h	OSCIN
		15h-16h	Reserved
		17h	PLL0_SYSCLK1
		18h	PLL0_SYSCLK2
		19h	PLL0_SYSCLK3
		1Ah	PLL0_SYSCLK4
		1Bh	PLL0_SYSCLK5
		1Ch	PLL0_SYSCLK6
		1Dh	PLL0_SYSCLK7
		1Eh	PLLC1 OBSCLK
		1Fh	Disabled

Copyright © 2013–2016, Texas Instruments Incorporated

7.3.8 PLLC1 OBSCLK Select Register (OCSEL)

The PLLC1 OBSCLK select register (OCSEL) controls which clock is output on PLLC1 OBSCLK so that it may be used for test and debug purposes (in addition to its normal function of being a direct input clock divider). The OCSEL is shown in Figure 7-9 and described in Table 7-11.

Figure 7-9. PLLC1 OBSCLK Select Register (OCSEL)

31	16
Reserved	
R-0	

15	5	4	0
Reserved		OCSRC	
R-0		R/W-14h	

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

Table 7-11. PLLC1 OBSCLK Select Register (OCSEL) Field Descriptions

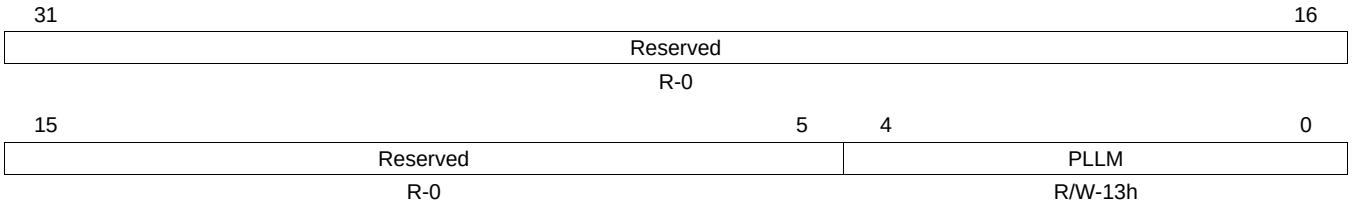
Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4-0	OCSRC	0-1Fh	PLLC1 OBSCLK source.
		0-13h	Reserved
		14h	OSCIN
		15h-16h	Reserved
		17h	PLL1_SYSCLK1
		18h	PLL1_SYSCLK2
		19h	PLL1_SYSCLK3
		1A-1Fh	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

7.3.9 PLL Multiplier Control Register (PLLM)

The PLL multiplier control register (PLLM) is shown in Figure 7-10 and described in Table 7-12.

Figure 7-10. PLL Multiplier Control Register (PLLM)



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

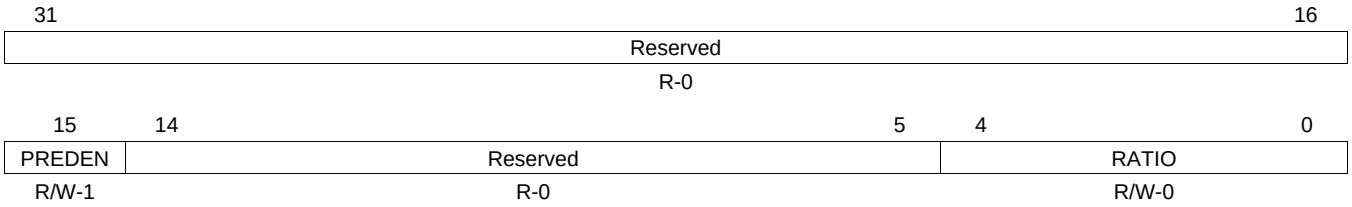
Table 7-12. PLL Multiplier Control Register (PLLM) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4-0	PLLM	0-1Fh	PLL multiplier select. Multiplier Value = PLLM + 1. The valid range of multiplier values for a given OSCIN is defined by the minimum and maximum frequency limits on the PLL VCO frequency. See the device-specific data manual for PLL VCO frequency specification limits.

7.3.10 PLLC0 Pre-Divider Control Register (PREDIV)

The PLLC0 pre-divider control register (PREDIV) is shown in Figure 7-11 and described in Table 7-13.

Figure 7-11. PLLC0 Pre-Divider Control Register (PREDIV)



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

Table 7-13. PLLC0 Pre-Divider Control Register (PREDIV) Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Reserved
15	PREDEN		PLLC0 pre-divider enable.
		0	PLLC0 pre-divider is disabled. Clock output from the PREDIV stage is disabled.
		1	PLLC0 pre-divider is enabled.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider Value = RATIO + 1. RATIO defaults to 0 (PLL pre-divide by 1).

Copyright © 2013–2016, Texas Instruments Incorporated

7.3.11 PLLC0 Divider 1 Register (PLLDIV1)

The PLLC0 divider 1 register (PLLDIV1) controls the divider for PLL0_SYSCLK1. PLLDIV1 is shown in Figure 7-12 and described in Table 7-14.

Figure 7-12. PLLC0 Divider 1 Register (PLLDIV1)

31	16
Reserved	
R-0	

15	14	5	4	0
D1EN	Reserved		RATIO	
R/W-1	R-0		R/W-0	

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

Table 7-14. PLLC0 Divider 1 Register (PLLDIV1) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D1EN		Divider 1 enable.
		0	Divider 1 is disabled.
		1	Divider 1 is enabled.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider Value = RATIO + 1. RATIO defaults to 0 (PLL divide by 1).

7.3.12 PLLC1 Divider 1 Register (PLLDIV1)

The PLLC1 divider 1 register (PLLDIV1) controls the divider for PLL1_SYSCLK1. PLLDIV1 is shown in Figure 7-13 and described in Table 7-15.

Figure 7-13. PLLC1 Divider 1 Register (PLLDIV1)

31	16
Reserved	
R-0	

15	14	5	4	0
D1EN	Reserved		RATIO	
R/W-0	R-0		R/W-0	

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

Table 7-15. PLLC1 Divider 1 Register (PLLDIV1) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D1EN		Divider 1 enable.
		0	Divider 1 is disabled.
		1	Divider 1 is enabled.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider Value = RATIO + 1. RATIO defaults to 0 (PLL divide by 1).

Copyright © 2013–2016, Texas Instruments Incorporated

7.3.13 PLLC0 Divider 2 Register (PLLDIV2)

The PLLC0 divider 2 register (PLLDIV2) controls the divider for PLL0_SYSCLK2. PLLDIV2 is shown in Figure 7-14 and described in Table 7-16.

Figure 7-14. PLLC0 Divider 2 Register (PLLDIV2)

31	16
Reserved	
R-0	

15	14	5	4	0
D2EN	Reserved		RATIO	
R/W-1	R-0		R/W-1	

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

Table 7-16. PLLC0 Divider 2 Register (PLLDIV2) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D2EN		Divider 2 enable.
		0	Divider 2 is disabled.
		1	Divider 2 is enabled.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider Value = RATIO + 1. RATIO defaults to 1 (PLL divide by 2).

7.3.14 PLLC1 Divider 2 Register (PLLDIV2)

The PLLC1 divider 2 register (PLLDIV2) controls the divider for PLL1_SYSCLK2. PLLDIV2 is shown in Figure 7-15 and described in Table 7-17.

Figure 7-15. PLLC1 Divider 2 Register (PLLDIV2)

31	16
Reserved	
R-0	

15	14	5	4	0
D2EN	Reserved		RATIO	
R/W-0	R-0		R/W-1	

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

Table 7-17. PLLC1 Divider 2 Register (PLLDIV2) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D2EN		Divider 2 enable.
		0	Divider 2 is disabled.
		1	Divider 2 is enabled.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider Value = RATIO + 1. RATIO defaults to 1 (PLL divide by 2).

Copyright © 2013–2016, Texas Instruments Incorporated

7.3.15 PLLC0 Divider 3 Register (PLLDIV3)

The PLLC0 divider 3 register (PLLDIV3) controls the divider for PLL0_SYSCLK3. PLLDIV3 is shown in Figure 7-16 and described in Table 7-18.

Figure 7-16. PLLC0 Divider 3 Register (PLLDIV3)

31	16
Reserved	
R-0	

15	14	5	4	0
D3EN	Reserved		RATIO	
R/W-1	R-0		R/W-2h	

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

Table 7-18. PLLC0 Divider 3 Register (PLLDIV3) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D3EN		Divider 3 enable.
		0	Divider 3 is disabled.
		1	Divider 3 is enabled.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider Value = RATIO + 1. RATIO defaults to 2h (PLL divide by 3).

7.3.16 PLLC1 Divider 3 Register (PLLDIV3)

The PLLC1 divider 3 register (PLLDIV3) controls the divider for PLL1_SYSCLK3. PLLDIV3 is shown in Figure 7-17 and described in Table 7-19.

Figure 7-17. PLLC1 Divider 3 Register (PLLDIV3)

31	16
Reserved	
R-0	

15	14	5	4	0
D3EN	Reserved		RATIO	
R/W-0	R-0		R/W-2h	

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

Table 7-19. PLLC1 Divider 3 Register (PLLDIV3) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D3EN		Divider 3 enable.
		0	Divider 3 is disabled.
		1	Divider 3 is enabled.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider Value = RATIO + 1. RATIO defaults to 2h (PLL divide by 3).

Copyright © 2013–2016, Texas Instruments Incorporated

7.3.17 PLLC0 Divider 4 Register (PLLDIV4)

The PLLC0 divider 4 register (PLLDIV4) controls the divider for PLL0_SYSCLK4. PLLDIV4 is shown inFigure 7-18 and described in Table 7-20.

Figure 7-18. PLLC0 Divider 4 Register (PLLDIV4)

31	16
Reserved	
R-0	

15	14	5	4	0
D4EN	Reserved		RATIO	
R/W-1	R-0		R/W-3h	

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

Table 7-20. PLLC0 Divider 4 Register (PLLDIV4) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D4EN		Divider 4 enable.
		0	Divider 4 is disabled.
		1	Divider 4 is enabled.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider Value = RATIO + 1. RATIO defaults 3 (PLL divide by 4).

7.3.18 PLLC0 Divider 5 Register (PLLDIV5)

The PLLC0 divider 5 register (PLLDIV5) controls the divider for PLL0_SYSCLK5. PLLDIV5 is shown in Figure 7-19 and described in Table 7-21.

Figure 7-19. PLLC0 Divider 5 Register (PLLDIV5)

31	16
Reserved	
R-0	

15	14	5	4	0
D5EN	Reserved		RATIO	
R/W-1	R-0		R/W-2h	

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

Table 7-21. PLLC0 Divider 5 Register (PLLDIV5) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D5EN		Divider 5 enable.
		0	Divider 5 is disabled.
		1	Divider 5 is enabled.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider Value = RATIO + 1. RATIO defaults 2 (PLL divide by 3).

Copyright © 2013–2016, Texas Instruments Incorporated

7.3.19 PLLC0 Divider 6 Register (PLLDIV6)

The PLLC0 divider 6 register (PLLDIV6) controls the divider for PLL0_SYSCLK6. PLLDIV6 is shown in Figure 7-20 and described in Table 7-22.

Figure 7-20. PLLC0 Divider 6 Register (PLLDIV6)

31	16
Reserved	
R-0	

15	14	5	4	0
D6EN	Reserved		RATIO	
R/W-1	R-0		R/W-0	

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

Table 7-22. PLLC0 Divider 6 Register (PLLDIV6) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D6EN		Divider 6 enable.
		0	Divider 6 is disabled.
		1	Divider 6 is enabled.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider Value = RATIO + 1. RATIO defaults to 0 (PLL divide by 1).

7.3.20 PLLC0 Divider 7 Register (PLLDIV7)

The PLLC0 divider 7 register (PLLDIV7) controls the divider for PLL0_SYSCLK7. PLLDIV7 is shown in Figure 7-21 and described in Table 7-23.

Figure 7-21. PLLC0 Divider 7 Register (PLLDIV7)

31	16
Reserved	
R-0	

15	14	5	4	0
D7EN	Reserved		RATIO	
R/W-1	R-0		R/W-5h	

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

Table 7-23. PLLC0 Divider 7 Register (PLLDIV7) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	D7EN		Divider 7 enable.
		0	Divider 7 is disabled.
		1	Divider 7 is enabled.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider Value = RATIO + 1. RATIO defaults to 5 (PLL divide by 6).

Copyright © 2013–2016, Texas Instruments Incorporated

7.3.21 PLLC0 Oscillator Divider 1 Register (OSCDIV)

The PLLC0 oscillator divider 1 register (OSCDIV) controls the divider for PLLC0 OBSCLK, dividing down the clock selected as the PLLC0 OBSCLK source. The PLLC0 OBSCLK is connected to the CLKOUT pin. The OSCDIV is shown in Figure 7-22 and described in Table 7-24.

Figure 7-22. PLLC0 Oscillator Divider 1 Register (OSCDIV)

31	16
Reserved	
R-0	

15	14	5	4	0
OD1EN	Reserved		RATIO	
R/W-1	R-0		R/W-0	

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

Table 7-24. PLLC0 Oscillator Divider 1 Register (OSCDIV) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	OD1EN		Oscillator divider 1 enable.
		0	Oscillator divider 1 is disabled.
		1	Oscillator divider 1 is enabled. For PLLC0 OBSCLK to toggle, both the OD1EN bit and the OBSEN bit in the PLLC0 clock enable control register (CKEN) must be set to 1.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider value = RATIO + 1. For example, RATIO = 0 means divide by 1.

7.3.22 PLLC1 Oscillator Divider 1 Register (OSCDIV)

The PLLC1 oscillator divider 1 register (OSCDIV) controls the divider for PLLC1 OBSCLK, dividing down the clock selected as the PLLC1 OBSCLK source. The PLLC1 OBSCLK signal may be selected as the output on the CLKOUT pin. The OSCDIV is shown in Figure 7-23 and described in Table 7-25.

Figure 7-23. PLLC1 Oscillator Divider 1 Register (OSCDIV)

31	16
Reserved	
R-0	

15	14	5	4	0
OD1EN	Reserved		RATIO	
R/W-1	R-0		R/W-0	

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

Table 7-25. PLLC1 Oscillator Divider 1 Register (OSCDIV) Field Descriptions

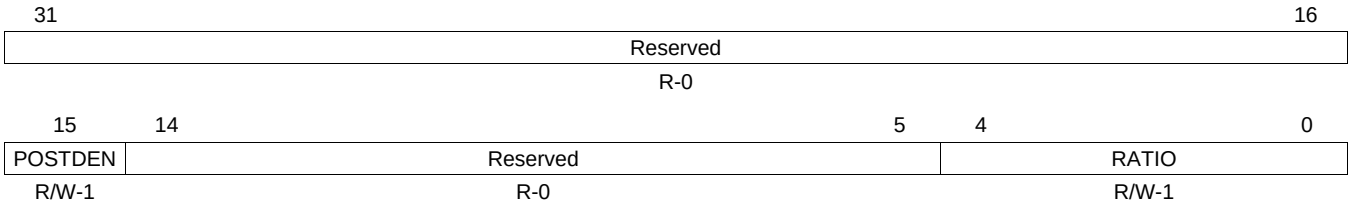
Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	OD1EN		Oscillator divider 1 enable.
		0	Oscillator divider 1 is disabled.
		1	Oscillator divider 1 is enabled. For PLLC1 OBSCLK to toggle, both the OD1EN bit and the OBSEN bit in the PLLC1 clock enable control register (CKEN) must be set to 1.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider value = RATIO + 1. For example, RATIO = 0 means divide by 1.

Copyright © 2013–2016, Texas Instruments Incorporated

7.3.23 PLL Post-Divider Control Register (POSTDIV)

The PLL post-divider control register (POSTDIV) is shown in Figure 7-24 and described in Table 7-26.

Figure 7-24. PLL Post-Divider Control Register (POSTDIV)



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

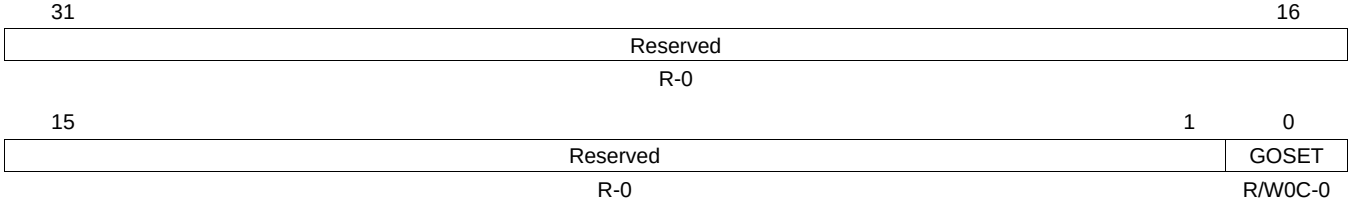
Table 7-26. PLL Post-Divider Control Register (POSTDIV) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	POSTDEN		Post-divider enable.
		0	Post-divider is disabled.
		1	Post-divider is enabled.
14-5	Reserved	0	Reserved
4-0	RATIO	0-1Fh	Divider ratio. Divider Value = RATIO + 1. RATIO defaults to 1 (PLL post-divide by 2).

7.3.24 PLL Controller Command Register (PLLCMD)

The PLL controller command register (PLLCMD) contains the command bit for phase alignment. A write of 1 initiates the command; a write of 0 clears the bit, but has no effect. PLLCMD is shown in Figure 7-25 and described in Table 7-27.

Figure 7-25. PLL Controller Command Register (PLLCMD)



LEGEND: R/W = Read/Write; R = Read only; W0C = Write 0 to clear bit; -*n* = value after reset

Table 7-27. PLL Controller Command Register (PLLCMD) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	GOSET		GO bit for phase alignment.
		0	Clear bit (no effect)
		1	Phase alignment

Copyright © 2013–2016, Texas Instruments Incorporated

7.3.25 PLL Controller Status Register (PLLSTAT)

The PLL controller status register (PLLSTAT) is shown in Figure 7-26 and described in Table 7-28.

Figure 7-26. PLL Controller Status Register (PLLSTAT)

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		STABLE	Reserved	GOSTAT
R-0		R-0	R-0	R-0

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

Table 7-28. PLL Controller Status Register (PLLSTAT) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	STABLE		OSC counter done, oscillator assumed to be stable. By the time the device comes out of reset, this bit should become 1.
		0	No
		1	Yes
1	Reserved	0	Reserved
0	GOSTAT		Status of GO operation. If 1, indicates GO operation is in progress.
		0	GO operation is not in progress.
		1	GO operation is in progress.

Copyright © 2013–2016, Texas Instruments Incorporated

7.3.26 PLLC0 Clock Align Control Register (ALNCTL)

The PLLC0 clock align control register (ALNCTL) indicates which PLL0_SYSCLK*n* needs to be aligned for proper device operation. ALNCTL is shown in Figure 7-27 and described in Table 7-29.

Figure 7-27. PLLC0 Clock Align Control Register (ALNCTL)

31	16
Reserved	
R-0	

15	7	6	5	4	3	2	1	0
Reserved		ALN7	ALN6	ALN5	ALN4	ALN3	ALN2	ALN1
R-3h		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 7-29. PLLC0 Clock Align Control Register (ALNCTL) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	3h	Reserved
6	ALN7		PLL0_SYSCLK7 needs to be aligned to others selected in this register.
		0	No
		1	Yes
5	ALN6		PLL0_SYSCLK6 needs to be aligned to others selected in this register.
		0	No
		1	Yes
4	ALN5		PLL0_SYSCLK5 needs to be aligned to others selected in this register.
		0	No
		1	Yes
3	ALN4		PLL0_SYSCLK4 needs to be aligned to others selected in this register.
		0	No
		1	Yes
2	ALN3		PLL0_SYSCLK3 needs to be aligned to others selected in this register.
		0	No
		1	Yes
1	ALN2		PLL0_SYSCLK2 needs to be aligned to others selected in this register.
		0	No
		1	Yes
0	ALN1		PLL0_SYSCLK1 needs to be aligned to others selected in this register.
		0	No
		1	Yes

Copyright © 2013–2016, Texas Instruments Incorporated

7.3.27 PLLC1 Clock Align Control Register (ALNCTL)

The PLLC1 clock align control register (ALNCTL) indicates which PLL1_SYSCLK*n* needs to be aligned for proper device operation. ALNCTL is shown in Figure 7-28 and described in Table 7-30.

Figure 7-28. PLLC1 Clock Align Control Register (ALNCTL)

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		ALN3	ALN2	ALN1
R-0		R/W-1	R/W-1	R/W-1

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

Table 7-30. PLLC1 Clock Align Control Register (ALNCTL) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	ALN3		PLL1_SYSCLK3 needs to be aligned to others selected in this register.
		0	No
		1	Yes
1	ALN2		PLL1_SYSCLK2 needs to be aligned to others selected in this register.
		0	No
		1	Yes
0	ALN1		PLL1_SYSCLK1 needs to be aligned to others selected in this register.
		0	No
		1	Yes

Copyright © 2013–2016, Texas Instruments Incorporated

7.3.28 PLLC0 PLLDIV Ratio Change Status Register (DCHANGE)

The PLLC0 PLLDIV ratio change status register (DCHANGE) indicates if the PLL0_SYSCLK*n* divide ratio has been modified. DCHANGE is shown in Figure 7-29 and described in Table 7-31.

Figure 7-29. PLLC0 PLLDIV Ratio Change Status Register (DCHANGE)

31	16
Reserved	
R-0	

15	7	6	5	4	3	2	1	0
Reserved		SYS7	SYS6	SYS5	SYS4	SYS3	SYS2	SYS1
R-0		R-0	R-0	R-0	R-0	R-0	R-0	R-0

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

Table 7-31. PLLC0 PLLDIV Ratio Change Status Register (DCHANGE) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reserved
6	SYS7		PLL0_SYSCLK7 divide ratio is modified.
		0	Ratio is not modified.
		1	Ratio is modified.
5	SYS6		PLL0_SYSCLK6 divide ratio is modified.
		0	Ratio is not modified.
		1	Ratio is modified.
4	SYS5		PLL0_SYSCLK5 divide ratio is modified.
		0	Ratio is not modified.
		1	Ratio is modified.
3	SYS4		PLL0_SYSCLK4 divide ratio is modified.
		0	Ratio is not modified.
		1	Ratio is modified.
2	SYS3		PLL0_SYSCLK3 divide ratio is modified.
		0	Ratio is not modified.
		1	Ratio is modified.
1	SYS2		PLL0_SYSCLK2 divide ratio is modified.
		0	Ratio is not modified.
		1	Ratio is modified.
0	SYS1		PLL0_SYSCLK1 divide ratio is modified.
		0	Ratio is not modified.
		1	Ratio is modified.

Copyright © 2013–2016, Texas Instruments Incorporated

7.3.29 PLLC1 PLLDIV Ratio Change Status Register (DCHANGE)

The PLLC1 PLLDIV ratio change status register (DCHANGE) indicates if the PLL1_SYSCLK*n* divide ratio has been modified. DCHANGE is shown in Figure 7-30 and described in Table 7-32.

Figure 7-30. PLLC1 PLLDIV Ratio Change Status Register (DCHANGE)

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		SYS3	SYS2	SYS1
R-0		R-0	R-0	R-0

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

Table 7-32. PLLC1 PLLDIV Ratio Change Status Register (DCHANGE) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	SYS3		PLL1_SYSCLK3 divide ratio is modified.
		0	Ratio is not modified.
		1	Ratio is modified.
1	SYS2		PLL1_SYSCLK2 divide ratio is modified.
		0	Ratio is not modified.
		1	Ratio is modified.
0	SYS1		PLL1_SYSCLK1 divide ratio is modified.
		0	Ratio is not modified.
		1	Ratio is modified.

Copyright © 2013–2016, Texas Instruments Incorporated

7.3.30 PLLC0 Clock Enable Control Register (CKEN)

The PLLC0 clock enable control register (CKEN) controls the PLLC0 OBSCLK and AUXCLK clock. CKEN is shown in Figure 7-31 and described in Table 7-33.

Figure 7-31. PLLC0 Clock Enable Control Register (CKEN)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		OBSEN	AUXEN
R-0		R/W-1	R/W-1

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

Table 7-33. PLLC0 Clock Enable Control Register (CKEN) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	OBSEN		OBSCLK enable. Actual PLLC0 OBSCLK status is shown in the PLLC0 clock status register (CKSTAT).
		0	PLLC0 OBSCLK is disabled.
		1	PLLC0 OBSCLK is enabled. For PLLC0 OBSCLK to toggle, both the OBSEN bit and the OD1EN bit in the PLLC0 oscillator divider 1 register (OSCDIV) must be set to 1.
0	AUXEN		AUXCLK enable. Actual PLLC0 AUXCLK status is shown in the PLLC0 clock status register (CKSTAT).
		0	PLLC0 AUXCLK is disabled.
		1	PLLC0 AUXCLK is enabled.

7.3.31 PLLC1 Clock Enable Control Register (CKEN)

The PLLC1 clock enable control register (CKEN) controls the PLLC1 OBSCLK clock. CKEN is shown in Figure 7-32 and described in Table 7-34.

Figure 7-32. PLLC1 Clock Enable Control Register (CKEN)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		OBSEN	Reserved
R-0		R/W-0	R-0

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

Table 7-34. PLLC1 Clock Enable Control Register (CKEN) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	OBSEN		OBSCLK enable. Actual PLLC1 OBSCLK status is shown in the PLLC1 clock status register (CKSTAT).
		0	PLLC1 OBSCLK is disabled.
		1	PLLC1 OBSCLK is enabled. For PLLC1 OBSCLK to toggle, both the OBSEN bit and the OD1EN bit in the PLLC1 oscillator divider 1 register (OSCDIV) must be set to 1.
0	Reserved	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

7.3.32 PLLC0 Clock Status Register (CKSTAT)

The PLLC0 clock status register (CKSTAT) indicates the PLLC0 OBSCLK and AUXCLK on/off status. The PLL0_SYSCLK status is shown in the PLLC0 SYSCLK status register (SYSTAT). CKSTAT is shown in Figure 7-33 and described in Table 7-35.

Figure 7-33. PLLC0 Clock Status Register (CKSTAT)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		OBSEN	AUXEN
R-0		R-1	R-1

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

Table 7-35. PLLC0 Clock Status Register (CKSTAT) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	OBSEN		OBSCLK on status. PLLC0 OBSCLK is controlled in the PLLC0 oscillator divider 1 register (OSCDIV) by the OBSEN bit in the PLLC0 clock enable control register (CKEN).
		0	PLLC0 OBSCLK is off.
		1	PLLC0 OBSCLK is on.
0	AUXEN		AUXCLK on status. PLLC0 AUXCLK is controlled by the AUXEN bit in the PLLC0 clock enable control register (CKEN).
		0	PLLC0 AUXCLK is off.
		1	PLLC0 AUXCLK is on.

Copyright © 2013–2016, Texas Instruments Incorporated

7.3.33 PLLC1 Clock Status Register (CKSTAT)

The PLLC1 clock status register (CKSTAT) indicates the PLLC1 OBSCLK on/off status. The PLL1_SYSCLK status is shown in the PLLC1 SYSCLK status register (SYSTAT). CKSTAT is shown in Figure 7-34 and described in Table 7-36.

Figure 7-34. PLLC1 Clock Status Register (CKSTAT)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		OBSEN	Reserved
R-2h		R-0	R-0

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

Table 7-36. PLLC1 Clock Status Register (CKSTAT) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	OBSEN		OBSCLK on status. PLLC1 OBSCLK is controlled in the PLLC1 oscillator divider 1 register (OSCDIV) by the OBSEN bit in the PLLC1 clock enable control register (CKEN).
		0	PLLC1 OBSCLK is off.
		1	PLLC1 OBSCLK is on.
0	Reserved	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

7.3.34 PLLC0 SYSCLK Status Register (SYSTAT)

The PLLC0 SYSCLK status register (SYSTAT) indicates the PLL0_SYSCLK*n* on/off status. The actual default is determined by the actual clock on/off status, which depends on the D*n*EN bit in PLLC0 PLLDIV*n*. SYSTAT is shown in Figure 7-35 and described in Table 7-37.

Figure 7-35. PLLC0 SYSCLK Status Register (SYSTAT)

31							8
Reserved							
R-1							

7	6	5	4	3	2	1	0
Reserved	SYS7ON	SYS6ON	SYS5ON	SYS4ON	SYS3ON	SYS2ON	SYS1ON
R-1	R-1	R-1	R-1	R-1	R-1	R-1	R-1

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

Table 7-37. PLLC0 SYSCLK Status Register (SYSTAT) Field Descriptions

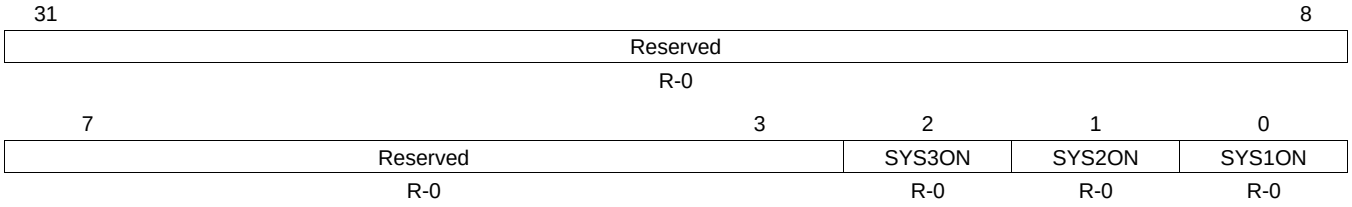
Bit	Field	Value	Description
31-7	Reserved	3h	Reserved
6	SYS7ON		PLL0_SYSCLK7 on status.
		0	Off
		1	On
5	SYS6ON		PLL0_SYSCLK6 on status.
		0	Off
		1	On
4	SYS5ON		PLL0_SYSCLK5 on status.
		0	Off
		1	On
3	SYS4ON		PLL0_SYSCLK4 on status.
		0	Off
		1	On
2	SYS3ON		PLL0_SYSCLK3 on status.
		0	Off
		1	On
1	SYS2ON		PLL0_SYSCLK2 on status.
		0	Off
		1	On
0	SYS1ON		PLL0_SYSCLK1 on status.
		0	Off
		1	On

Copyright © 2013–2016, Texas Instruments Incorporated

7.3.35 PLLC1 SYSCLK Status Register (SYSTAT)

The PLLC1 SYSCLK status register (SYSTAT) indicates the PLL1_SYSCLK*n* on/off status. The actual default is determined by the actual clock on/off status, which depends on the D*n*EN bit in PLLC1 PLLDIV*n*. SYSTAT is shown in Figure 7-36 and described in Table 7-38.

Figure 7-36. PLLC1 SYSCLK Status Register (SYSTAT)



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

Table 7-38. PLLC1 SYSCLK Status Register (SYSTAT) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	SYS3ON		PLL1_SYSCLK3 on status.
		0	Off
		1	On
1	SYS2ON		PLL1_SYSCLK2 on status.
		0	Off
		1	On
0	SYS1ON		PLL1_SYSCLK1 on status.
		0	Off
		1	On

Copyright © 2013–2016, Texas Instruments Incorporated

7.3.36 Emulation Performance Counter 0 Register (EMUCNT0)

The emulation performance counter 0 register (EMUCNT0) is shown in Figure 7-37 and described in Table 7-39. EMUCNT0 is for emulation performance profiling. It counts in a divide-by-4 of the system clock. To start the counter, a write must be made to EMUCNT0. This register is not writable, but only used to start the register. After the register is started, it can not be stopped except for power on reset. When EMUCNT0 is read, it snapshots EMUCNT0 and EMUCNT1. The snapshot version is what is read. It is important to read the EMUCNT0 followed by EMUCNT1 or else the snapshot version may not get updated correctly.

Figure 7-37. Emulation Performance Counter 0 Register (EMUCNT0)

31	0
COUNT	
R-0	

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

Table 7-39. Emulation Performance Counter 0 Register (EMUCNT0) Field Descriptions

Bit	Field	Value	Description
31-0	COUNT	0-FFFF FFFFh	Counter value for lower 64-bits.

7.3.37 Emulation Performance Counter 1 Register (EMUCNT1)

The emulation performance counter 1 register (EMUCNT1) is shown in Figure 7-38 and described in Table 7-40. EMUCNT1 is for emulation performance profiling. To start the counter, a write must be made to EMUCNT0. This register is not writable, but only used to start the register. After the register is started, it can not be stopped except for power on reset. When EMUCNT0 is read, it snapshots EMUCNT0 and EMUCNT1. The snapshot version is what is read. It is important to read the EMUCNT0 followed by EMUCNT1 or else the snapshot version may not get updated correctly.

Figure 7-38. Emulation Performance Counter 1 Register (EMUCNT1)

31	0
COUNT	
R-0	

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

Table 7-40. Emulation Performance Counter 1 Register (EMUCNT1) Field Descriptions

Bit	Field	Value	Description
31-0	COUNT	0-FFFF FFFFh	Counter value for upper 64-bits.

Copyright © 2013–2016, Texas Instruments Incorporated

Power and Sleep Controller (PSC)

Topic **Page**

8.1	Introduction	153
8.2	Power Domain and Module Topology	153
8.3	Executing State Transitions	158
8.4	IcePick Emulation Support in the PSC	159
8.5	PSC Interrupts	159
8.6	PSC Registers	162

Copyright © 2013–2016, Texas Instruments Incorporated

8.1 Introduction

The Power and Sleep Controllers (PSC) are responsible for managing transitions of system power on/off, clock on/off, resets (device level and module level). It is used primarily to provide granular power control for on chip modules (peripherals and CPU). A PSC module consists of a Global PSC (GPSC) and a set of Local PSCs (LPSCs).

The GPSC contains memory mapped registers, PSC interrupts, a state machine for each peripheral/module it controls. An LPSC is associated with every module that is controlled by the PSC and provides clock and reset control. Many of the operations of the PSC are transparent to user (software), such as power on and reset control. However, the PSC module(s) also provide you with interface to control several important power, clock and reset operations. The module level power, clock and reset operations managed and controlled by the PSC are the focus of this chapter.

The PSC includes the following features:

- Manages chip power-on/off
- Provides a software interface to:
  - Control module clock enable/disable
  - Control module reset
  - Control CPU local reset
- Manages on-chip RAM sleep modes (for L3 RAM)
- Supports IcePick emulation features: power, clock and reset

8.2 Power Domain and Module Topology

This device includes two PSC modules. Each PSC module consists of:

- an Always On power domain
- an additional pseudo/internal power domain that manages the sleep modes for the RAMs present in the L3 RAM

Each PSC module controls clock states for several on the on chip modules, controllers and interconnect components. Table 8-1 and Table 8-2 lists the set of peripherals/modules that are controlled by the PSC, the power domain they are associated with, the LPSC assignment and the default (power-on reset) module states. See the device-specific data manual for the peripherals available on a given device. The module states and terminology are defined in Section 8.2.2.

Even though there are 2 PSC modules with 2 power domains each on the device, both PSC modules and all the power domains are powered by the CVDD pins of the device. All power domains are on when the chip is powered on. There is no provision to remove power externally for the non Always On domains, that is, the pseudo/internal power domains.

There are a few modules/peripherals on the device that do not have an LPSC assigned to them. These modules do not have their module reset/clocks controlled by the PSC module. The decision to assign an LPSC to a module on a device is primarily based on whether or not disabling the clocks to a module will result in significant power savings. This typically depends on the size and the frequency of operation of the module.

NOTE: There are no LPSCs for peripherals in the Async2 clock domain (this includes RTC, Timer64P0/P1, and I2C0); from a power savings stand point, clock-gating these peripherals does not result in significant power savings.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 8-1. PSC0 Default Module Configuration

LPSC Number	Module Name	Power Domain	Default Module State	Auto Sleep/ Wake Only
0	EDMA3_0 Channel Controller 0	AlwaysON (PD0)	SwRstDisable	—
1	EDMA3_0 Transfer Controller 0	AlwaysON (PD0)	SwRstDisable	—
2	EDMA3_0 Transfer Controller 1	AlwaysON (PD0)	SwRstDisable	—
3	EMIFA (BR7)	AlwaysON (PD0)	SwRstDisable	—
4	SPI0	AlwaysON (PD0)	SwRstDisable	—
5	MMC/SD0	AlwaysON (PD0)	SwRstDisable	—
6	ARM Interrupt Controller	AlwaysON (PD0)	Enable	—
7	ARM RAM/ROM	AlwaysON (PD0)	Enable	Yes
8	Not Used	—	—	—
9	UART0	AlwaysON (PD0)	SwRstDisable	—
10	SCR0 (BR0, BR1, BR2, BR8)	AlwaysON (PD0)	Enable	Yes
11	SCR1 (BR4)	AlwaysON (PD0)	Enable	Yes
12	SCR2 (BR3, BR5, BR6)	AlwaysON (PD0)	Enable	Yes
13	PRU	AlwaysON (PD0)	SwRstDisable	—
14	ARM	AlwaysON (PD0)	SwRstDisable	—
15	Not Used	—	—	—

Table 8-2. PSC1 Default Module Configuration

LPSC Number	Module Name	Power Domain	Default Module State	Auto Sleep/ Wake Only
0	EDMA3_1 Channel Controller 0	AlwaysON (PD0)	SwRstDisable	—
1	USB0 (USB2.0)	AlwaysON (PD0)	SwRstDisable	—
2	Not Used	—	—	—
3	GPIO	AlwaysON (PD0)	SwRstDisable	—
4	HPI	AlwaysON (PD0)	SwRstDisable	—
5	Not Used	—	—	—
6	DDR2/mDDR	AlwaysON (PD0)	SwRstDisable	—
7	McASP0 (+ McASP0 FIFO)	AlwaysON (PD0)	SwRstDisable	—
8	Not Used	—	—	—
9	VPIF	AlwaysON (PD0)	SwRstDisable	—
10	SPI1	AlwaysON (PD0)	SwRstDisable	—
11	I2C1	AlwaysON (PD0)	SwRstDisable	—
12	UART1	AlwaysON (PD0)	SwRstDisable	—
13	UART2	AlwaysON (PD0)	SwRstDisable	—
14	McBSP0 (+ McBSP0 FIFO)	AlwaysON (PD0)	SwRstDisable	—
15	McBSP1 (+ McBSP1 FIFO)	AlwaysON (PD0)	SwRstDisable	—
16	LCDC	AlwaysON (PD0)	SwRstDisable	—
17	eHRPWM0/1	AlwaysON (PD0)	SwRstDisable	—
18	MMC/SD1	AlwaysON (PD0)	SwRstDisable	—
19	uPP	AlwaysON (PD0)	SwRstDisable	—
20	eCAP0/1/2	AlwaysON (PD0)	SwRstDisable	—
21	EDMA3_1 Transfer Controller 0	AlwaysON (PD0)	SwRstDisable	—
22-23	Not Used	—	—	—
24	SCR F0	AlwaysON (PD0)	Enable	Yes

Copyright © 2013–2016, Texas Instruments Incorporated

Table 8-2. PSC1 Default Module Configuration (continued)

LPSC Number	Module Name	Power Domain	Default Module State	Auto Sleep/ Wake Only
25	SCR F1	AlwaysON (PD0)	Enable	Yes
26	SCR F2	AlwaysON (PD0)	Enable	Yes
27	SCR F6	AlwaysON (PD0)	Enable	Yes
28	SCR F7	AlwaysON (PD0)	Enable	Yes
29	SCR F8	AlwaysON (PD0)	Enable	Yes
30	BR F7	AlwaysON (PD0)	Enable	Yes
31	On-chip RAM	PD_SHRAM	Enable	—

8.2.1 Power Domain States

A power domain can only be in one of the two states: ON or OFF, defined as follows:

- ON: power to the domain is on
- OFF: power to the domain is off

In this device, for both PSC0 and PSC1, the Always ON domain (or PD0 power domain), is always in the ON state when the chip is powered-on. This domain is not programmable to OFF state (See details on PDCTL register).

Additionally, for both PSC0 and PSC1, the PD1 power domains, the internal/pseudo power domain can either be in the ON state or OFF state. Furthermore, for these power domains the transition from ON to OFF state is further qualified by the PSC0/1.PDCTL1.PDMODE settings. The PDCTL1.PDMODE settings determines the various sleep mode for the on-chip RAM associated with module in the PD1 domain.

- On PSC1 PD1/PD_SHRAM Domain: Controls the sleep state for the 128KB On-chip RAM

NOTE: Currently programming the PD1 power domain state to OFF is not supported. You should leave both the PDCTL1.NEXT and PDCTL1.PDMODE values at default/power on reset values.

Both PD0 and PD1 power domains in PSC0 and PSC1 are powered by the CVDD pins of the device. There is no capability to individually remove voltage/power from the on-chip RAM power domains.

Copyright © 2013–2016, Texas Instruments Incorporated

8.2.2 Module States

The PSC defines several possible states for a module. This various states are essentially a combination of the module reset asserted or de-asserted and module clock on/enabled or off/disabled. The various module states are defined in Table 8-3.

The key difference between the Auto Sleep and Auto Wake states is that once the module is configured in Auto Sleep mode, it will transition back to the clock disabled state (automatically sleep) after servicing the internal read/write access request where as in Auto Wake mode, on receiving the first internal read/write access request, the module will permanently transition from the clock disabled to clock enabled state (automatically wake).

When the module state is programmed to Disable, SwRstDisable, Auto Sleep or Auto Wake modes, where in the module clocks are off/disabled, an external event or I/O request cannot enable the clocks. For the module to appropriately respond to such external request, it would need to be reconfigured to the Enable state.

8.2.2.1 Auto Sleep/Wake Only Configurations and Limitation

NOTE: Currently no modules should be configured in Auto Sleep or Auto Wake modes. If the module clocks need to gated/disabled for power savings, you should program the module state to Disable. For Auto Sleep/Auto Wake Only modules, disabling the clock is not supported and they should be kept in their default “Enable” state.

Table 8-1 and Table 8-2 each have a column to indicate whether or not the LPSC configuration for a module is Auto Sleep/Wake Only. Modules that have a “Yes” marked for the Auto Sleep/Wake Only column can be programmed in software to be in Enable, Auto Sleep and Auto Wake states only; that is, if the software tries to program these modules to Disable, SyncReset, or SwRstDisable state the power sleep controller ignores these transition requests and transitions the module state to Enable.

8.2.2.2 Local Reset

In addition to module reset, the following module can be reset using a special local reset that is also a part of the PSC module control for resets.

- ARM: When the ARM local reset is asserted the entire ARM processor is reset , including cache etc. This does not include the ARM RAM/ROM or ARM interrupt controller module as these exist outside the ARM core. The local reset for ARM additionally ensures that any outstanding requests are completed before ARM is reset, therefore for scenarios where it is needed to just reset the ARM locally but not change the state of clocks, user can use ARM local reset feature.

The procedures for asserting and de-asserting the local reset are as follows (where *n* corresponds to the module that supports local reset):

1. Clear the LRST bit in the module control register (MDCTL*n*) to 0 to assert the module’s local reset.
2. Set the LRST bit in the module control register (MDCTL*n*) to 1 to de-assert module’s local reset.

If the CPU is in the enable state, it immediately executes program instructions after reset is de-asserted.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 8-3. Module States

Module State	Module Reset	Module Clock	Module State Definition
Enable	De-asserted	On	A module in the enable state has its module reset de-asserted and it has its clock on. This is the normal operational state for a given module
Disable	De-asserted	Off	A module in the disabled state has its module reset de-asserted and it has its module clock off. This state is typically used for disabling a module clock to save power. This device is designed in full static CMOS, so when you stop a module clock, it retains the module's state. When the clock is restarted, the module resumes operating from the stopping point.
SyncReset	Asserted	On	A module state in the SyncReset state has its module reset asserted and it has its clock on. Generally, software is not expected to initiate this state
SwRstDisable	Asserted	Off	A module in the SwResetDisable state has its module reset asserted and it has its clock disabled. After initial power-on, several modules come up in the SwRstDisable state. Generally, software is not expected to initiate this state
Auto Sleep	De-asserted	Off	A module in the Auto Sleep state also has its module reset de-asserted and its module clock disabled, similar to the Disable state. However this is a special state, once a module is configured in this state by software, it can "automatically" transition to "Enable" state whenever there is an internal read/write request made to it, and after servicing the request it will "automatically" transition into the sleep state (with module reset re de-asserted and module clock disabled), without any software intervention. The transition from sleep to enabled and back to sleep state has some cycle latency associated with it. It is not envisioned to use this mode when peripherals are fully operational and moving data. See Section 8.2.2.1 for additional considerations, constraints, limitations around this mode.
Auto Wake	De-asserted	Off	A module in the Auto Wake state also has its module reset de-asserted and its module clock disabled, similar to the Disable state. However this is a special state, once a module is configured in this state by software, it will "automatically" transition to "Enable" state whenever there is an internal read/write request made to it, and will remain in the "Enabled" state from then on (with module reset re de-asserted and module clock on), without any software intervention. The transition from sleep to enabled state has some cycle latency associated with it. It is not envisioned to use this mode when peripherals are fully operational and moving data. See Section 8.2.2.1 for additional considerations, constraints, limitations around this mode.

Copyright © 2013–2016, Texas Instruments Incorporated

8.3 Executing State Transitions

This section describes how to execute the state transitions modules.

8.3.1 Power Domain State Transitions

This device consists of two types of domain (in each PSC controller):

- Always On domain(s)
- pseudo/RAM power domain(s)

The Always On power domains are always in the ON state when the chip is powered on. You are not allowed to change the power domain state to OFF.

The pseudo/RAM power domains allow internally powering down the state of the RAMs associated with these domains (On-chip RAM for PD_SHRAM in PSC1) so that these RAMs can run in lower power sleep modes via the power sleep controller.

NOTE: Currently powering down the RAMs via the pseudo/RAM power domain is not supported; therefore, these domains and the RAM should be left in their default power on state.

As mentioned in Section 8.2, the pseudo/RAM power domains are powered down internally, and in this context powering down does not imply removing the core voltage from pins externally.

8.3.2 Module State Transitions

This section describes the procedure for transitioning the module state (clock and reset control). Note that some peripherals have special programming requirements and additional recommended steps you must take before you can invoke the PSC module state transition. See the individual peripheral user guides for more details. For example, the external memory controller requires that you first place the SDRAM memory in self-refresh mode before you invoke the PSC module state transitions, if you want to maintain the memory contents.

The following procedure is directly applicable for all modules that are controlled via the PSC (shown in Table 8-1 and Table 8-2), except for the core(s). To transition the ARM module state, there are additional system considerations and constraints that you should be aware of. These system considerations and the procedure for transitioning the ARM module state are described in details in the *Power Management* chapter.

NOTE: In the following procedure, x is 0 for modules in PD0 (Power Domain 0 or Always On domain) and x is 1 for modules in PD1 (Power Domain 1). See Table 8-1 and Table 8-2 for power domain associations.

The procedure for module state transitions is:

1. Wait for the GOSTAT[x] bit in PTSTAT to clear to 0. You must wait for any previously initiated transitions to finish before initiating a new transition.
2. Set the NEXT bit in MDCTL*n* to SwRstDisable (0), SyncReset (1), Disable (2h), Enable (3h), Auto Sleep (4h) or Auto Wake (5h).

   NOTE: You may set transitions in multiple NEXT bits in MDCTL*n* in this step. Transitions do not actually take place until you set the GO[x] bit in PTCMD in a later step.
3. Set the GO[x] bit in PTCMD to 1 to initiate the transition(s).
4. Wait for the GOSTAT[x] bit in PTSTAT to clear to 0. The modules are safely in the new states only after the GOSTAT[x] bit in PTSTAT is cleared to 0.

Copyright © 2013–2016, Texas Instruments Incorporated

8.4 IcePick Emulation Support in the PSC

The PSC supports IcePick commands that allow IcePick emulation tools to have some control over the state of power domains and modules.

- ARM [MDCTL14]

In particular, Table 8-4 shows IcePick emulation commands recognized by the PSC.

Table 8-4. IcePick Emulation Commands

Power On and Enable Features	Power On and Enable Descriptions	Reset Features	Reset Descriptions
Inhibit Sleep	Allows emulation to prevent software from transitioning the module out of the enable state.	Assert Reset	Allows emulation to assert the module's local reset.
Force Power	Allows emulation to force the power domain into an on state. Not applicable as AlwaysOn power domain is always on.	Wait Reset	Allows emulation to keep local reset asserted for an extended period of time after software initiates local reset de-assert.
Force Active	Allows emulation to force the module into the enable state.	Block Reset	Allows emulation to block software initiated local and module resets.

NOTE: When emulation tools remove the above commands, the PSC immediately executes a state transition based on the current values in the NEXT bit in PDCTL0 and the NEXT bit in MDCTL*n*, as set by software.

8.5 PSC Interrupts

The PSC has an interrupt that is tied to the core interrupt controller. This interrupt is named PSCINT in the interrupt map. The PSC interrupt is generated when certain IcePick emulation events occur.

8.5.1 Interrupt Events

The PSC interrupt is generated when any of the following events occur:

- Power Domain Emulation Event (applies to pseudo/RAM power domain only)
- Module State Emulation event
- Module Local Reset Emulation event

These interrupt events are summarized in Table 8-5 and described in more detail in this section.

Table 8-5. PSC Interrupt Events

Interrupt Enable Bits		
Control Register	Enable Bit	Interrupt Condition
PDCTL*n*	EMUIHBIE	Interrupt occurs when the emulation alters the power domain state
MDCTL*n*	EMUIHBIE	Interrupt occurs when the emulation alters the module state
MDCTL*n*	EMURSTIE	Interrupt occurs when the emulation tries to alter the module's local reset

The PSC interrupt events only apply when IcePick emulation alters the state of the module from the user-programmed state in the NEXT bit in the MDCTL/PDCTL registers. IcePick support only applies to the modules listed in Section 8.4; therefore, the PSC interrupt conditions only apply to those modules listed.

Copyright © 2013–2016, Texas Instruments Incorporated

8.5.1.1 Power Domain Emulation Events

A power domain emulation event occurs when emulation alters the state of a power domain (does not apply to the Always On domain). Status is reflected in the EMUIHB bit in PDSTAT*n*. In particular, a power domain emulation event occurs under the following conditions:

- When inhibit sleep is asserted by emulation and software attempts to transition the module out of the on state
- When force power is asserted by emulation and power domain is not already in the on state
- When force active is asserted by emulation and power domain is not already in the on state

8.5.1.2 Module State Emulation Events

A module state emulation event occurs when emulation alters the state of a module. Status is reflected in the EMUIHB bit in the module status register (MDSTAT*n*). In particular, a module state emulation event occurs under the following conditions:

- When inhibit sleep is asserted by emulation and software attempts to transition the module out of the enable state
- When force active is asserted by emulation and module is not already in the enable state

8.5.1.3 Local Reset Emulation Events

A local reset emulation event occurs when emulation alters the local reset of a module. Status is reflected in the EMURST bit in the module status register (MDSTAT*n*). In particular, a module local reset emulation event occurs under the following conditions:

- When assert reset is asserted by emulation although software de-asserted the local reset
- When wait reset is asserted by emulation
- When block reset is asserted by emulation and software attempts to change the state of local reset

8.5.2 Interrupt Registers

The PSC interrupt enable bits are: the EMUIHBIE bit in PDCTL1 (PSC0), the EMUIHBIE and the EMURSTIE bits in MDCTL*n* (where n is the modules that have IcePick emulation support, as specified in Section 8.4).

NOTE: To interrupt the CPU, the power sleep controller interrupt (PSC0_ALLINT and PSC1_ALLINT) must also be enabled appropriately in the ARM interrupt controller. For details on the ARM interrupt controller, see the *ARM Interrupt Controller (AINTC)* chapter.

The PSC interrupt status bits are:

- For ARM:
  - The M[14] bit in the module error pending register 0 (MERRPR0) in PSC0 module.
  - The EMUIHB and the EMURST bits in the module status register for ARM (MDSTAT14).

The status bit in MERRPR0 and PERRPR registers is read by software to determine which module or power domain has generated an emulation interrupt and then software can read the corresponding status bits in MDSTAT register or the PDSTATn (PDCTL1 for pseudo/RAM power domain in PSC0) to determine which event caused the interrupt.

The PSC interrupt can be cleared by writing to bit corresponding to the module number in the module error clear register (MERRCR0), or the bit corresponding to the power domain number in the power error clear register (PERRCR) in PSC0 module.

The PSC interrupt evaluation bit is the ALLEV bit in the INTEVAL register. When set, this bit forces the PSC interrupt logic to re-evaluate event status. If any events are still active (if any status bits are set) when the ALLEV bit in the INTEVAL is set to 1, the PSC interrupt is re-asserted to the interrupt controller. Set the ALLEV bit in the INTEVAL before exiting your PSC interrupt service routine to ensure that you do not miss any PSC interrupts.

Copyright © 2013–2016, Texas Instruments Incorporated

See Section 8.6 for a description of the PSC registers.

8.5.3 Interrupt Handling

Handle the PSC interrupts as described in the following procedure:

First, enable the interrupt:

1. Set the EMUIHBIE bit in PDCTL*n*, the EMUIHBIE and the EMURSTIE bits in MDCTL*n* to enable the interrupt events that you want.

NOTE: The PSC interrupt is sent to the device interrupt controller when at least one enabled event becomes active.

2. Enable the power sleep controller interrupt (PSC*n*_ALLINT) in the device interrupt controller. To interrupt the CPU, PSC*n*_ALLINT must be enabled in the device interrupt controller. See the for more information on interrupts.

The CPU enters the interrupt service routine (ISR) when it receives the interrupt.

1. Read the P[n] bit in PERRPR, and/or the M[n] bit in MERRPR0, the M[n] bit in MERRPR1, to determine the source of the interrupt(s).
2. For each active event that you want to service:
   (a) Read the event status bits in PDSTAT*n* and MDSTAT*n*, depending on the status bits read in the previous step to determine the event that caused the interrupt.
   (b) Service the interrupt as required by your application.
   (c) Write the M[n] bit in MERRCR*n* and the P[n] bit in PERRCR to clear corresponding status.
   (d) Set the ALLEV bit in INTEVAL. Setting this bit reasserts the PSC interrupt to the device interrupt controller, if there are still any active interrupt events.

Copyright © 2013–2016, Texas Instruments Incorporated

8.6 PSC Registers

Table 8-6 lists the memory-mapped registers for the PSC0 and Table 8-7 lists the memory-mapped registers for the PSC1.

Table 8-6. Power and Sleep Controller 0 (PSC0) Registers

Address	Acronym	Register Description	Section
01C1 0000h	REVID	Revision Identification Register	Section 8.6.1
01C1 0018h	INTEVAL	Interrupt Evaluation Register	Section 8.6.2
01C1 0040h	MERRPR0	Module Error Pending Register 0 (module 0-15)	Section 8.6.3
01C1 0050h	MERRCR0	Module Error Clear Register 0 (module 0-15)	Section 8.6.5
01C1 0060h	PERRPR	Power Error Pending Register	Section 8.6.7
01C1 0068h	PERRCR	Power Error Clear Register	Section 8.6.8
01C1 0120h	PTCMD	Power Domain Transition Command Register	Section 8.6.9
01C1 0128h	PTSTAT	Power Domain Transition Status Register	Section 8.6.10
01C1 0200h	PDSTAT0	Power Domain 0 Status Register	Section 8.6.11
01C1 0204h	PDSTAT1	Power Domain 1 Status Register	Section 8.6.12
01C1 0300h	PDCTL0	Power Domain 0 Control Register	Section 8.6.13
01C1 0304h	PDCTL1	Power Domain 1 Control Register	Section 8.6.14
01C1 0400h	PDCFG0	Power Domain 0 Configuration Register	Section 8.6.15
01C1 0404h	PDCFG1	Power Domain 1 Configuration Register	Section 8.6.16
01C1 0800h-01C1 083Ch	MDSTAT0-MDSTAT15	Module Status *n* Register (modules 0-15)	Section 8.6.17
01C1 0A00h-01C1 0A3Ch	MDCTL0-MDCTL15	Module Control *n* Register (modules 0-15)	Section 8.6.18

Table 8-7. Power and Sleep Controller 1 (PSC1) Registers

Address	Acronym	Register Description	Section
01E2 7000h	REVID	Revision Identification Register	Section 8.6.1
01E2 7018h	INTEVAL	Interrupt Evaluation Register	Section 8.6.2
01E2 7040h	MERRPR0	Module Error Pending Register 0 (module 0-31)	Section 8.6.4
01E2 7050h	MERRCR0	Module Error Clear Register 0 (module 0-31)	Section 8.6.6
01E2 7060h	PERRPR	Power Error Pending Register	Section 8.6.7
01E2 7068h	PERRCR	Power Error Clear Register	Section 8.6.8
01E2 7120h	PTCMD	Power Domain Transition Command Register	Section 8.6.9
01E2 7128h	PTSTAT	Power Domain Transition Status Register	Section 8.6.10
01E2 7200h	PDSTAT0	Power Domain 0 Status Register	Section 8.6.11
01E2 7204h	PDSTAT1	Power Domain 1 Status Register	Section 8.6.12
01E2 7300h	PDCTL0	Power Domain 0 Control Register	Section 8.6.13
01E2 7304h	PDCTL1	Power Domain 1 Control Register	Section 8.6.14
01E2 7400h	PDCFG0	Power Domain 0 Configuration Register	Section 8.6.15
01E2 7404h	PDCFG1	Power Domain 1 Configuration Register	Section 8.6.16
01E2 7800h-01E2 787Ch	MDSTAT0-MDSTAT31	Module Status *n* Register (modules 0-31)	Section 8.6.17
01E2 7A00h-01E2 7A7Ch	MDCTL0-MDCTL31	Module Control *n* Register (modules 0-31)	Section 8.6.19

Copyright © 2013–2016, Texas Instruments Incorporated

8.6.1 Revision Identification Register (REVID)

The revision identification register (REVID) is shown in Figure 8-1 and described in Table 8-8.

Figure 8-1. Revision Identification Register (REVID)

31	0
REV	
R-4482 5A00h	

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

Table 8-8. Revision Identification Register (REVID) Field Descriptions

Bit	Field	Value	Description
31-0	REV	4482 5A00h	Peripheral revision ID.

8.6.2 Interrupt Evaluation Register (INTEVAL)

The interrupt evaluation register (INTEVAL) is shown in Figure 8-2 and described in Table 8-9.

Figure 8-2. Interrupt Evaluation Register (INTEVAL)

31	16
Reserved	
R-0	

15	1	0
Reserved		ALLEV
R-0		W-0

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

Table 8-9. Interrupt Evaluation Register (INTEVAL) Field Descriptions

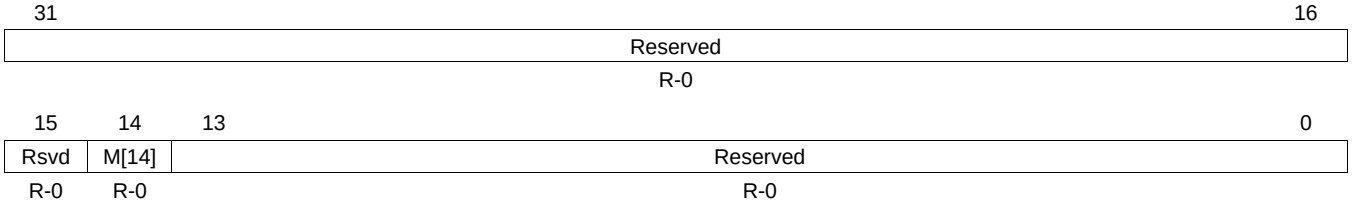
Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	ALLEV		Evaluate PSC interrupt (PSC*n*_ALLINT).
		0	A write of 0 has no effect.
		1	A write of 1 re-evaluates the interrupt condition.

Copyright © 2013–2016, Texas Instruments Incorporated

8.6.3 PSC0 Module Error Pending Register 0 (modules 0-15) (MERRPR0)

The PSC0 module error pending register 0 (MERRPR0) is shown in Figure 8-3 and described in Table 8-10.

Figure 8-3. PSC0 Module Error Pending Register 0 (MERRPR0)



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

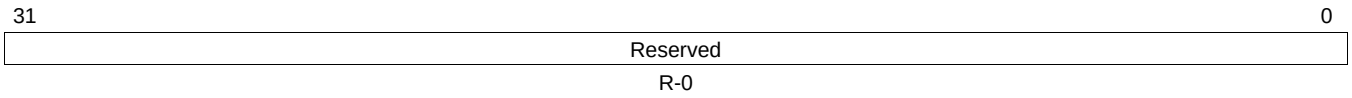
Table 8-10. PSC0 Module Error Pending Register 0 (MERRPR0) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved	0	Reserved
14	M[14]		Module interrupt status bit for module 14 (ARM).
		0	Module 14 does not have an error condition.
		1	Module 14 has an error condition. See the module status 14 register (MDSTAT14) for the error condition.
13-0	Reserved	0	Reserved

8.6.4 PSC1 Module Error Pending Register 0 (modules 0-31) (MERRPR0)

The PSC1 module error pending register 0 (MERRPR0) is shown in Figure 8-4.

Figure 8-4. PSC1 Module Error Pending Register 0 (MERRPR0)



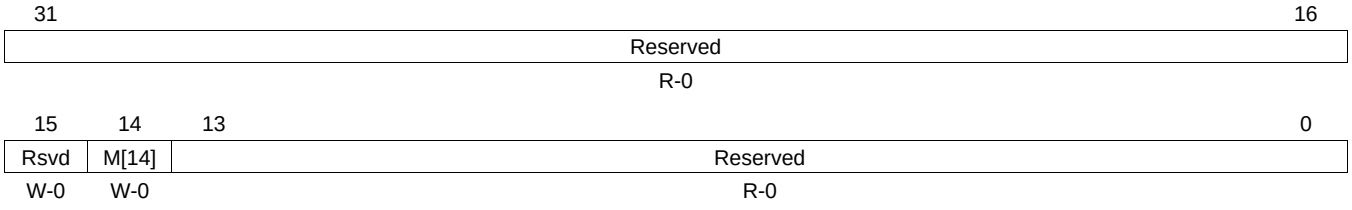
LEGEND: R = Read only; -*n* = value after reset

Copyright © 2013–2016, Texas Instruments Incorporated

8.6.5 PSC0 Module Error Clear Register 0 (modules 0-15) (MERRCR0)

The PSC0 module error clear register 0 (MERRCR0) is shown in Figure 8-5 and described in Table 8-11.

Figure 8-5. PSC0 Module Error Clear Register 0 (MERRCR0)



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

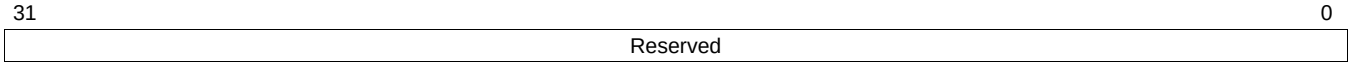
Table 8-11. PSC0 Module Error Clear Register 0 (MERRCR0) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	Reserved	0	Reserved. Write the default value when modifying this register.
14	M[14]		Clears the interrupt status bit (M[14]) set in the PSC0 module error pending register 0 (MERRPR0) and the interrupt status bits set in the module status 14 register (MDSTAT14).
		0	A write of 0 has no effect.
		1	A write of 1 clears the M[14] bit in MERRPR0 and the EMUIHB and EMURST bits in MDSTAT14.
13-0	Reserved	0	Reserved

8.6.6 PSC1 Module Error Clear Register 0 (modules 0-31) (MERRCR0)

The PSC1 module error clear register 0 (MERRCR0) is shown in Figure 8-6.

Figure 8-6. PSC1 Module Error Clear Register 0 (MERRCR0)



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

Copyright © 2013–2016, Texas Instruments Incorporated

8.6.7 Power Error Pending Register (PERRPR)

The power error pending register (PERRPR) is shown in Figure 8-7 and described in Table 8-12.

Figure 8-7. Power Error Pending Register (PERRPR)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		P[1]	Rsvd
R-0		R-0	R-0

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

Table 8-12. Power Error Pending Register (PERRPR) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	P[1]		RAM/Pseudo (PD1) power domain interrupt status.
		0	RAM/Pseudo power domain does not have an error condition.
		1	RAM/Pseudo power domain has an error condition. See the power domain 1 status register (PDSTAT1) for the error condition.
0	Reserved	0	Reserved

8.6.8 Power Error Clear Register (PERRCR)

The power error clear register (PERRCR) is shown in Figure 8-8 and described in Table 8-13.

Figure 8-8. Power Error Clear Register (PERRCR)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		P[1]	Rsvd
R-0		W-0	R-0

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

Table 8-13. Power Error Clear Register (PERRCR) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	P[1]		Clears the interrupt status bit (P) set in the power error pending register (PERRPR) and the interrupt status bits set in the power domain 1 status register (PDSTAT1).
		0	A write of 0 has no effect.
		1	A write of 1 clears the P bit in PERRPR and the interrupt status bits in PDSTAT1.
0	Reserved	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

8.6.9 Power Domain Transition Command Register (PTCMD)

The power domain transition command register (PTCMD) is shown in Figure 8-9 and described in Table 8-14.

Figure 8-9. Power Domain Transition Command Register (PTCMD)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		GO[1]	GO[0]
R-0		W-0	W-0

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

Table 8-14. Power Domain Transition Command Register (PTCMD) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	GO[1]		RAM/Pseudo (PD1) power domain GO transition command.
		0	A write of 0 has no effect.
		1	A write of 1 causes the PSC to evaluate all the NEXT fields relevant to this power domain (including PDCTL.NEXT for this domain, and MDCTL.NEXT for all the modules residing on this domain). If any of the NEXT fields are not matching the corresponding current state (PDSTAT.STATE, MDSTAT.STATE), the PSC will transition those respective domain/modules to the new NEXT state.
0	GO[0]		Always ON (PD0) power domain GO transition command.
		0	A write of 0 has no effect.
		1	A write of 1 causes the PSC to evaluate all the NEXT fields relevant to this power domain (including MDCTL.NEXT for all the modules residing on this domain). If any of the NEXT fields are not matching the corresponding current state (MDSTAT.STATE), the PSC will transition those respective domain/modules to the new NEXT state.

Copyright © 2013–2016, Texas Instruments Incorporated

8.6.10 Power Domain Transition Status Register (PTSTAT)

The power domain transition status register (PTSTAT) is shown in Figure 8-10 and described in Table 8-15 .

Figure 8-10. Power Domain Transition Status Register (PTSTAT)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		GOSTAT[1]	GOSTAT[0]
R-0		R-0	R-0

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

Table 8-15. Power Domain Transition Status Register (PTSTAT) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	GOSTAT[1]		RAM/Pseudo (PD1) power domain transition status.
		0	No transition in progress.
		1	RAM/Pseudo power domain is transitioning (that is, either the power domain is transitioning or modules in this power domain are transitioning).
0	GOSTAT[0]		Always ON (PD0) power domain transition status.
		0	No transition in progress.
		1	Modules in Always ON power domain are transitioning. Always On power domain is transitioning.

Copyright © 2013–2016, Texas Instruments Incorporated

8.6.11 Power Domain 0 Status Register (PDSTAT0)

The power domain 0 status register (PDSTAT0) is shown in Figure 8-11 and described in Table 8-16.

Figure 8-11. Power Domain 0 Status Register (PDSTAT0)

31	16
Reserved	
R-0	

15	12	11	10	9	8	7	5	4	0
Reserved		EMUIHB	Rsvd	PORDONE	POR	Reserved		STATE	
R-0		R-0	R-0	R-0	R-0	R-0		R-0	

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

Table 8-16. Power Domain 0 Status Register (PDSTAT0) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reserved
11	EMUIHB		Emulation alters domain state.
		0	Interrupt is not active. No emulation altering user-desired power domain states.
		1	Interrupt is active. Emulation alters user-desired power domain state.
10	Reserved	0	Reserved
9	PORDONE		Power_On_Reset (POR) Done status
		0	Power domain POR is not done.
		1	Power domain POR is done.
8	POR		Power Domain Power_On_Reset (POR) status. This bit reflects the POR status for this power domain including all modules in the domain.
		0	Power domain POR is asserted.
		1	Power domain POR is de-asserted.
7-5	Reserved	0	Reserved
4-0	STATE	0-1Fh	Power Domain Status.
		0	Power domain is in the off state.
		1h	Power domain is in the on state.
		2h-Fh	Reserved
		10h-1Ah	Power domain is in transition.
		1Bh-1Fh	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

8.6.12 Power Domain 1 Status Register (PDSTAT1)

The power domain 1 status register (PDSTAT1) is shown in Figure 8-12 and described in Table 8-17.

Figure 8-12. Power Domain 1 Status Register (PDSTAT1)

31	16
Reserved	
R-0	

15	12	11	10	9	8	7	5	4	0
Reserved		EMUIHB	Rsvd	PORDONE	POR	Reserved		STATE	
R-0		R-0	R-0	R-0	R-0	R-0		R-0	

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

Table 8-17. Power Domain 1 Status Register (PDSTAT1) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reserved
11	EMUIHB		Emulation alters domain state.
		0	Interrupt is not active. No emulation altering user-desired power domain states.
		1	Interrupt is active. Emulation alters user-desired power domain state.
10	Reserved	0	Reserved
9	PORDONE		Power_On_Reset (POR) Done status
		0	Power domain POR is not done.
		1	Power domain POR is done.
8	POR		Power Domain Power_On_Reset (POR) status. This bit reflects the POR status for this power domain including all modules in the domain.
		0	Power domain POR is asserted.
		1	Power domain POR is de-asserted.
7-5	Reserved	0	Reserved
4-0	STATE	0-1Fh	Power Domain Status.
		0	Power domain is in the off state.
		1h	Power domain is in the on state.
		2h-Fh	Reserved
		10h-1Ah	Power domain is in transition.
		1Bh-1Fh	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

8.6.13 Power Domain 0 Control Register (PDCTL0)

The power domain 0 control register (PDCTL0) is shown in Figure 8-13 and described in Table 8-18.

Figure 8-13. Power Domain 0 Control Register (PDCTL0)

31	24	23	16
Reserved		WAKECNT	
R-0		R/W-1Fh	

15	12	11	10	9	8	7	1	0
PDMODE		Reserved		EMUIHBIE	Rsvd	Reserved		NEXT
R-Fh		R-0		R/W-0	R-1	R-0		R/W-1

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

Table 8-18. Power Domain 0 Control Register (PDCTL0) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reserved
23-16	WAKECNT	0-FFh	RAM wake count delay value. Not recommended to change the default value (1Fh). Bits 23-30: GOOD2ACCESS wake delay. Bits 19-16: ON2GOOD wake delay.
15-12	PDMODE	0-Fh	Power down mode.
		0-Eh	Reserved
		Fh	Core on, RAM array on, RAM periphery on.
11-10	Reserved	0	Reserved
9	EMUIHBIE		Emulation alters power domain state interrupt enable.
		0	Disable interrupt.
		1	Enable interrupt.
8	Reserved	1	Reserved
7-1	Reserved	0	Reserved
0	NEXT		Power domain next state. For Always ON power domain this bit is read/write, but writes have no effect since internally this power domain always remains in the on state.
		0	Power domain off.
		1	Power domain on.

Copyright © 2013–2016, Texas Instruments Incorporated

8.6.14 Power Domain 1 Control Register (PDCTL1)

The power domain 1 control register (PDCTL1) is shown in Figure 8-14 and described in Table 8-19.

Figure 8-14. Power Domain 1 Control Register (PDCTL1)

31	24	23	16
Reserved		WAKECNT	
R-0		R/W-1Fh	

15	12	11	10	9	8	7	1	0
PDMODE		Reserved		EMUIHBIE	Rsvd	Reserved		NEXT
R-Fh		R-0		R/W-0	R-1	R-0		R/W-1

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

Table 8-19. Power Domain 1 Control Register (PDCTL1) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reserved
23-16	WAKECNT	0-FFh	RAM wake count delay value. Not recommended to change the default value (1Fh). Bits 23-30: GOOD2ACCESS wake delay. Bits 19-16: ON2GOOD wake delay.
15-12	PDMODE	0-Fh	Power down mode.
		0	Core off, RAM array off, RAM periphery off.
		1h	Core off, RAM array retention, RAM periphery off (deep sleep).
		2h-3h	Reserved
		4h	Core retention, RAM array off, RAM periphery off.
		5h	Core retention, RAM array retention, RAM periphery off (deep sleep).
		6h-7h	Reserved
		8h	Core on, RAM array off, RAM periphery off.
		9h	Core on, RAM array retention, RAM periphery off (deep sleep).
		Ah	Core on, RAM array retention, RAM periphery off (light sleep).
		Bh	Core on, RAM array retention, RAM periphery on.
		Ch-Eh	Reserved
		Fh	Core on, RAM array on, RAM periphery on.
11-10	Reserved	0	Reserved
9	EMUIHBIE		Emulation alters power domain state interrupt enable.
		0	Disable interrupt.
		1	Enable interrupt.
8	Reserved	1	Reserved
7-1	Reserved	0	Reserved
0	NEXT		User-desired power domain next state.
		0	Power domain off.
		1	Power domain on.

Copyright © 2013–2016, Texas Instruments Incorporated

8.6.15 Power Domain 0 Configuration Register (PDCFG0)

The power domain 0 configuration register (PDCFG0) is shown in Figure 8-15 and described in Table 8-20.

Figure 8-15. Power Domain 0 Configuration Register (PDCFG0)

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		PD_LOCK	ICEPICK	RAM_PSM	ALWAYSON
R-0		R-1	R-1	R-0	R-1

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

Table 8-20. Power Domain 0 Configuration Register (PDCFG0) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	PD_LOCK		PDCTL.NEXT lock. For Always ON power domain this bit is a don't care.
		0	PDCTL.NEXT bit is locked and cannot be changed in software.
		1	PDCTL.NEXT bit is not locked.
2	ICEPICK		IcePick support.
		0	Not present
		1	Present
1	RAM_PSM		RAM power domain.
		0	Not a RAM power domain.
		1	RAM power domain.
0	ALWAYSON		Always ON power domain.
		0	Not an Always ON power domain.
		1	Always ON power domain.

Copyright © 2013–2016, Texas Instruments Incorporated

8.6.16 Power Domain 1 Configuration Register (PDCFG1)

The power domain 1 configuration register (PDCFG1) is shown in Figure 8-16 and described in Table 8-21.

Figure 8-16. Power Domain 1 Configuration Register (PDCFG1)

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		PD_LOCK	ICEPICK	RAM_PSM	ALWAYSON
R-0		R-1	R-1	R-0	R-1

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

Table 8-21. Power Domain 1 Configuration Register (PDCFG1) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	PD_LOCK		PDCTL.NEXT lock. For Always ON power domain this bit is a don't care.
		0	PDCTL.NEXT bit is locked and cannot be changed in software.
		1	PDCTL.NEXT bit is not locked.
2	ICEPICK		IcePick support.
		0	Not present
		1	Present
1	RAM_PSM		RAM power domain.
		0	Not a RAM power domain.
		1	RAM power domain.
0	ALWAYSON		Always ON power domain.
		0	Not an Always ON power domain.
		1	Always ON power domain.

Copyright © 2013–2016, Texas Instruments Incorporated

8.6.17 Module Status *n* Register (MDSTAT*n*)

The module status *n* register (MDSTAT*n*) is shown in Figure 8-17 and described in Table 8-22.

Figure 8-17. Module Status *n* Register (MDSTAT*n*)

31	18	17	16
Reserved		EMUIHB	EMURST
R-0		R-0	R-0

15	13	12	11	10	9	8	7	6	5	0
Reserved		MCKOUT	Rsvd	MRST	LRSTDONE	LRST	Reserved		STATE	
R-0		R-0	R-1	R-0	R-1	R-1	R-0		R-0	

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

Table 8-22. Module Status *n* Register (MDSTAT*n*) Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17	EMUIHB		Emulation alters module state. This bit applies to ARM module (module 14). This field is 0 for all other modules.
		0	No emulation altering user-desired module state programmed in the NEXT bit in the module control 14 register (MDCTL14).
		1	Emulation altered user-desired state programmed in the NEXT bit in MDCTL14. If you desire to generate a PSCINT upon this event, you must set the EMUIHBIE bit in MDCTL14.
16	EMURST		Emulation alters module reset. This bit applies to ARM module (module 14). This field is 0 for all other modules.
		0	No emulation altering user-desired module reset state.
		1	Emulation altered user-desired module reset state. If you desire to generate a PSCINT upon this event, you must set the EMURSTIE bit in the module control 14 register (MDCTL14).
15-13	Reserved	0	Reserved
12	MCKOUT		Module clock output status. Shows status of module clock.
		0	Module clock is off.
		1	Module clock is on.
11	Reserved	1	Reserved
10	MRST		Module reset status. Reflects actual state of module reset.
		0	Module reset is asserted.
		1	Module reset is de-asserted.
9	LRSTDONE		Local reset done. Software is responsible for checking if local reset is done before accessing this module. This bit applies to ARM module (module 14). This field is 1 for all other modules.
		0	Local reset is not done.
		1	Local reset is done.
8	LRST		Module local reset status. This bit applies to ARM module (module 14).
		0	Local reset is asserted.
		1	Local reset is de-asserted.
7-6	Reserved	0	Reserved
5-0	STATE	0-3Fh	Module state status: indicates current module status.
		0	SwRstDisable state
		1h	SyncReset state
		2h	Disable state
		3h	Enable state
		4h-3Fh	Indicates transition

Copyright © 2013–2016, Texas Instruments Incorporated

8.6.18 PSC0 Module Control *n* Register (modules 0-15) (MDCTL*n*)

The PSC0 module control *n* register (MDCTL*n*) is shown in Figure 8-18 and described in Table 8-23.

Figure 8-18. PSC0 Module Control *n* Register (MDCTL*n*)

31	30	16
FORCE	Reserved	
R/W-0	R-0	

15	11	10	9	8	7	3	2	0
Reserved		EMUIHBIE	EMURSTIE	LRST	Reserved		NEXT	
R-0		R/W-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 8-23. PSC0 Module Control *n* Register (MDCTL*n*) Field Descriptions

Bit	Field	Value	Description
31	FORCE		Force enable. This bit forces the module state programmed in the NEXT bit in the module control 14 register (MDCTL14), ignoring and bypassing all the clock stop request handshakes managed by the PSC to change the state of the clocks to the module.
			Note: It is **not** recommended to use the FORCE bit to disable the module clock, unless specified.
		0	Force is disabled.
		1	Force is enabled.
30-11	Reserved	0	Reserved
10	EMUIHBIE		Interrupt enable for emulation alters module state. This bit applies to ARM module (module 14).
		0	Disable interrupt.
		1	Enable interrupt.
9	EMURSTIE		Interrupt enable for emulation alters reset. This bit applies to ARM module (module 14).
		0	Disable interrupt.
		1	Enable interrupt.
8	LRST		Module local reset control. This bit applies to ARM module (module 14).
		0	Assert local reset
		1	De-assert local reset
7-3	Reserved	0	Reserved
2-0	NEXT	0-3h	Module next state.
		0	SwRstDisable state
		1h	SyncReset state
		2h	Disable state
		3h	Enable state

Copyright © 2013–2016, Texas Instruments Incorporated

8.6.19 PSC1 Module Control *n* Register (modules 0-31) (MDCTL*n*)

The PSC1 module control *n* register (MDCTL*n*) is shown in Figure 8-19 and described in Table 8-24.

Figure 8-19. PSC1 Module Control *n* Register (MDCTL*n*)

31	30	16
FORCE	Reserved	
R/W-0	R-0	

15	3	2	0
Reserved		NEXT	
R-0		R/W-0	

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

Table 8-24. PSC1 Module Control *n* Register (MDCTL*n*) Field Descriptions

Bit	Field	Value	Description
31	FORCE		Force enable. This bit forces the module state programmed in the NEXT bit in the , ignoring and bypassing all the clock stop request handshakes managed by the PSC to change the state of the clocks to the module.
			Note: It is **not** recommended to use the FORCE bit to disable the module clock, unless specified.
		0	Force is disabled.
		1	Force is enabled.
30-3	Reserved	0	Reserved
2-0	NEXT	0-3h	Module next state.
		0	SwRstDisable state
		1h	SyncReset state
		2h	Disable state
		3h	Enable state

Copyright © 2013–2016, Texas Instruments Incorporated

Power Management

Topic **Page**

9.1	Introduction	179
9.2	Power Consumption Overview	179
9.3	PSC and PLLC Overview	179
9.4	Features	180
9.5	Clock Management	181
9.6	ARM Sleep Mode Management	182
9.7	RTC-Only Mode	184
9.8	Dynamic Voltage and Frequency Scaling (DVFS)	184
9.9	Deep Sleep Mode	185
9.10	Additional Peripheral Power Management Considerations	189

Copyright © 2013–2016, Texas Instruments Incorporated

9.1 Introduction

Power management is an important aspect for most embedded applications. For several applications and target markets, there may be a specific power budget and requirements to minimize power consumption for both power supply sizing and battery life considerations. Additionally, lower power consumption results in more optimal and efficient designs from cost, design, and energy perspectives. This device has several means of managing the power consumption. This chapter discusses the various power management features.

9.2 Power Consumption Overview

Power consumed by semiconductor devices has two components: dynamic and static. This can be shown as:

$Ptotal = P_{dynamic} + P_{static}$

The dynamic power is the power consumed to perform work when the device is in active modes (clocks applied, busses, and I/O switching), that is, analog circuits changing states. The dynamic power is defined by:

$P_{dynamic} = Capacitance \times Voltage^2 \times Frequency$

From the above formula, the dynamic power scales with the clock frequency (device/module frequency for core operations and switching frequency for I/O). Dynamic power can be reduced by controlling the clocks in such a way as to either operate at a clock setting just high enough to complete the required operation in the required timeline or to run at a clock setting until the work is complete and then drastically reduce the clock frequency or cut off the clocks until additional work must be performed.

In the formula, the dynamic power varies with the voltage squared, so the voltage of operations has significant impact on overall power consumption and, thus, on the battery life. Dynamic power can be reduced by scaling the operating voltage, when the performance requirements are not that high and the device can be operated at a corresponding lower frequency.

The capacitance is the capacitance of the switching nodes, or the load capacitances on the switching I/O pins.

The static power, as the name suggests, is independent of the switching frequency of the logic. It can be shown as:

$P_{static} = f_{(leakage\ current)}$

It is essentially a function of the “leakage”, or the power consumed by the logic when it is not switching or is not performing any work. Leakage current is dependent mostly on the manufacturing process used, the size of the die, etc. Leakage current is unavoidable while power is applied and scales roughly with the operating junction temperatures. Leakage power can only be avoided by removing power completely from a device or subsystem. The static power consumption plays a significant role in the Standby Modes (when the application is not running and in a dormant state) and plays an important role in the battery life for portable applications, etc.

9.3 PSC and PLLC Overview

The power and sleep controller (PSC) module plays an important role in managing the enabling/disabling of the clocks to the core and various peripheral modules. The PSC provides a granular support to turn on/off clocks on a module by module basis. Similarly, the two PLL controllers (PLLC0 and PLLC1) play an important role in device and module clock generation, and manage the frequency scaling operations for the device. Together these modules play a significant role in managing the clocks from a power management feature standpoint. For detailed information on the PSC, see the *Power and Sleep Controller (PSC)* chapter. For detailed information on the PLLC0 and PLLC1, see the *Device Clocking* chapter and the *Phase-Locked Loop Controller (PLLC)* chapter.

Copyright © 2013–2016, Texas Instruments Incorporated

9.4 Features

This device has several means of managing power consumption, as detailed in the subsequent sections. This device uses the state-of-the-art 65 nm process, which provides a good balance on power and performance, providing high-performance transistors with relatively less leakage current and, thereby, low standby-power consumption modes.

There are several features in design as well as user driven software control to reduce dynamic power consumption. The design features (not under user control) include a power optimized clock tree design to reduce overall clock tree power consumption and automatic clock gating in several modules when the logic in the modules is not active.

The on-chip power and sleep controller (PSC) module provides granular software controlled module level clock gating, which reduces both clock tree and module power by basically disabling the clocks when the modules are not being used. Clock management also allows you to slow down the clocks, to reduce the dynamic power.

Table 9-1 describes the power management features.

Table 9-1. Power Management Features

Power Management	Description	Features
	Clock Management	
PLL bypass and power-down	Both PLLs can be powered-down and run in bypass mode when not in use.	Reduces the dynamic power consumption of the core.
Module clock ON	Module clocks can be turned on/off without requiring reconfiguring the registers.	Reduces the dynamic power consumption of the core and I/O (if any free running I/O clocks).
	Core Sleep Management	
ARM subsystem sleep modes	The ARM CPU can be put in sleep mode. Additionally, the ARM subsystem clock can be completely gated when not in use.	Reduces the dynamic power consumption.
	Voltage Management	
RTC-only mode	Allows removing power from all core and I/O supply and just have the real-time clock (RTC) running.	Reduces the dynamic and static power for standby modes that require only the RTC to be functional.
	Dynamic Voltage and Frequency Scaling	
Dynamic Voltage and Frequency Scaling (DVFS)	The operating voltage and frequency of the device can be dynamically scaled to meet the requirements of the application.	Reduces the dynamic power consumption of the core and I/O as well as standby power
	System/Device Sleep Management	
Deep Sleep Mode	All internal clocks of the device can be turned on/off at the OSCIN level. The deep sleep function can be controlled externally through the $\overline{\text{DEEPSLEEP}}$ pin or internally through the RTC_ALARM pin.	Reduces the dynamic power consumption of the core and I/O.
	Peripheral I/O Power Management	
USB PHY power-down	The USB2.0 PHY can be powered-down.	Minimizes the USB2.0 I/O power consumption when not in use.
DDR2/mDDR self-refresh mode	Allows memory to retain its contents while the rest of the system is powered down.	mDDR and DDR2 can be clock gated to reduce the dynamic power consumption or the entire device can be powered down to reduce the static power consumption.
LVCMOS I/O buffer receiver disable	LVCMOS I/O buffer receivers are disabled.	Minimizes the I/O power consumption.
Internal pull-up and pull-down resistor control	The internal pull-ups and pull-downs are enabled/disabled by groups.	Reduces the I/O leakage power.

Copyright © 2013–2016, Texas Instruments Incorporated

9.5 Clock Management

9.5.1 Module Clock ON/OFF

The module clock on/off feature allows software to disable clocks to module individually, in order to reduce the module's dynamic/switching power consumption down to zero. This device is designed in full static CMOS; thus, when a module clock stops, the module's state is preserved and retained. When the clock is restarted, the module resumes operating from the stopping point.

NOTE: Stopping clocks to a module only affects dynamic power consumption, it does not affect static power consumption of the module or the device.

The power and sleep controller (PSC) module controls module clock gating. If a module's clock(s) is stopped while being accessed, the access may not occur, and it can potentially result in unexpected behavior. The PSC provides some protection against such erroneous conditions by monitoring the internal bus activity to ensure there are no accesses to the module from the internal bus, before allowing module's internal clock to be gated. However, it is still recommended that software must ensure that all of the transactions to the module are finished prior to disabling the clocks.

The procedure to turn module clocks on/off using the PSC is described in the *Power and Sleep Controller (PSC)* chapter.

NOTE: To preserve the state of the module, the module state in the PSC must be set to Disable. In this state, the module reset is not asserted and only the module clock is turned off.

Additionally some peripherals implement additional power saving features by automatically shutting of clock to components within the module, when the logic is not active. This is transparent to you, but reduces overall dynamic power consumption when modules are not active.

9.5.2 Module Clock Frequency Scaling

Module clock frequency is scalable by programming the PLL multiply and divide parameters. Additionally, some modules might also have internal clock dividers. Reducing the clock frequency reduces the dynamic/switching power consumption, which scales linearly with frequency.

The *Device Clocking* chapter details the clocking structure of the device. The *Phase-Locked Loop Controller (PLLC)* chapter describes how to program the PLL0 and PLL1 frequency and the frequency constraints.

9.5.3 PLL Bypass and Power Down

You can bypass each PLL in this device. Bypassing the PLL sends a bypass clock instead of the PLL VCO output (PLLOUT) to the system clocks of the PLLC. For PLLC0, the bypass clock is selected from either the PLL reference clock (OSCIN) or PLL1_SYSCLK3. For PLLC1, the bypass clock is always OSCIN. The OSCIN frequency is typically, at most, up to 50 MHz.

You can use the OSCIN bypass mode to reduce the core and module clock frequencies to very low maintenance levels without using the PLL during periods of very low system activity. This can lower the overall dynamic power consumption, which is linearly proportional to the frequency.

When the PLL controller is placed in bypass mode, the PLL retains its frequency lock. This allows you to switch between bypass mode and PLL mode without having to wait for the PLL to relock. However, keeping the PLL locked consumes power. You can also power-down the PLL when bypassing it to minimize the overall power consumed by the PLL module. The advantage of bypassing the PLL without powering it down is that you do not have to incur the PLL lock time when switching back to a normal operating level.

The *Device Clocking* chapter and the *Phase-Locked Loop Controller (PLLC)* chapter describe PLL bypass and PLL power down.

Copyright © 2013–2016, Texas Instruments Incorporated

9.6 ARM Sleep Mode Management

9.6.1 ARM Wait-For-Interrupt Sleep Mode

The ARM module can be put into a low-power state using a special sleep mode called wait-for-interrupt (WFI). When the wait-for-interrupt mode is enabled, all internal clocks within the ARM9 module are shut off, the core is completely inactive and only resumes operation after receiving an interrupt. This is a feature for dynamic power management of the ARM processor itself, it does not impact the static power.

NOTE: To enable the WFI mode, the ARM needs to be in supervisor mode.

You can enable the WFI mode via the CP15 register #7 using the following instruction:

- MCR p15, #0, <Rd>, c7, c0, #4

Once the ARM module transitions into the WFI mode, it will remain in this state until an interrupt request (IRQ/FIQ) occurs.

The following sequence exemplifies how to enter the WFI mode:

- Enable any interrupt (for example, an external interrupt) that you plan to use as the wake-up interrupt to exit from the WFI mode.
- Enable the WFI mode using the following CP15 instruction:
  - MCR p15, #0, r3, c7, c0, #4

The following sequence describes the procedure to wake-up from the WFI mode:

- To wake-up from the WFI mode, trigger any enabled interrupt (for example, an external interrupt).
- The ARM's PC jumps to the IRQ/FIQ vector and you must handle the interrupt in an interrupt service routine (ISR).

Exit the ISR and continue normal program execution starting from the instruction immediately following the instruction that enabled the WFI mode.

NOTE: The ARM interrupt controller (AINTC) and the module sourcing the wake-up interrupt (for example, GPIO or watchdog timer) must not be disabled, or the device will never wake up.

For more information on this sleep mode, see the ARM926EJ-S Technical Reference Manual (TRM), downloadable from http://infocenter.arm.com/help/index.jsp.

Copyright © 2013–2016, Texas Instruments Incorporated

9.6.2 ARM Clock OFF

The software must be structured such that no peripheral is allowed to access the ARM resources before disabling the clocks to the ARM subsystem. The ARM must check for the completion of all its master peripheral initiated requests (that is, CFG and DMA port operations, etc.).

ARM module clock off sequence:

1. The ARM must have the ARM Clock Stop Request interrupt (ARMCLKSTOPREQ, ARM interrupt # 90) enabled and the associated interrupt service routine (ISR) set up before the following ARM clock shutdown procedure.
   (a) Initiate the ARM clock off sequence by issuing the ARM clock stop command (PSC DISABLE Command) to the ARM subsystem by writing a 2h to the NEXT bit field in the ARM local power sleep controller (LPSC) module control register (PSC0.MDCTL14).
   (b) Write a 1 to the GO[0] bit (ARM subsystem is part of the PD_ALWAYSON domain) in the power domain transition command register (PSC0.PTCMD) to start the state transition sequence for the ARM module. This generates the ARMCLKSTOPREQ interrupt to the ARM.
   (c) Check (poll for 0) the GOSTAT[0] bit in the power domain transition status register (PSC0.PTSTAT) for power transition sequence completion. The GOSTAT[0] bit transitions to 0 when the ARM executes the wait-for-interrupt instruction from inside its interrupt service routine (ISR).
   (d) Check (poll for 2h) the STATE bit field in the ARM LPSC module status register (PSC0.MDSTAT14) indicating the ARM clock stop sequence completion (STATE: Disable).

The following sequence should be executed by the ARM within the ARM Clock Stop Request interrupt ISR:

1. Check for completion of all ARM master requests (the ARM polls transfer completion statuses of all Master peripherals).
2. Enable the interrupt to be used as the “wake-up” interrupt (for example, one of the CHIPSIG interrupts controlled by the chip signal register (CHIPSIG) in the *System Configuration (SYSCFG) Module* chapter—CHIPSIG[0], CHIPSIG[1], etc.) that will be used to wake-up the ARM during the ARM clock-on sequence.
3. Execute the wait-for-interrupt (WFI) ARM instruction.

9.6.3 ARM Subsystem Clock ON

The ARM module defaults to the SwRstDisable state; therefore, the software is responsible for enabling the clock and releasing the reset to the ARM at power-on reset.

1. Wait for the GOSTAT[0] bit in the power domain transition status register (PSC0.PTSTAT) to clear to 0. You must wait for the power domain to finish any previously initiated transitions before initiating a new transition.
2. Write a 3h to the NEXT bit in the ARM local power sleep controller (LPSC) module control register (PSC0.MDCTL14) to prepare the ARM module for an enable transition.
3. Write a 1 to the GO[0] bit (ARM subsystem is part of the PD_ALWAYSON domain) in the power domain transition command register (PSC0.PTCMD) to start the state transition sequence for the ARM module.
4. Check (poll for 0) the GOSTAT[0] bit in PSC0.PTSTAT for power transition sequence completion. The domain is only safely in the new state after the GOSTAT[0] bit is cleared to 0.
5. Wait for the STATE bit field in the ARM LPSC module status register (PSC0.MDSTAT14) to change to 3h. The module is only safely in the new state after the STATE bit field changes to reflect the new state.

NOTE: This only applies if you are transitioning from the Disable state. If previously in the Disable state, a wake-up interrupt must be triggered in order to wake the ARM (to exit the wait-for-interrupt mode). This example assumes that the ARM enabled this interrupt before entering its wait-for-interrupt sleep mode state.

Copyright © 2013–2016, Texas Instruments Incorporated

9.7 RTC-Only Mode

In real-time clock (RTC)-only mode, the RTC is powered on and the rest of the device is completely powered off (all supplies except the RTC supply are removed). In this mode, the RTC is fully functional and keeps track of date, hours, minutes, and seconds. In this mode, the overall power consumption would be significantly lower, as voltage from the rest of the core and I/O logic can be completely removed, eliminating most of the active and static power of the device, except for what is consumed by the RTC module, running at 32 kHz.

NOTE: To put the device in RTC-only mode, there is no software control sequence. You can put the device in the RTC-only mode by removing the power supply from all core and I/O logic, except for the RTC core logic supply (RTC_CVDD). During wake up, all power sequencing requirements described in the device-specific data manual must be followed.

Some limitations apply in the RTC-only mode. First, the RTC_ALARM pin is not available as an option for use as a control to signal an external power supply to reapply power to the rest of the device. This is because the RTC_ALARM pin is powered by the I/O supply that is powered down in RTC-only mode. Second, in RTC-only mode, only the RTC register contents are preserved, all other internal memory and register contents are lost. Mobile DDR and DDR2 contents can be preserved through the use of self-refresh (see Section 9.9.2). However, software must be in place to restore the context of the device, for example, reinitialize internal registers, setup cache memory configurations, interrupt vectors, etc.

9.8 Dynamic Voltage and Frequency Scaling (DVFS)

Dynamic voltage and frequency scaling (DVFS) consists of minimizing the idle time of the system. The DVFS technique uses dynamic selection of the optimal frequency and voltage to allow a task to be performed in the required amount of time. This reduces the total power consumption of the device while still meeting task requirements. DVFS requires control over the clock frequency and the operating voltage of the device elements. By intelligently switching these elements to their optimal operating points, it is possible to minimize the power consumption of the device for a given task.

For reasons related to the device (clock architecture, process, etc.), DVFS is used only for a few discrete steps, not over a continuum of voltage and frequency values. Each step, or operating performance point (OPP), is composed of a voltage and frequency pair. For an OPP, the frequency corresponds to the maximum frequency allowed at a voltage, or reciprocally; the voltage corresponds to the minimum voltage allowed for a frequency. See your device data manual for a list of the OPPs supported by the device.

When applying DVFS, a processor or system always runs at the lowest OPP that meets the performance requirement at a given time. You determine the optimal OPP for a given task and then switch to that OPP to save power.

9.8.1 Frequency Scaling Considerations

The operating frequency of the device is controlled through its two PLL controllers (PLLC0 and PLLC1). Through a series of multipliers and dividers you can change the frequencies of various clocks throughout the device. See the *Device Clocking* chapter for information on the clock architecture of the device and see the *Phase-Locked Loop Controller (PLLC)* chapter for information on the PLL controllers. A few things must be noted when changing the various internal frequencies of the device:

- Changing the SYSCLK frequency

  The PLL_VCO (PLLOUT) frequency can be programmed through a PLL multiplier. A series of dividers divide PLLOUT to generate the various device SYSCLKs.

  To change the SYSCLK frequency you can change the PLL multiplier or you can change the SYSCLK divider ratio. When changing the PLL multiplier, you must put the PLL controller in bypass mode while the PLL multiplier value is modified and a lock on the new frequency is reached. The lock time is given in the device data manual. When changing the divider ratios it is not required to put the PLL controller in bypass mode.

  Changing the SYSCLK frequency through the dividers is faster as there is no need to reprogram the PLL. However, the SYSCLK frequency will depend solely on the divider ratios used.

Copyright © 2013–2016, Texas Instruments Incorporated

- SYSCLK domain fixed ratios

  Certain SYSCLK domains need to operate at a fixed ratio with respect to the ARM clock. Care should be taken to ensure that these fixed ratios are maintained. For additional details, see the *Device Clocking* chapter.

- PLLC0 bypass clock

  When switching the PLL multiplier, the PLL controller must be placed in bypass mode. Bypassing the PLL sends a bypass clock instead of the PLL VCO output (PLLOUT) to the system clock dividers of the PLL controller.

  For PLLC0 the bypass clock is selected from either the PLL reference clock (OSCIN) or PLL1_SYSCLK3. For PLLC1, the bypass clock is always OSCIN. The OSCIN frequency is typically, at most, up to 50 MHZ.

  You can use the OSCIN bypass mode to reduce the core and module clock frequencies to very low maintenance levels without using the PLL during periods of very low system activity.

  It may be desirable for the bypass clock to not revert to OSCIN in some situations to preserved bandwidth during frequency scaling transitions. For this reason, the PLLC0 bypass clock can be set to PLL1_SYSCLK3. This selection is made through the EXTCLKSRC bit in the PLLCTL register of PLLC0.

- Peripheral immunity from ARM clock frequency changes

  Peripherals that are clocked by the PLL0_AUXCLK are immune to changes in the PLL0 frequency. The PLL0_AUXCLK is derived from OSCIN.

  Peripherals in the ASYNC3 domain are clocked off from either PLL1_SYSCLK2 or PLL0_SYSCLK2. Furthermore, PLL0_SYSCLK2 must always be /2 of the ARM clock frequency. To keep these peripherals immune from changes in PLL0 frequency (such as when the ARM frequency is modified), you can configure the ASYNC3 domain to be clocked from PLL1_SYSCLK2. PLL1 is mainly used to clock the DDR2/mDDR memory controller.

  When peripherals are immune to changes in the ARM clock frequency, their internal clock dividers do not have to be adjusted for changes in their input clock frequencies.

9.8.2 Voltage Scaling Considerations

The operating voltage of the device must be totally controlled through mechanisms outside the device. I2C ports on the device can be used to communicate with external power management chips. A few things must be noted when changing the operating voltage of the device:

- Voltage ramp rate: The ramp rate of the operating voltage must be observed during operating performance point (OPP) transitions. See the device data manual for ramp rate specifications.
- Switching to a lower voltage: When switching to a lower voltage, the maximum operating frequency changes. Care must be taken such that the maximum operating frequency supported at the new voltage is not violated. For this reason, it is recommended to change the operating frequency before switching the operating voltage.

9.9 Deep Sleep Mode

This device supports a Deep Sleep mode where all device clocks are stopped and the on-chip oscillator is shut down to save power. Registers and memory contents are preserved, thus, upon recovery, the program may continue from where it left off with minimal overhead involved.

The Deep Sleep mode is initiated when the $\overline{\text{DEEPSLEEP}}$ pin is driven low. The device wakes up from Deep Sleep mode when the $\overline{\text{DEEPSLEEP}}$ pin is driven high. The $\overline{\text{DEEPSLEEP}}$ pin can be driven by an external controller or it can be driven internally by the real-time clock (RTC). The RTC method allows for automatic wake-up at a programmed time.

NOTE: Due to pin multiplexing, the $\overline{\text{DEEPSLEEP}}$ pin can only be driven by an external controller or its internal real-time clock (RTC). The $\overline{\text{DEEPSLEEP}}$ pin cannot be driven by both an external controller and its internal real-time clock at the same time.

Copyright © 2013–2016, Texas Instruments Incorporated

9.9.1 Entering/Exiting Deep Sleep Mode Using Externally Controlled Wake-Up

9.9.1.1 Entering Deep Sleep Mode

Use the following procedure to enter the Deep Sleep mode if an external signal is used to wake-up the device:

1. To preserve DDR2/mDDR memory contents, activate the self-refresh mode and gate the clocks to the DDR2/mDDR memory controller. You can use partial array self-refresh (PASR) for additional power savings for mDDR memory.
2. The USB2.0 (USB0) PHY should be disabled, if this interface is used and internal clocks are selected (see Section 9.10.1).
3. PLL/PLLC0 and PLL/PLLC1 should be placed in bypass mode (clear the PLLEN bit in the PLL control register (PLLCTL) of each PLLC to 0).
4. PLL/PLLC0 and PLL/PLLC1 should be powered down (set the PLLPWRDN bit in PLLCTL of each PLLC to 1).
5. Configure the $\overline{\text{DEEPSLEEP}}$ pin as input-only using the PINMUX0_31_28 bits in the PINMUX0 register in the *System Configuration (SYSCFG) Module* chapter.
6. The external controller should drive the $\overline{\text{DEEPSLEEP}}$ pin high (not in Deep Sleep).
7. Configure the desired delay in the SLEEPCOUNT bit field in the deep sleep register (DEEPSLEEP) in the *System Configuration (SYSCFG) Module* chapter. This count determines the delay before the Deep Sleep logic releases the clocks to the device during wake up (allowing the oscillator to stabilize).
8. Set the SLEEPENABLE bit in DEEPSLEEP to 1. This automatically clears the SLEEPCOMPLETE bit.
9. Begin polling the SLEEPCOMPLETE bit until it is set to 1. This bit is set once the device is woken up from Deep Sleep mode.
10. The external controller drives the $\overline{\text{DEEPSLEEP}}$ pin low to initiate Deep Sleep mode.

For more details on the clock stop procedure of the DDR2/mDDR memory controller, see the *DDR2/mDDR Memory Controller* chapter.

9.9.1.2 Exiting Deep Sleep Mode

Use the following procedure to exit the Deep Sleep state if an external signal is used to wake-up the device:

1. The external controller drives the $\overline{\text{DEEPSLEEP}}$ pin high.
2. When the SLEEPCOUNT delay is complete, the Deep Sleep logic releases the clock to the device and sets the SLEEPCOMPLETE bit in the deep sleep register (DEEPSLEEP) in the *System Configuration (SYSCFG) Module* chapter.
3. Clear the SLEEPENABLE bit in DEEPSLEEP to 0. This automatically clears the SLEEPCOMPLETE bit.
4. Initialize the PLL controllers as described in Section 7.2.2.2. Note that the state of the PLL controller registers is preserved during Deep Sleep mode. Therefore, it is not necessary to reprogram all the PLL controller registers unless a new setting is desired. At minimum, steps 3, 4, and 7-10 of the PLL initialization procedure must be followed.
5. Enable the clocks to the DDR2/mDDR memory controller, reset the DDR PHY, and then take the DDR2/mDDR out of self-refresh mode.
6. Configure the desired states to the peripherals and enable as required.

For more details on the clock enable procedure of the DDR2/mDDR memory controller, see the *DDR2/mDDR Memory Controller* chapter.

Copyright © 2013–2016, Texas Instruments Incorporated

9.9.2 Entering/Exiting Deep Sleep Mode Using RTC Controlled Wake-Up

9.9.2.1 Entering Deep Sleep Mode

Use the following procedure to enter the Deep Sleep state if the RTC is used to wake-up the device:

1. To preserve DDR2/mDDR memory contents, activate the self-refresh mode and gate the clocks to the DDR2/mDDR memory controller. You can use partial array self-refresh (PASR) for additional power savings for mDDR memory.
2. The USB2.0 (USB0) PHY should be disabled, if this interface is used and internal clocks are selected (see Section 9.10.1).
3. PLL/PLLC0 and PLL/PLLC1 should be placed in bypass mode (clear the PLLEN bit in the PLL control register (PLLCTL) of each PLLC to 0).
4. PLL/PLLC0 and PLL/PLLC1 should be powered down (set the PLLPWRDN bit in PLLCTL of each PLLC to 1).
5. Configure the desired wake-up time as an alarm in the RTC.
6. Configure the $\overline{\text{DEEPSLEEP}}$/RTC_ALARM pin to output RTC_ALARM using the PINMUX0_31_28 bits in the PINMUX0 register in the *System Configuration (SYSCFG) Module* chapter. The pin is driven low since the alarm has not yet occurred.
7. Configure the desired delay in the SLEEPCOUNT bit field in the deep sleep register (DEEPSLEEP) in the *System Configuration (SYSCFG) Module* chapter. This count determines the delay before the Deep Sleep logic releases the clocks to the device during wake up (allowing the oscillator to stabilize).
8. Set the SLEEPENABLE bit in DEEPSLEEP to 1. This automatically clears the SLEEPCOMPLETE bit. Also, the device now enters the Deep Sleep mode since the $\overline{\text{DEEPSLEEP}}$ pin is low.

For more details on the clock stop procedure of the DDR2/mDDR memory controller, see the *DDR2/mDDR Memory Controller* chapter.

9.9.2.2 Exiting Deep Sleep Mode

Use the following procedure to exit the Deep Sleep state if the RTC is used to wake-up the device:

1. The RTC alarm occurs and the RTC_ALARM pin is driven high (which is internally connected to the $\overline{\text{DEEPSLEEP}}$ pin). This causes the Deep Sleep logic to exit the Deep Sleep mode.
2. When the SLEEPCOUNT delay is complete, the Deep Sleep logic releases the clock to the device and sets the SLEEPCOMPLETE bit in the deep sleep register (DEEPSLEEP) in the *System Configuration (SYSCFG) Module* chapter.
3. Clear the SLEEPENABLE bit in DEEPSLEEP to 0. This automatically clears the SLEEPCOMPLETE bit.
4. Initialize the PLL controllers as described in Section 7.2.2.2. Note that the state of the PLL controller registers is preserved during Deep Sleep mode. Therefore, it is not necessary to reprogram all the PLL controller registers unless a new setting is desired. At minimum, steps 3, 4, and 7-10 of the PLL initialization procedure must be followed.
5. Enable the clocks to the DDR2/mDDR memory controller, reset the DDR PHY, and then take the DDR2/mDDR out of self-refresh mode.
6. Configure the desired states to the peripherals and enable as required.

For more details on the clock enable procedure of the DDR2/mDDR memory controller, see the *DDR2/mDDR Memory Controller* chapter.

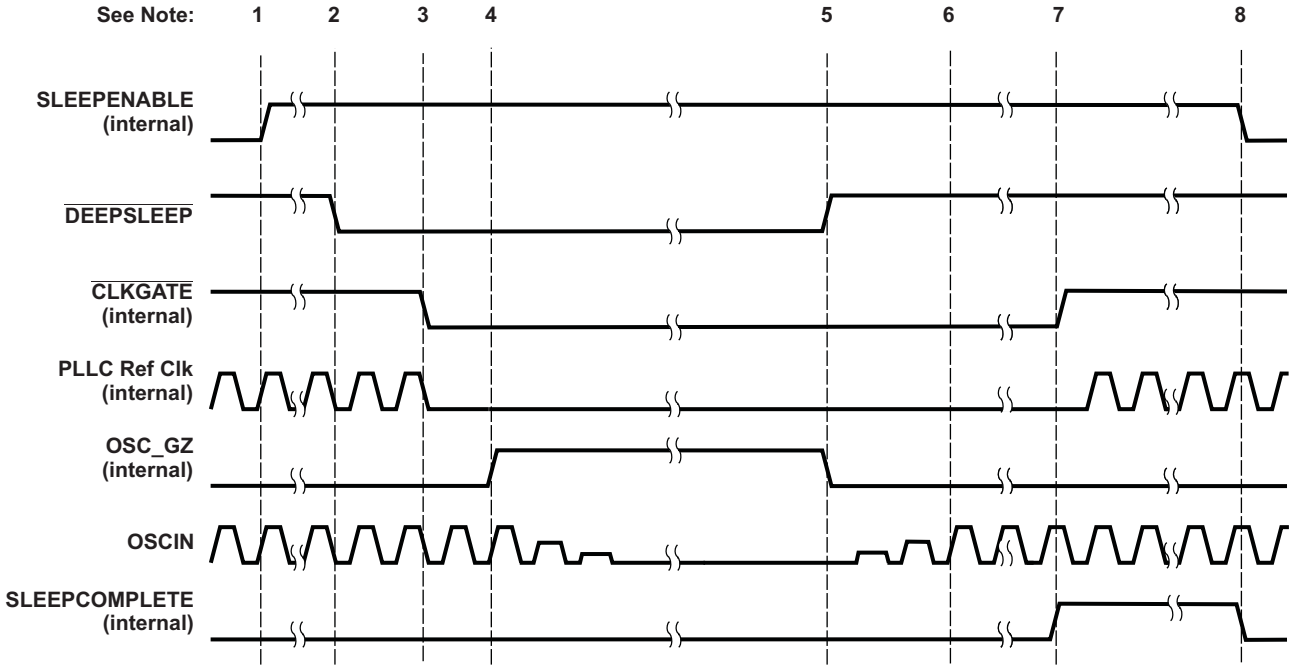
Copyright © 2013–2016, Texas Instruments Incorporated

9.9.3 Deep Sleep Sequence

Figure 9-1 illustrates the Deep Sleep sequence:

1. Software sets the SLEEPENABLE bit in the deep sleep register (DEEPSLEEP) in the *System Configuration (SYSCFG) Module* chapter.
2. The $\overline{\text{DEEPSLEEP}}$ pin is driven low by either an external device or the RTC_ALARM pin. The Deep Sleep mode begins.
3. The PLL controller reference clock is gated.
4. The on-chip oscillator is disabled. If the device is being clocked by an external source, this clock may stay enabled; the power savings from turning off this clock is minimal.
5. The $\overline{\text{DEEPSLEEP}}$ pin is driven high and the on-chip oscillator is enabled.
6. The Deep Sleep counter beings counting valid clock cycles.
7. The count has reached the number specified in the SLEEPCOUNT bit field and the SLEEPCOMPLETE bit is set. The PLL reference clock is enabled and the Deep Sleep mode ends.
8. Software clears the SLEEPENABLE bit. The SLEEPCOMPLETE bit is automatically cleared.

Figure 9-1. Deep Sleep Mode Sequence



Copyright © 2013–2016, Texas Instruments Incorporated

9.9.4 Entering/Exiting Deep Sleep Mode Using Software Handshaking

Entering the Deep Sleep mode stops all of the clocks to the device so it is the responsibility of the software to ensure that all peripheral accesses have been completed and peripheral interfaces appropriately configured for clocks to stop. Therefore, before an external controller drives the DEESPLEEP pin, a handshaking mechanism must be in place to give software time to prepare the device for Deep Sleep mode. The implementation of the handshake mechanism is up to the system designer.

9.9.4.1 Entering Deep Sleep Mode

The following example sequence can be used to activate the Deep Sleep mode using a handshaking mechanism between your device and an external device:

1. Clear the SLEEPENABLE bit in the deep sleep register (DEEPSLEEP) in the *System Configuration (SYSCFG) Module* chapter to 0. The DEEPSLEEP pin has no effect until software running on the device sets this bit.
2. Configure the GP0[8]/DEEPSLEEP/RTC_ALARM pin to output GP0[8] using the PINMUX0_31_28 bits in the PINMUX0 register in the *System Configuration (SYSCFG) Module* chapter. When the pin is configured for GPIO functionality, the internal DEEPSLEEP signal is still driven by the value on the pin.
3. Configure the GP0[8] pin to generate interrupts on the falling edge of the GPIO signal.
4. An external device drives the GP0[8] pin low.
5. Software prepares the device for Deep Sleep mode.
6. Set the SLEEPENABLE bit in DEEPSLEEP to 1. The Deep Sleep mode is immediately started and all device clocks are stopped. Also, the SLEEPCOMPLETE bit is automatically cleared.

9.9.4.2 Exiting Deep Sleep Mode

To exit the Deep Sleep mode, follow this sequence:

1. An external device drives the GP0[8] pin high.
2. The device exits the Deep Sleep mode. When the SLEEPCOUNT delay is complete, the Deep Sleep logic releases the clock to the device and sets the SLEEPCOMPLETE bit in the deep sleep register (DEEPSLEEP) in the *System Configuration (SYSCFG) Module* chapter.
3. Clear the SLEEPENABLE bit in DEEPSLEEP to 0.

9.10 Additional Peripheral Power Management Considerations

This section lists additional power management features and considerations that might be part of other chip-level or peripheral logic, apart from the features supported by the core, PLL controller (PLLC), and power and sleep controller (PSC).

9.10.1 USB PHY Power Down Control

The USB modules can be clock gated using the PSC; however, this does not power down/clock gate the PHY logic. You can put the USB2.0 PHY and OTG module in the lowest power state, when not in use, by writing to the USB0PHYPWDN and the USB0OTGPWRDN bits in the Chip Configuration 2 Register (CFGCHIP2) in the *System Configuration (SYSCFG) Module* chapter.

Copyright © 2013–2016, Texas Instruments Incorporated

9.10.2 DDR2/mDDR Memory Controller Clock Gating and Self-Refresh Mode

The DDR2/mDDR memory controller supports different methods for reducing its power consumption including self-refresh mode, power-down mode, and clock gating. Additionally, the DDR2/mDDR memory controller DLL, PHY, and the receivers at the I/O pins can be disabled. Even if the PHY is active, the receivers can be configured to disable whenever writes are in progress and the receivers are not needed.

Self-refresh mode can be used to preserve the contents of DDR2/mDDR memory when the DDR2/mDDR memory controller is clock gated or when the device is placed in RTC-only mode. However, in the RTC-only mode, care must be taken to correctly take the DDR2/mDDR out of self-refresh mode.

NOTE: To preserve the contents of the external memory while the DDR2/mDDR memory controller is clock gated, its self-refresh mode must be enabled before the DDR2/mDDR memory controller clock is turned off.

In RTC-only mode, all portions of the device except for the RTC are powered down, including the DDR2/mDDR memory controller. During power-up, the DDR2/mDDR memory controller defaults to its reset state. When the DDR2/mDDR memory controller is taken out of reset, it automatically runs its memory initialization routine; the self-refresh state of the memory is ignored. This hardware sequence cannot be stopped by software running on the device.

To correctly take the memory out of self-refresh after coming back from RTC-only mode, follow these steps:

1. Before going into RTC-only mode, disconnect the DDR2/mDDR memory controller CKE output pin from the memory; ensure the memory's CKE input pin continues to be driven low.
2. After coming back from RTC-only mode, configure the device to the desired operating state.
3. Program the DDR2/mDDR memory controller following the normal sequence.
4. Enable the self-refresh mode of the DDR2/mDDR memory controller.
5. Connect the DDR2/mDDR memory controller CKE output pin to the memory.
6. Disable the self-refresh mode of the DDR2/mDDR memory controller.

After this sequence, the DDR2/mDDR memory controller is ready for use. Note that hardware logic is needed to disconnect the CKE output pin from the memory and to drive the memory's CKE input pin low.

For more details on the power management features of the DDR2/mDDR memory controller, see the *DDR2/mDDR Memory Controller* chapter.

9.10.3 LVCMOS I/O Buffer Receiver Disable

This device supports two types of LVCMOS I/Os: 1.8V I/Os and low-static current dual-voltage I/Os that operate at either 1.8V or 3.3V. The receivers on the LCVMOS I/Os are enabled and disabled by software (see the RXACTIVE Control Register (RXACTIVE) in the *System Configuration (SYSCFG) Module* chapter). In the event that certain receivers are not used (such as in a low-power state), they can be disabled to conserve power.

9.10.4 Pull-Up/Pull-Down Disable

In general, you must ensure that all input pins are always pulled to a logic-high or a logic-low voltage level. A floating input pin can consume a small amount of I/O leakage current. The I/O leakage current can be greatly multiplied in the case of several floating inputs pins.

This device includes internal pull-up and pull-down resistors that prevent floating input pins. These internal resistors are generally very weak and their use is intended for pins that are not connected on the board design. For pins that are connected, external pull-up and pull-down resistors are recommended.

When an input pin is externally driven to a valid logic level, through an external pull-up resistor or by an external device for example, it is recommended to disable the internal resistor. Opposing an internal pull-up or pull-down resistor can consume a small amount of current. Internal resistors are disabled through the pullup/pulldown enable register (PUPD_ENA) in the *System Configuration (SYSCFG) Module* chapter.

Copyright © 2013–2016, Texas Instruments Incorporated

System Configuration (SYSCFG) Module

Topic **Page**

10.1 Introduction 192
10.2 Protection 192
10.3 Master Priority Control 193
10.4 Interrupt Support 194
10.5 SYSCFG Registers 195

Copyright © 2013–2016, Texas Instruments Incorporated

10.1 Introduction

The system configuration (SYSCFG) module is a system-level module containing status and top level control logic required by the device. The system configuration module consists of a set of memory-mapped status and control registers, accessible by the CPU, supporting all of the following system features, and miscellaneous functions and operations.

- Device Identification
- Device Configuration
  - Pin multiplexing control
  - Device Boot Configuration Status
- Master Priority Control
  - Controls the system priority for all master peripherals (including EDMA3TC)
- Emulation Control
  - Emulation suspend control for peripherals that support the feature
- Special Peripheral Status and Control
  - Locking of PLL control settings
  - Default burst size configuration for EDMA3 transfer controllers
  - Event source selection for the eCAP peripheral input capture
  - McASP0 AMUTEIN selection and clearing of AMUTE
  - USB PHY Control
  - Clock source selection for EMIFA and DDR2/mDDR
  - HPI Control

The system configuration module controls several global operations of the device; therefore, the module supports protection against erroneous and illegal accesses to the registers in its memory-map. The protection mechanisms that are present in the module are:

- A special key sequence that needs to be written into a set of registers in the system configuration module, to allow write ability to the rest of registers in the system configuration module.
- Several registers in the module are only accessible when the CPU requesting read/write access is in privileged mode.

10.2 Protection

The SYSCFG module controls several global operations of the device; therefore, it has a protection mechanism that prevents spurious and illegal accesses to the registers in its memory map. The protection mechanism enables accesses to these registers only if certain conditions are met.

10.2.1 Privilege Mode Protection

The CPU supports two privilege levels: Supervisor and User. Several registers in the SYSCFG memory-map can only be accessed when the accessing host (CPU or master peripheral) is operating in privileged mode, that is, in Supervisor mode. The registers that can only be accessed in privileged mode are listed in Section 10.5. See the ARM926EJ-S Technical Reference Manual (TRM), downloadable from http://infocenter.arm.com/help/index.jsp for details on privilege levels.

Copyright © 2013–2016, Texas Instruments Incorporated

10.2.2 Kicker Mechanism Protection

NOTE: The Kick registers are disabled in silicon revision 2 and later. The SYSCFG registers are always unlocked and writes to the Kick registers have no functional effect.

The Kick registers (KICK0R and KICK1R) can only be accessed in privileged mode (the host needs to be in Supervisor mode). Any number of accesses may be performed to the SYSCFG module, while the module is unlocked.

The SYSCFG module remains unlocked after the unlock sequence, until locked again. Locking the module is accomplished by writing any value other then the key values to either KICK0R or KICK1R.

To access any registers in the SYSCFG module, it is required to follow a special sequence of writes to the Kick registers (KICK0R and KICK1R) with correct key values. Writing the correct key value to the kick registers unlocks the registers in the SYSCFG memory-map. In order to access the SYSCFG registers, the following unlock sequence needs to be executed in software:

1. Write the key value of 83E7 0B13h to KICK0R.
2. Write the key value of 95A4 F1E0h to KICK1R.

After steps 1 and 2, the SYSCFG module registers are accessible and can be configured as per the application requirements.

10.3 Master Priority Control

The on-chip peripherals/modules are essentially divided into two broad categories, masters and slaves. The master peripherals are typically capable of initiating their own read/write data access requests, this includes the ARM, EDMA3 transfer controllers, and peripherals that do not rely on the CPU or EDMA3 for initiating the data transfer to/from them. In order to determine allowed connection between masters and slave, each master request source must have a unique master ID (mstid) associated with it. The master ID is shown in Table 10-1. See the device-specific data manual to determine the masters present on your device.

Each switched central resource (SCR) performs prioritization based on priority level of the master that sends the read/write requests. For all peripherals/ports classified as masters on the device, the priority is programmed in the master priority registers (MSTPRI0-3) in the SYSCFG modules. The default priority levels for each bus master is shown in Table 10-2. Application software is expected to modify these values to obtain the desired performance.

Table 10-1. Master IDs

Master ID	Peripheral
0	ARM - Instruction
1	ARM - Data
2-7	Reserved
8	PRU0
9	PRU1
10	EDMA3_0_CC0
11	EDMA3_1_CC0
12-15	Reserved
16	EDMA3_0_TC0 - read
17	EDMA3_0_TC0 - write
18	EDMA3_0_TC1 - read
19	EDMA3_0_TC1 - write
20	EDMA3_1_TC0 – read
21	EDMA3_1_TC0 – write
22-33	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-1. Master IDs (continued)

Master ID	Peripheral
34	USB2.0 CFG
35	USB2.0 DMA
36	Reserved
37	HPI
38-65	Reserved
66	uPP
67	Reserved
68	VPIF DMA0
69	VPIF DMA1
70-95	Reserved
96	LCDC
97-255	Reserved

Table 10-2. Default Master Priority

Master	Default Priority[1]	Master Priority Register
PRU0	0	MSTPRI1
PRU1	0	MSTPRI1
EDMA3_0_TC0[2]	0	MSTPRI1
EDMA3_0_TC1[2]	0	MSTPRI1
ARM - Instruction	2	MSTPRI0
ARM - Data	2	MSTPRI0
uPP	4	MSTPRI0
EDMA3_1_TC0[2]	4	MSTPRI1
VPIF DMA0	4	MSTPRI1
VPIF DMA1	4	MSTPRI1
USB2.0 CFG	4	MSTPRI2
USB2.0 DMA	4	MSTPRI2
LCDC[3]	5	MSTPRI2
HPI	6	MSTPRI2

(1) The default priority settings might not be optimal for all applications. The master priority should be changed from default based on application specific requirement, in order to get optimal performance and prioritization for masters moving data that is real time sensitive.

(2) The priority for EDMA3_0_TC0, EDMA3_0_TC1, and EDMA3_1_TC0 is configurable through fields in the master priority 1 register (MSTPRI1), not the EDMA3CC QUEPRI register.

(3) LCDC traffic is typically real-time sensitive, therefore, the default priority of 5, which is lower as compared to the default priority of several masters, is not recommended. You should reconfigure the LCDC priority to the highest or equal to other high-priority masters in an application to ensure that the throughput/latency requirements for the LCDC are met.

10.4 Interrupt Support

10.4.1 Interrupt Events and Requests

The SYSCFG module generates two interrupts: an address error interrupt (BOOTCFG_ADDR_ERR) and a protection interrupt (BOOTCFG_PROT_ERR). The BOOTCFG_ADDR_ERR is generated when there is an addressing violation due to an access to a non-existent location in the SYSCFG register space. The BOOTCFG_PROT_ERR interrupt is generated when there is a protection violation of either in the defined ranges or to the SYSCFG registers. It is required to write a value of 0 to the end of interrupt register (EOI) after the software has processed the SYSCFG interrupt, this acts as an acknowledgement of completion of the SYSCFG interrupt so that the module can reliably generate subsequent interrupts.

Copyright © 2013–2016, Texas Instruments Incorporated

The transfer parameters that caused the violation are saved in the fault address register (FLTADDRR) and the fault status register (FLTSTAT).

10.4.2 Interrupt Multiplexing

The interrupts from the SYSCFG module are combined with the interrupts from the MPU module into a single interrupt called MPU_BOOTCFG_ERR. The combined interrupt is routed to the ARM interrupt controller.

10.5 SYSCFG Registers

Table 10-3 lists the memory-mapped registers for the system configuration module 0 (SYSCFG0) and Table 10-4 lists the memory-mapped registers for the system configuration module 1 (SYSCFG1). These tables also indicate whether a particular register can be accessed only when the CPU is in privileged mode.

Table 10-3. System Configuration Module 0 (SYSCFG0) Registers

Address	Acronym	Register Description	Access	Section
01C1 4000h	REVID	Revision Identification Register	—	Section 10.5.1
01C1 4008h	DIEIDR0[(1)]	Die Identification Register 0	—	—
01C1 400Ch	DIEIDR1[(1)]	Die Identification Register 1	—	—
01C1 4010h	DIEIDR2[(1)]	Die Identification Register 2	—	—
01C1 4014h	DIEIDR3[(1)]	Die Identification Register 3	—	—
01C1 4018h	DEVIDR0	Device Identification Register 0	Privileged mode	Section 10.5.2
01C1 4020h	BOOTCFG	Boot Configuration Register	Privileged mode	Section 10.5.3
01C1 4024h	CHIPREVIDR	Chip Revision Identification Register	Privileged mode	Section 10.5.4
01C1 4038h	KICK0R	Kick 0 Register	Privileged mode	Section 10.5.5.1
01C1 403Ch	KICK1R	Kick 1 Register	Privileged mode	Section 10.5.5.2
01C1 4040h	HOST0CFG	Host 0 Configuration Register	—	Section 10.5.6
01C1 40E0h	IRAWSTAT	Interrupt Raw Status/Set Register	Privileged mode	Section 10.5.7.1
01C1 40E4h	IENSTAT	Interrupt Enable Status/Clear Register	Privileged mode	Section 10.5.7.2
01C1 40E8h	IENSET	Interrupt Enable Register	Privileged mode	Section 10.5.7.3
01C1 40ECh	IENCLR	Interrupt Enable Clear Register	Privileged mode	Section 10.5.7.4
01C1 40F0h	EOI	End of Interrupt Register	Privileged mode	Section 10.5.7.5
01C1 40F4h	FLTADDRR	Fault Address Register	Privileged mode	Section 10.5.8.1
01C1 40F8h	FLTSTAT	Fault Status Register	—	Section 10.5.8.2
01C1 4110h	MSTPRI0	Master Priority 0 Register	Privileged mode	Section 10.5.9.1
01C1 4114h	MSTPRI1	Master Priority 1 Register	Privileged mode	Section 10.5.9.2
01C1 4118h	MSTPRI2	Master Priority 2 Register	Privileged mode	Section 10.5.9.3
01C1 4120h	PINMUX0	Pin Multiplexing Control 0 Register	Privileged mode	Section 10.5.10.1
01C1 4124h	PINMUX1	Pin Multiplexing Control 1 Register	Privileged mode	Section 10.5.10.2
01C1 4128h	PINMUX2	Pin Multiplexing Control 2 Register	Privileged mode	Section 10.5.10.3
01C1 412Ch	PINMUX3	Pin Multiplexing Control 3 Register	Privileged mode	Section 10.5.10.4
01C1 4130h	PINMUX4	Pin Multiplexing Control 4 Register	Privileged mode	Section 10.5.10.5
01C1 4134h	PINMUX5	Pin Multiplexing Control 5 Register	Privileged mode	Section 10.5.10.6
01C1 4138h	PINMUX6	Pin Multiplexing Control 6 Register	Privileged mode	Section 10.5.10.7
01C1 413Ch	PINMUX7	Pin Multiplexing Control 7 Register	Privileged mode	Section 10.5.10.8
01C1 4140h	PINMUX8	Pin Multiplexing Control 8 Register	Privileged mode	Section 10.5.10.9
01C1 4144h	PINMUX9	Pin Multiplexing Control 9 Register	Privileged mode	Section 10.5.10.10
01C1 4148h	PINMUX10	Pin Multiplexing Control 10 Register	Privileged mode	Section 10.5.10.11

(1) This register is for internal-use only.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-3. System Configuration Module 0 (SYSCFG0) Registers (continued)

Address	Acronym	Register Description	Access	Section
01C1 414Ch	PINMUX11	Pin Multiplexing Control 11 Register	Privileged mode	Section 10.5.10.12
01C1 4150h	PINMUX12	Pin Multiplexing Control 12 Register	Privileged mode	Section 10.5.10.13
01C1 4154h	PINMUX13	Pin Multiplexing Control 13 Register	Privileged mode	Section 10.5.10.14
01C1 4158h	PINMUX14	Pin Multiplexing Control 14 Register	Privileged mode	Section 10.5.10.15
01C1 415Ch	PINMUX15	Pin Multiplexing Control 15 Register	Privileged mode	Section 10.5.10.16
01C1 4160h	PINMUX16	Pin Multiplexing Control 16 Register	Privileged mode	Section 10.5.10.17
01C1 4164h	PINMUX17	Pin Multiplexing Control 17 Register	Privileged mode	Section 10.5.10.18
01C1 4168h	PINMUX18	Pin Multiplexing Control 18 Register	Privileged mode	Section 10.5.10.19
01C1 416Ch	PINMUX19	Pin Multiplexing Control 19 Register	Privileged mode	Section 10.5.10.20
01C1 4170h	SUSPSRC	Suspend Source Register	Privileged mode	Section 10.5.11
01C1 4174h	CHIPSIG	Chip Signal Register	—	Section 10.5.12
01C1 4178h	CHIPSIG_CLR	Chip Signal Clear Register	—	Section 10.5.13
01C1 417Ch	CFGCHIP0	Chip Configuration 0 Register	Privileged mode	Section 10.5.14
01C1 4180h	CFGCHIP1	Chip Configuration 1 Register	Privileged mode	Section 10.5.15
01C1 4188h	CFGCHIP3	Chip Configuration 3 Register	Privileged mode	Section 10.5.17
01C1 418Ch	CFGCHIP4	Chip Configuration 4 Register	Privileged mode	Section 10.5.18

Table 10-4. System Configuration Module 1 (SYSCFG1) Registers

Address	Acronym	Register Description	Access	Section
01E2 C000h	VTPIO_CTL	VTP I/O Control Register	Privileged mode	Section 10.5.19
01E2 C004h	DDR_SLEW	DDR Slew Register	Privileged mode	Section 10.5.20
01E2 C008h	DEEPSLEEP	Deep Sleep Register	Privileged mode	Section 10.5.21
01E2 C00Ch	PUPD_ENA	Pullup/Pulldown Enable Register	Privileged mode	Section 10.5.22
01E2 C010h	PUPD_SEL	Pullup/Pulldown Selection Register	Privileged mode	Section 10.5.23
01E2 C014h	RXACTIVE	RXACTIVE Control Register	Privileged mode	Section 10.5.24

10.5.1 Revision Identification Register (REVID)

The revision identification register (REVID) provides the revision information for the SYSCFG module. The REVID is shown in Figure 10-1 and described in Table 10-5.

Figure 10-1. Revision Identification Register (REVID)

31	0
REV	
R-4E84 0102h	

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

Table 10-5. Revision Identification Register (REVID) Field Descriptions

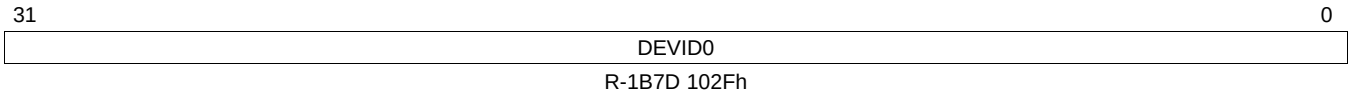
Bit	Field	Value	Description
31-0	REV	4E84 0102h	**Revision ID.** Revision information for the SYSCFG module.

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.2 Device Identification Register 0 (DEVIDR0)

The device identification register 0 (DEVIDR0) contains a software readable version of the JTAG ID device. Software can use this register to determine the version of the device on which it is executing. The DEVIDR0 is shown in Figure 10-2 and described in Table 10-6.

Figure 10-2. Device Identification Register 0 (DEVIDR0)



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

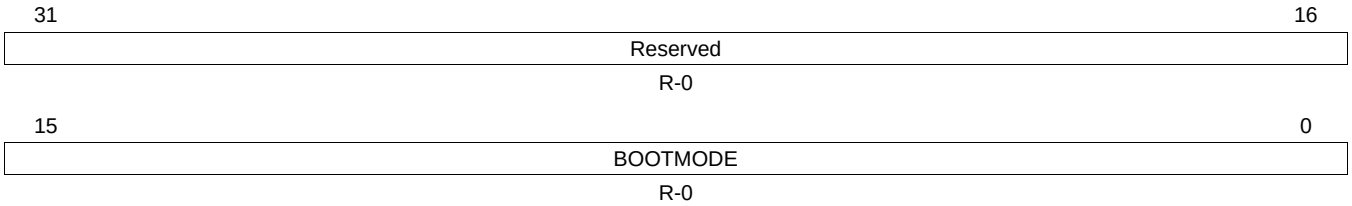
Table 10-6. Device Identification Register 0 (DEVIDR0) Field Descriptions

Bit	Field	Value	Description
31-0	DEVID0	1B7D 102Fh	Device identification.

10.5.3 Boot Configuration Register (BOOTCFG)

The device boot and configuration settings are latched at device reset, and captured in the boot configuration register (BOOTCFG). See your device-specific data manual and the *Boot Considerations* chapter for details on boot and configuration settings. The BOOTCFG is shown in Figure 10-3 and described in Table 10-7.

Figure 10-3. Boot Configuration Register (BOOTCFG)



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

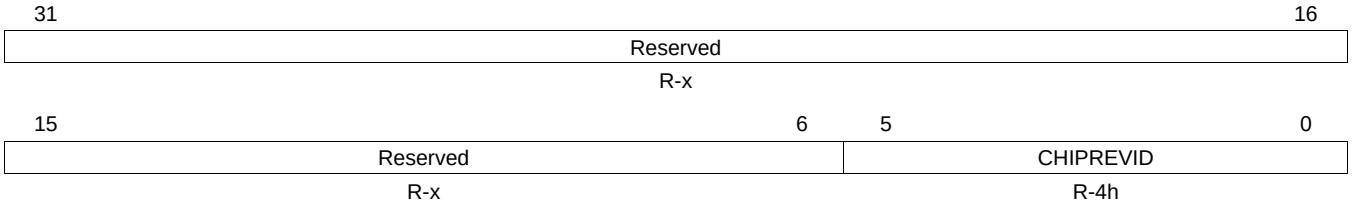
Table 10-7. Boot Configuration Register (BOOTCFG) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	BOOTMODE	0-FFFFh	**Boot Mode.** This reflects the state of the boot mode pins.

10.5.4 Chip Revision Identification Register (CHIPREVIDR)

The chip revision identification register (CHIPREVIDR) provides the software-readable silicon revision information for the device. The CHIPREVID is shown in Figure 10-4 and described in Table 10-8.

Figure 10-4. Chip Revision Identification Register (CHIPREVIDR)



LEGEND: R = Read only; -*n* = value after reset; x = value is indeterminate after reset

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-8. Chip Revision Identification Register (CHIPREVIDR) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reserved
5-0	CHIPREVID		**Identifies silicon revision of device.**
		0-3h	Older silicon revision
		4h	Silicon revision 2.2

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.5 Kick Registers (KICK0R-KICK1R)

NOTE: The kick registers are disabled in silicon revision 2 and later. The SYSCFG registers are always unlocked and writes to the kick registers have no functional effect.

The SYSCFG module has a protection mechanism to prevent any spurious writes from changing any of the modules memory-mapped registers. At power-on reset, none of the SYSCFG module registers are writeable (they are readable). To allow writing to the registers in the module, it is required to “unlock” the registers by writing to two memory-mapped registers in the SYSCFG module, Kick0 and Kick1, with exact data values. Once these values are written, then all the registers in the SYSCFG module that are writeable can be written to. See Section 10.2.2 for the exact key values and sequence of steps. Writing any other data value to either of these kick registers will cause the memory mapped registers to be “locked” again and block out any write accesses to registers in the SYSCFG module.

10.5.5.1 Kick 0 Register (KICK0R)

The KICK0R is shown in Figure 10-5 and described in Table 10-9.

Figure 10-5. Kick 0 Register (KICK0R)

31	0
KICK1	
R/W-0	

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

Table 10-9. Kick 0 Register (KICK0R) Field Descriptions

Bit	Field	Value	Description
31-0	KICK0	0-FFFF FFFFh	**KICK0R allows writing to unlock the kick0 data.** The written data must be 83E7 0B13h to unlock this register. It must be written before writing to the kick1 register. Writing any other value will lock the other MMRs.

10.5.5.2 Kick 1 Register (KICK1R)

The KICK1R is shown in Figure 10-6 and described in Table 10-10.

Figure 10-6. Kick 1 Register (KICK1R)

31	0
KICK0	
R/W-0	

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

Table 10-10. Kick 1 Register (KICK1R) Field Descriptions

Bit	Field	Value	Description
31-0	KICK1	0-FFFF FFFFh	**KICK1R allows writing to unlock the kick1 data and the kicker mechanism to write to other MMRs.** The written data must be 95A4 F1E0h to unlock this register. KICK0R must be written before writing to the kick1 register. Writing any other value will lock the other MMRs.

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.6 Host 0 Configuration Register (HOST0CFG)

The ARM subsystem is held in reset when 0 is written to the BOOTRDY bit in the host 0 configuration register (HOST0CFG). In a typical application, the BOOTRDY bit should not be cleared.

The HOST0CFG is shown in Figure 10-7 and described in Table 10-11.

Figure 10-7. Host 0 Configuration Register (HOST0CFG)

31	16
Reserved	
R-0	

15	1	0
Reserved		BOOTRDY
R-0		R/W-1

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

Table 10-11. Host 0 Configuration Register (HOST0CFG) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	BOOTRDY		**ARM boot ready bit allowing ARM to boot.**
		0	ARM held in reset mode.
		1	ARM released from wait in reset mode.

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.7 Interrupt Registers

The interrupt registers are a set of registers that provide control for the address and protection violation error interrupt generated by the SYSCFG module when there is an address or protection violation to the module's memory-mapped register address space. This includes enable control, interrupt set and clear control, and end of interrupt (EOI) control.

10.5.7.1 Interrupt Raw Status/Set Register (IRAWSTAT)

The interrupt raw status/set register (IRAWSTAT) shows the interrupt status before enabling the interrupt and allows setting of the interrupt status. The IRAWSTAT is shown in Figure 10-8 and described in Table 10-12.

Figure 10-8. Interrupt Raw Status/Set Register (IRAWSTAT)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		ADDRERR	PROTERR
R-0		R/W-0	R/W-0

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

Table 10-12. Interrupt Raw Status/Set Register (IRAWSTAT) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved. Always read 0.
1	ADDRERR		**Addressing violation error.** Reading this bit field reflects the raw status of the interrupt before enabling.
		0	Indicates the interrupt is not set. Writing 0 has no effect.
		1	Indicates the interrupt is set. Writing 1 sets the status.
0	PROTERR		**Protection violation error.** Reading this bit field reflects the raw status of the interrupt before enabling.
		0	Indicates the interrupt is not set. Writing 0 has no effect.
		1	Indicates the interrupt is set. Writing 1 sets the status.

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.7.2 Interrupt Enable Status/Clear Register (IENSTAT)

The interrupt enable status/clear register (IENSTAT) shows the status of enabled interrupt and allows clearing of the interrupt status. The IENSTAT is shown in Figure 10-9 and described in Table 10-13.

Figure 10-9. Interrupt Enable Status/Clear Register (IENSTAT)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		ADDRERR	PROTERR
R-0		R/W-0	R/W-0

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

Table 10-13. Interrupt Enable Status/Clear Register (IENSTAT) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved. Always read 0.
1	ADDRERR		**Addressing violation error.** Reading this bit field reflects the interrupt enabled status.
		0	Indicates the interrupt is not set. Writing 0 has no effect.
		1	Indicates the interrupt is set. Writing 1 clears the status.
0	PROTERR		**Protection violation error.** Reading this bit field reflects the interrupt enabled status.
		0	Indicates the interrupt is not set. Writing 0 has no effect.
		1	Indicates the interrupt is set. Writing 1 clears the status.

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.7.3 Interrupt Enable Register (IENSET)

The interrupt enable register (IENSET) allows setting/enabling the interrupt for address and/or protection violation condition. It also shows the value of the register (whether or not interrupt is enabled). The IENSET is shown in Figure 10-10 and described in Table 10-14.

Figure 10-10. Interrupt Enable Register (IENSET)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		ADDRERR_EN	PROTERR_EN
R-0		R/W-0	R/W-0

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

Table 10-14. Interrupt Enable Register (IENSET) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved. Always read 0.
1	ADDRERR_EN		**Addressing violation error.**
		0	Writing a 0 has not effect.
		1	Writing a 1 enables this interrupt.
0	PROTERR_EN		**Protection violation error.**
		0	Writing a 0 has not effect.
		1	Writing a 1 enables this interrupt.

10.5.7.4 Interrupt Enable Clear Register (IENCLR)

The interrupt enable clear register (IENCLR) allows clearing/disable the interrupt for address and/or protection violation condition. It also shows the value of the interrupt enable register (IENSET). The IENCLR is shown in Figure 10-11 and described in Table 10-15.

Figure 10-11. Interrupt Enable Clear Register (IENCLR)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		ADDRERR_CLR	PROTERR_CLR
R-0		R/W-0	R/W-0

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

Table 10-15. Interrupt Enable Clear Register (IENCLR) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved. Always read 0.
1	ADDRERR_CLR		**Addressing violation error.**
		0	Writing a 0 has not effect.
		1	Writing a 1 clears/disables this interrupt.
0	PROTERR_CLR		**Protection violation error.**
		0	Writing a 0 has not effect.
		1	Writing a 1 clears/disables this interrupt.

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.7.5 End of Interrupt Register (EOI)

The end of interrupt register (EOI) is used in software to indicate completion of the interrupt servicing of the SYSCFG interrupt (for address/protection violation). It is required to write a value of 0 to the EOI register after the software has processed the SYSCFG interrupt, this acts as an acknowledgement of completion of the SYSCFG interrupt so that the module can reliably generate the subsequent interrupts. The EOI is shown in Figure 10-12 and described in Table 10-16.

Figure 10-12. End of Interrupt Register (EOI)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		EOIVECT	
R-0		W-0	

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

Table 10-16. End of Interrupt Register (EOI) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved. Always read 0.
7-0	EOIVECT	0-FFh	**EOI vector value.** Write the interrupt distribution value of the chip.

10.5.8 Fault Registers

The fault registers are a group of registers responsible for capturing the details on the faulty (address/protection violation errors) accesses, such as address and type of error.

10.5.8.1 Fault Address Register (FLTADDRR)

The fault address register (FLTADDRR) captures the address of the first transfer that causes the address or memory violation error. The FLTADDRR is shown in Figure 10-13 and described in Table 10-17.

Figure 10-13. Fault Address Register (FLTADDRR)

31	0
FLTADDR	
R-0	

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

Table 10-17. Fault Address Register (FLTADDRR) Field Descriptions

Bit	Field	Value	Description
31-0	FLTADDR	0-FFFF FFFFh	**Fault address for the first fault transfer.**

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.8.2 Fault Status Register (FLTSTAT)

The fault status register (FLTSTAT) holds/captures additional attributes and status of the first erroneous transaction. This includes things like the master id for the master that caused the address/memory violation error, details on whether it is a user or supervisor level read/write or execute fault. The FLTSTAT is shown in Figure 10-14 and described in Table 10-18.

Figure 10-14. Fault Status Register (FLTSTAT)

31	24	23	16
ID		MSTID	
R-0		R-0	

15	13	12	9	8	6	5	0
Reserved		PRIVID		Reserved		TYPE	
R-0		R-0		R-0		R-0	

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

Table 10-18. Fault Status Register (FLTSTAT) Field Descriptions

Bit	Field	Value	Description
31-24	ID	0-FFh	**Transfer ID of the first fault transfer.**
23-16	MSTID	0-FFh	**Master ID of the first fault transfer.**
15-13	Reserved	0	Reserved. Always read 0
12-9	PRIVID	0-Fh	**Privilege ID of the first fault transfer.**
8-6	Reserved	0	Reserved. Always read 0
5-0	TYPE		**Fault type of first fault transfer.**
		0	No transfer fault
		1h	User execute fault
		2h	User write fault
		3h	*Reserved*
		4h	User read fault
		5h-7h	*Reserved*
		8h	Supervisor execute fault
		9h-Fh	*Reserved*
		10h	Supervisor write fault
		11h-1Fh	*Reserved*
		20h	Supervisor read fault
		21h-3Fh	*Reserved*

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.9 Master Priority Registers (MSTPRI0-MSTPRI2)

10.5.9.1 Master Priority 0 Register (MSTPRI0)

The master priority 0 register (MSTPRI0) is shown in Figure 10-15 and described in Table 10-19.

Figure 10-15. Master Priority 0 Register (MSTPRI0)

31	30 28	27	26 24	23	22 20	19	18 16
Rsvd	Reserved	Rsvd	Reserved	Rsvd	Reserved	Rsvd	UPP
R/W-0	R/W-4h	R/W-0	R/W-4h	R/W-0	R/W-4h	R/W-0	R/W-4h

15	14 12	11	10 8	7	6 4	3	2 0
Rsvd	Reserved	Rsvd	Reserved	Rsvd	ARM_D	Rsvd	ARM_I
R/W-0	R/W-2h	R-0	R/W-2h	R-0	R/W-2h	R-0	R/W-2h

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

Table 10-19. Master Priority 0 Register (MSTPRI0) Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reserved. Write the default value when modifying this register.
30-28	Reserved	4h	Reserved. Write the default value when modifying this register.
27	Reserved	0	Reserved. Write the default value when modifying this register.
26-24	Reserved	4h	Reserved. Write the default value when modifying this register.
23	Reserved	0	Reserved. Write the default value when modifying this register.
22-20	Reserved	4h	Reserved. Write the default value when modifying this register.
19	Reserved	0	Reserved. Write the default value when modifying this register.
18-16	UPP	0-7h	**uPP port priority.** Bit = 0 = priority 0 (highest); bit = 7h = priority 7 (lowest).
15	Reserved	0	Reserved. Write the default value when modifying this register.
14-12	Reserved	2h	Reserved. Write the default value when modifying this register.
11	Reserved	0	Reserved. Always read as 0.
10-8	Reserved	2h	Reserved. Write the default value when modifying this register.
7	Reserved	0	Reserved. Always read as 0.
6-4	ARM_D	0-7h	**ARM_D port priority.** Bit = 0 = priority 0 (highest); bit = 7h = priority 7 (lowest).
3	Reserved	0	Reserved. Always read as 0.
2-0	ARM_I	0-7h	**ARM_I port priority.** Bit = 0 = priority 0 (highest); bit = 7h = priority 7 (lowest).

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.9.2 Master Priority 1 Register (MSTPRI1)

The master priority 1 register (MSTPRI1) is shown in Figure 10-16 and described in Table 10-20.

Figure 10-16. Master Priority 1 Register (MSTPRI1)

31	30 28	27	26 24	23	22 20	19	18 16
Rsvd	VPIF_DMA_1	Rsvd	VPIF_DMA_0	Rsvd	Reserved	Rsvd	EDMA31TC0
R/W-0	R/W-4h	R/W-0	R/W-4h	R/W-0	R/W-4h	R/W-0	R/W-4h

15	14 12	11	10 8	7	6 4	3	2 0
Rsvd	EDMA30TC1	Rsvd	EDMA30TC0	Rsvd	PRU1	Rsvd	PRU0
R/W-0	R/W-0	R-0	R/W-0	R-0	R/W-0	R-0	R/W-0

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

Table 10-20. Master Priority 1 Register (MSTPRI1) Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reserved. Write the default value when modifying this register.
30-28	VPIF_DMA_1	0-7h	**VPIF DMA1 port priority.** Bit = 0 = priority 0 (highest); bit = 7h = priority 7 (lowest).
27	Reserved	0	Reserved. Write the default value when modifying this register.
26-24	VPIF_DMA_0	0-7h	**VPIF DMA0 port priority.** Bit = 0 = priority 0 (highest); bit = 7h = priority 7 (lowest).
23	Reserved	0	Reserved. Write the default value when modifying this register.
22-20	Reserved	4h	Reserved. Write the default value when modifying this register.
19	Reserved	0	Reserved. Write the default value when modifying this register.
18-16	EDMA31TC0	0-7h	**EDMA3_1_TC0 port priority.** Bit = 0 = priority 0 (highest); bit = 7h = priority 7 (lowest).
15	Reserved	0	Reserved. Write the default value when modifying this register.
14-12	EDMA30TC1	0-7h	**EDMA3_0_TC1 port priority.** Bit = 0 = priority 0 (highest); bit = 7h = priority 7 (lowest).
11	Reserved	0	Reserved. Always read as 0.
10-8	EDMA30TC0	0-7h	**EDMA3_0_TC0 port priority.** Bit = 0 = priority 0 (highest); bit = 7h = priority 7 (lowest).
7	Reserved	0	Reserved. Always read as 0.
6-4	PRU1	0-7h	**PRU1 port priority.** Bit = 0 = priority 0 (highest); bit = 7h = priority 7 (lowest).
3	Reserved	0	Reserved. Always read as 0.
2-0	PRU0	0-7h	**PRU0 port priority.** Bit = 0 = priority 0 (highest); bit = 7h = priority 7 (lowest).

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.9.3 Master Priority 2 Register (MSTPRI2)

The master priority 2 register (MSTPRI2) is shown in Figure 10-17 and described in Table 10-21.

Figure 10-17. Master Priority 2 Register (MSTPRI2)

31	30 28	27	26 24	23	22 20	19	18 16
Rsvd	LCDC	Rsvd	Reserved	Rsvd	UHPI	Rsvd	Reserved
R/W-0	R/W-5h	R/W-0	R/W-4h	R/W-0	R/W-6h	R/W-0	R/W-0

15	14 12	11	10 8	7	6 4	3	2 0
Rsvd	USB0CDMA	Rsvd	USB0CFG	Rsvd	Reserved	Rsvd	Reserved
R/W-0	R/W-4h	R/W-0	R/W-4h	R/W-0	R/W-0	R/W-0	R/W-4h

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

Table 10-21. Master Priority 2 Register (MSTPRI2) Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reserved. Write the default value when modifying this register.
30-28	LCDC	0-7h	**LCDC port priority.** Bit = 0 = priority 0 (highest); bit = 7h = priority 7 (lowest).
27	Reserved	0	Reserved. Write the default value when modifying this register.
26-24	Reserved	4h	Reserved. Write the default value when modifying this register.
23	Reserved	0	Reserved. Write the default value when modifying this register.
22-20	UHPI	0-7h	**HPI port priority.** Bit = 0 = priority 0 (highest); bit = 7h = priority 7 (lowest).
19	Reserved	0	Reserved. Write the default value when modifying this register.
18-16	Reserved	0	Reserved. Write the default value to all bits when modifying this register.
15	Reserved	0	Reserved. Write the default value when modifying this register.
14-12	USB0CDMA	0-7h	**USB0 (USB2.0) CDMA port priority.** Bit = 0 = priority 0 (highest); bit = 7h = priority 7 (lowest).
11	Reserved	0	Reserved. Write the default value when modifying this register.
10-8	USB0CFG	0-7h	**USB0 (USB2.0) CFG port priority.** Bit = 0 = priority 0 (highest); bit = 7h = priority 7 (lowest).
7	Reserved	0	Reserved. Write the default value to all bits when modifying this register.
6-4	Reserved	0	Reserved. Write the default value to all bits when modifying this register.
3	Reserved	0	Reserved. Write the default value when modifying this register.
2-0	Reserved	4h	Reserved. Write the default value when modifying this register.

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.10 Pin Multiplexing Control Registers (PINMUX0-PINMUX19)

Extensive use of pin multiplexing is used to accommodate the large number of peripheral functions in the smallest possible package. On the device, pin multiplexing can be controlled on a pin by pin basis. This is done by the pin multiplexing registers (PINMUX0-PINMUX19). Each pin that is multiplexed with several different functions has a corresponding 4-bit field in PINMUX*n*. Pin multiplexing selects which of several peripheral pin functions control the pins I/O buffer output data and output enable values only. Note that the input from each pin is always routed to all of the peripherals that share the pin; the PINMUX registers have no effect on input from a pin. Hardware does not attempt to ensure that the proper pin multiplexing is selected for the peripherals or that interface mode is being used. Detailed information about the pin multiplexing and control is covered in the device-specific data manual. Access to the pin multiplexing utility is available in *AM18xx Pin Multiplexing Utility Application Report* (SPRABA2).

10.5.10.1 Pin Multiplexing Control 0 Register (PINMUX0)

Figure 10-18. Pin Multiplexing Control 0 Register (PINMUX0)

31 28	27 24	23 20	19 16
PINMUX0_31_28	PINMUX0_27_24	PINMUX0_23_20	PINMUX0_19_16
R/W-0	R/W-0	R/W-0	R/W-0
15 12	**11 8**	**7 4**	**3 0**
PINMUX0_15_12	PINMUX0_11_8	PINMUX0_7_4	PINMUX0_3_0
R/W-0	R/W-0	R/W-0	R/W-0

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

Table 10-22. Pin Multiplexing Control 0 Register (PINMUX0) Field Descriptions

Bit	Field	Value	Description	Type[1]
31-28	PINMUX0_31_28		**RTC_ALARM/$\overline{\text{UART2_CTS}}$/GP0[8]/$\overline{\text{DEEPSLEEP}}$ Control**	
		0	Selects Function $\overline{\text{DEEPSLEEP}}$	I
		1h	*Reserved*	X
		2h	Selects Function RTC_ALARM	O
		3h	*Reserved*	X
		4h	Selects Function $\overline{\text{UART2_CTS}}$	I
		5h-7h	*Reserved*	X
		8h	Selects Function GP0[8]	I/O
		9h-Fh	*Reserved*	X
27-24	PINMUX0_27_24		**AMUTE/PRU0_R30[16]/$\overline{\text{UART2_RTS}}$/GP0[9]/PRU0_R31[16] Control**	
		0	Selects Function PRU0_R31[16]	I
		1h	Selects Function AMUTE	I/O
		2h	Selects Function PRU0_R30[16]	O
		3h	*Reserved*	X
		4h	Selects Function $\overline{\text{UART2_RTS}}$	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP0[9]	I/O
		9h-Fh	*Reserved*	X

(1) I = Input, O = Output, I/O = Bidirectional, X = Undefined

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-22. Pin Multiplexing Control 0 Register (PINMUX0) Field Descriptions (continued)

Bit	Field	Value	Description	Type[1]
23-20	PINMUX0_23_20		**AHCLKX/USB_REFCLKIN/$\overline{\text{UART1_CTS}}$/GP0[10]/PRU0_R31[17] Control**	
		0	Selects Function PRU0_R31[17]	I
		1h	Selects Function AHCLKX	I/O
		2h	Selects Function USB_REFCLKIN	I
		3h	*Reserved*	X
		4h	Selects Function $\overline{\text{UART1_CTS}}$	I
		5h-7h	*Reserved*	X
		8h	Selects Function GP0[10]	I/O
		9h-Fh	*Reserved*	X
19-16	PINMUX0_19_16		**AHCLKR/PRU0_R30[18]/$\overline{\text{UART1_RTS}}$/GP0[11]/PRU0_R31[18] Control**	
		0	Selects Function PRU0_R31[18]	I
		1h	Selects Function AHCLKR	I/O
		2h	Selects Function PRU0_R30[18]	O
		3h	*Reserved*	X
		4h	Selects Function $\overline{\text{UART1_RTS}}$	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP0[11]	I/O
		9h-Fh	*Reserved*	X
15-12	PINMUX0_15_12		**AFSX/GP0[12]/PRU0_R31[19] Control**	
		0	Selects Function PRU0_R31[19]	I
		1h	Selects Function AFSX	I/O
		2h-7h	*Reserved*	X
		8h	Selects Function GP0[12]	I/O
		9h-Fh	*Reserved*	X
11-8	PINMUX0_11_8		**AFSR/GP0[13]/PRU0_R31[20] Control**	
		0	Selects Function PRU0_R31[20]	I
		1h	Selects Function AFSR	I/O
		2h-7h	*Reserved*	X
		8h	Selects Function GP0[13]	I/O
		9h-Fh	*Reserved*	X
7-4	PINMUX0_7_4		**ACLKX/PRU0_R30[19]/GP0[14]/PRU0_R31[21] Control**	
		0	Selects Function PRU0_R31[21]	I
		1h	Selects Function ACLKX	I/O
		2h-3h	*Reserved*	X
		4h	Selects Function PRU0_R30[19]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP0[14]	I/O
		9h-Fh	*Reserved*	X
3-0	PINMUX0_3_0		**ACLKR/PRU0_R30[20]/GP0[15]/PRU0_R31[22] Control**	
		0	Selects Function PRU0_R31[22]	I
		1h	Selects Function ACLKR	I/O
		2h-3h	*Reserved*	X
		4h	Selects Function PRU0_R30[20]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP0[15]	I/O
		9h-Fh	*Reserved*	X

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.10.2 Pin Multiplexing Control 1 Register (PINMUX1)

Figure 10-19. Pin Multiplexing Control 1 Register (PINMUX1)

31 28	27 24	23 20	19 16
PINMUX1_31_28	PINMUX1_27_24	PINMUX1_23_20	PINMUX1_19_16
R/W-0	R/W-0	R/W-0	R/W-0
15 12	**11 8**	**7 4**	**3 0**
PINMUX1_15_12	PINMUX1_11_8	PINMUX1_7_4	PINMUX1_3_0
R/W-0	R/W-0	R/W-0	R/W-0

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

Table 10-23. Pin Multiplexing Control 1 Register (PINMUX1) Field Descriptions

Bit	Field	Value	Description	Type[1]
31-28	PINMUX1_31_28		**AXR8/CLKS1/ECAP1_APWM1/GP0[0]/PRU0_R31[8] Control**	
		0	Selects Function PRU0_R31[8]	I
		1h	Selects Function AXR8	I/O
		2h	Selects Function CLKS1	I
		3h	*Reserved*	X
		4h	Selects Function ECAP1_APWM1	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP0[0]	I/O
		9h-Fh	*Reserved*	X
27-24	PINMUX1_27_24		**AXR9/DX1/GP0[1] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function AXR9	I/O
		2h	Selects Function DX1	O
		3h-7h	*Reserved*	X
		8h	Selects Function GP0[1]	I/O
		9h-Fh	*Reserved*	X
23-20	PINMUX1_23_20		**AXR10/DR1/GP0[2] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function AXR10	I/O
		2h	Selects Function DR1	I
		3h-7h	*Reserved*	X
		8h	Selects Function GP0[2]	I/O
		9h-Fh	*Reserved*	X
19-16	PINMUX1_19_16		**AXR11/FSX1/GP0[3] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function AXR11	I/O
		2h	Selects Function FSX1	I/O
		3h-7h	*Reserved*	X
		8h	Selects Function GP0[3]	I/O
		9h-Fh	*Reserved*	X

(1) I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-23. Pin Multiplexing Control 1 Register (PINMUX1) Field Descriptions (continued)

Bit	Field	Value	Description	Type[1]
15-12	PINMUX1_15_12		**AXR12/FSR1/GP0[4] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function AXR12	I/O
		2h	Selects Function FSR1	I/O
		3h-7h	*Reserved*	X
		8h	Selects Function GP0[4]	I/O
		9h-Fh	*Reserved*	X
11-8	PINMUX1_11_8		**AXR13/CLKX1/GP0[5] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function AXR13	I/O
		2h	Selects Function CLKX1	I/O
		3h-7h	*Reserved*	X
		8h	Selects Function GP0[5]	I/O
		9h-Fh	*Reserved*	X
7-4	PINMUX1_7_4		**AXR14/CLKR1/GP0[6] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function AXR14	I/O
		2h	Selects Function CLKR1	I/O
		3h-7h	*Reserved*	X
		8h	Selects Function GP0[6]	I/O
		9h-Fh	*Reserved*	X
3-0	PINMUX1_3_0		**AXR15/EPWM0TZ[0]/ECAP2_APWM2/GP0[7] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function AXR15	I/O
		2h	Selects Function EPWM0TZ[0]	I
		3h	*Reserved*	X
		4h	Selects Function ECAP2_APWM2	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP0[7]	I/O
		9h-Fh	*Reserved*	X

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.10.3 Pin Multiplexing Control 2 Register (PINMUX2)

Figure 10-20. Pin Multiplexing Control 2 Register (PINMUX2)

31 28	27 24	23 20	19 16
PINMUX2_31_28	PINMUX2_27_24	PINMUX2_23_20	PINMUX2_19_16
R/W-0	R/W-0	R/W-0	R/W-0
15 12	**11 8**	**7 4**	**3 0**
PINMUX2_15_12	PINMUX2_11_8	PINMUX2_7_4	PINMUX2_3_0
R/W-0	R/W-0	R/W-0	R/W-0

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

Table 10-24. Pin Multiplexing Control 2 Register (PINMUX2) Field Descriptions

Bit	Field	Value	Description	Type [(1)]
31-28	PINMUX2_31_28		**AXR0/ECAP0_APWM0/GP8[7]/CLKS0 Control**	
		0	Selects Function CLKS0	I
		1h	Selects Function AXR0	I/O
		2h	Selects Function ECAP0_APWM0	I/O
		3h	*Reserved*	X
		4h	Selects Function GP8[7]	I/O
		5h-Fh	*Reserved*	X
		27-24	PINMUX2_27_24	**AXR1/DX0/GP1[9] Control**
		0	Pin is 3-stated.	Z
		1h	Selects Function AXR1	I/O
		2h	Selects Function DX0	O
		3h	*Reserved*	X
		4h	Selects Function GP1[9]	I/O
		5h-Fh	*Reserved*	X
23-20	PINMUX2_23_20		**AXR2/DR0/GP1[10] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function AXR2	I/O
		2h	Selects Function DR0	I
		3h	*Reserved*	X
		4h	Selects Function GP1[10]	I/O
		5h-Fh	*Reserved*	X
19-16	PINMUX2_19_16		**AXR3/FSX0/GP1[11] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function AXR3	I/O
		2h	Selects Function FSX0	I/O
		3h	*Reserved*	X
		4h	Selects Function GP1[11]	I/O
		5h-Fh	*Reserved*	X

(1) I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-24. Pin Multiplexing Control 2 Register (PINMUX2) Field Descriptions (continued)

Bit	Field	Value	Description	Type (1)
15-12	PINMUX2_15_12		**AXR4/FSR0/GP1[12] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function AXR4	I/O
		2h	Selects Function FSR0	I/O
		3h	*Reserved*	X
		4h	Selects Function GP1[12]	I/O
		5h-Fh	*Reserved*	X
		11-8	PINMUX2_11_8	**AXR5/CLKX0/GP1[13] Control**
		0	Pin is 3-stated.	Z
		1h	Selects Function AXR5	I/O
		2h	Selects Function CLKX0	I/O
		3h	*Reserved*	X
		4h	Selects Function GP1[13]	I/O
		5h-Fh	*Reserved*	X
7-4	PINMUX2_7_4		**AXR6/CLKR0/GP1[14]/PRU0_R31[6] Control**	
		0	Selects Function PRU0_R31[6]	I
		1h	Selects Function AXR6	I/O
		2h	Selects Function CLKR0	I/O
		3h	*Reserved*	X
		4h	Selects Function GP1[14]	I/O
		5h-Fh	*Reserved*	X
		3-0	PINMUX2_3_0	**AXR7/EPWM1TZ[0]/PRU0_R30[17]/GP1[15]/PRU0_R31[7] Control**
		0	Selects Function PRU0_R31[7]	I
		1h	Selects Function AXR7	I/O
		2h	Selects Function EPWM1TZ[0]	I
		3h	*Reserved*	X
		4h	Selects Function PRU0_R30[17]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP1[15]	I/O
		9h-Fh	*Reserved*	X

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.10.4 Pin Multiplexing Control 3 Register (PINMUX3)

Figure 10-21. Pin Multiplexing Control 3 Register (PINMUX3)

31 28	27 24	23 20	19 16
PINMUX3_31_28	PINMUX3_27_24	PINMUX3_23_20	PINMUX3_19_16
R/W-0	R/W-0	R/W-0	R/W-0
15 12	**11 8**	**7 4**	**3 0**
PINMUX3_15_12	PINMUX3_11_8	PINMUX3_7_4	PINMUX3_3_0
R/W-0	R/W-0	R/W-0	R/W-0

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

Table 10-25. Pin Multiplexing Control 3 Register (PINMUX3) Field Descriptions

Bit	Field	Value	Description	Type[1]
31-28	PINMUX3_31_28		**$\overline{\text{SPI0_SCS[2]}}$/UART0_$\overline{\text{RTS}}$/GP8[1] Control**	
		0	*Reserved*	X
		1h	Selects Function $\overline{\text{SPI0_SCS[2]}}$	I/O
		2h	Selects Function $\overline{\text{UART0_RTS}}$	O
		3h	*Reserved*	X
		4h	Selects Function GP8[1]	I/O
		5h-Fh	*Reserved*	X
27-24	PINMUX3_27_24		**$\overline{\text{SPI0_SCS[3]}}$/UART0_$\overline{\text{CTS}}$/GP8[2] Control**	
		0	*Reserved*	X
		1h	Selects Function $\overline{\text{SPI0_SCS[3]}}$	I/O
		2h	Selects Function $\overline{\text{UART0_CTS}}$	I
		3h	*Reserved*	X
		4h	Selects Function GP8[2]	I/O
		5h-Fh	*Reserved*	X
23-20	PINMUX3_23_20		**$\overline{\text{SPI0_SCS[4]}}$/UART0_TXD/GP8[3] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{\text{SPI0_SCS[4]}}$	I/O
		2h	Selects Function UART0_TXD	O
		3h	*Reserved*	X
		4h	Selects Function GP8[3]	I/O
		5h-Fh	*Reserved*	X
19-16	PINMUX3_19_16		**$\overline{\text{SPI0_SCS[5]}}$/UART0_RXD/GP8[4] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{\text{SPI0_SCS[5]}}$	I/O
		2h	Selects Function UART0_RXD	I
		3h	*Reserved*	X
		4h	Selects Function GP8[4]	I/O
		5h-Fh	*Reserved*	X
15-12	PINMUX3_15_12		**SPI0_SIMO/EPWMSYNCO/GP8[5] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function SPI0_SIMO	I/O
		2h	Selects Function EPWMSYNCO	O
		3h	*Reserved*	X
		4h	Selects Function GP8[5]	I/O
		5h-Fh	*Reserved*	X

(1) I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-25. Pin Multiplexing Control 3 Register (PINMUX3) Field Descriptions (continued)

Bit	Field	Value	Description	Type[1]
11-8	PINMUX3_11_8		**SPI0_SOMI/EPWMSYNCI/GP8[6] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function SPI0_SOMI	I/O
		2h	Selects Function EPWMSYNCI	I
		3h	*Reserved*	X
		4h	Selects Function GP8[6]	I/O
		5h-Fh	*Reserved*	X
7-4	PINMUX3_7_4		**SPI0_ENA/EPWM0B/PRU0_R30[6] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function SPI0_ENA	I/O
		2h	Selects Function EPWM0B	I/O
		3h	*Reserved*	X
		4h	Selects Function PRU0_R30[6]	O
		5h-Fh	*Reserved*	X
3-0	PINMUX3_3_0		**SPI0_CLK/EPWM0A/GP1[8] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function SPI0_CLK	I/O
		2h	Selects Function EPWM0A	I/O
		3h	*Reserved*	X
		4h	Selects Function GP1[8]	I/O
		5h-Fh	*Reserved*	X

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.10.5 Pin Multiplexing Control 4 Register (PINMUX4)

Figure 10-22. Pin Multiplexing Control 4 Register (PINMUX4)

31 28	27 24	23 20	19 16
PINMUX4_31_28	PINMUX4_27_24	PINMUX4_23_20	PINMUX4_19_16
R/W-0	R/W-0	R/W-0	R/W-0
15 12	**11 8**	**7 4**	**3 0**
PINMUX4_15_12	PINMUX4_11_8	PINMUX4_7_4	PINMUX4_3_0
R/W-0	R/W-0	R/W-0	R/W-0

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

Table 10-26. Pin Multiplexing Control 4 Register (PINMUX4) Field Descriptions

Bit	Field	Value	Description	Type[1]
31-28	PINMUX4_31_28		**$\overline{\text{SP1_SCS[2]}}$/UART1_TXD/GP1[0] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{\text{SP1_SCS[2]}}$	I/O
		2h	Selects Function UART1_TXD	O
		3h-7h	*Reserved*	X
		8h	Selects Function GP1[0]	I/O
		9h-Fh	*Reserved*	X
27-24	PINMUX4_27_24		**$\overline{\text{SPI1_SCS[3]}}$/UART1_RXD/GP1[1] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{\text{SPI1_SCS[3]}}$	I/O
		2h	Selects Function UART1_RXD	I
		3h-7h	*Reserved*	X
		8h	Selects Function GP1[1]	I/O
		9h-Fh	*Reserved*	X
23-20	PINMUX4_23_20		**$\overline{\text{SPI1_SCS[4]}}$/UART2_TXD/I2C1_SDA/GP1[2] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{\text{SPI1_SCS[4]}}$	I/O
		2h	Selects Function UART2_TXD	O
		3h	*Reserved*	X
		4h	Selects Function I2C1_SDA	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP1[2]	I/O
		9h-Fh	*Reserved*	X
19-16	PINMUX4_19_16		**$\overline{\text{SPI1_SCS[5]}}$/UART2_RXD/I2C1_SCL/GP1[3] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{\text{SPI1_SCS[5]}}$	I/O
		2h	Selects Function UART2_RXD	I
		3h	*Reserved*	X
		4h	Selects Function I2C1_SCL	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP1[3]	I/O
		9h-Fh	*Reserved*	X

(1) I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-26. Pin Multiplexing Control 4 Register (PINMUX4) Field Descriptions (continued)

Bit	Field	Value	Description	Type[1]
15-12	PINMUX4_15_12		**$\overline{\text{SPI1_SCS[6]}}$/I2C0_SDA/TM64P3_OUT12/GP1[4] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{\text{SPI1_SCS[6]}}$	I/O
		2h	Selects Function I2C0_SDA	I/O
		3h	*Reserved*	X
		4h	Selects Function TM64P3_OUT12	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP1[4]	I/O
		9h-Fh	*Reserved*	X
11-8	PINMUX4_11_8		**$\overline{\text{SPI1_SCS[7]}}$/I2C0_SCL/TM64P2_OUT12/GP1[5] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{\text{SPI1_SCS[7]}}$	I/O
		2h	Selects Function I2C0_SCL	I/O
		3h	*Reserved*	X
		4h	Selects Function TM64P2_OUT12	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP1[5]	I/O
		9h-Fh	*Reserved*	X
7-4	PINMUX4_7_4		**$\overline{\text{SPI0_SCS[0]}}$/TM64P1_OUT12/GP1[6]/TM64P1_IN12 Control**	
		0	Selects Function TM64P1_IN12	I
		1h	Selects Function $\overline{\text{SPI0_SCS[0]}}$	I/O
		2h	Selects Function TM64P1_OUT12	O
		3h	*Reserved*	X
		4h	Selects Function GP1[6]	I/O
		5h-Fh	*Reserved*	X
3-0	PINMUX4_3_0		**$\overline{\text{SPI0_SCS[1]}}$/TM64P0_OUT12/GP1[7]/TM64P0_IN12 Control**	
		0	Selects Function TM64P0_IN12	I
		1h	Selects Function $\overline{\text{SPI0_SCS[1]}}$	I/O
		2h	Selects Function TM64P0_OUT12	O
		3h	*Reserved*	X
		4h	Selects Function GP1[7]	I/O
		5h-Fh	*Reserved*	X

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.10.6 Pin Multiplexing Control 5 Register (PINMUX5)

Figure 10-23. Pin Multiplexing Control 5 Register (PINMUX5)

31 28	27 24	23 20	19 16
PINMUX5_31_28	PINMUX5_27_24	PINMUX5_23_20	PINMUX5_19_16
R/W-0	R/W-0	R/W-0	R/W-0
15 12	**11 8**	**7 4**	**3 0**
PINMUX5_15_12	PINMUX5_11_8	PINMUX5_7_4	PINMUX5_3_0
R/W-0	R/W-0	R/W-0	R/W-0

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

Table 10-27. Pin Multiplexing Control 5 Register (PINMUX5) Field Descriptions

Bit	Field	Value	Description	Type[1]
31-28	PINMUX5_31_28		**EMA_BA[0]/GP2[8] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_BA[0]	O
		2h-7h	*Reserved*	X
		8h	Selects Function GP2[8]	I/O
		9h-Fh	*Reserved*	X
27-24	PINMUX5_27_24		**EMA_BA[1]/GP2[9] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_BA[1]	O
		2h-7h	*Reserved*	X
		8h	Selects Function GP2[9]	I/O
		9h-Fh	*Reserved*	X
23-20	PINMUX5_23_20		**SPI1_SIMO/GP2[10] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function SPI1_SIMO	I/O
		2h-7h	*Reserved*	X
		8h	Selects Function GP2[10]	I/O
		9h-Fh	*Reserved*	X
19-16	PINMUX5_19_16		**SPI1_SOMI/GP2[11] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function SPI1_SOMI	I/O
		2h-7h	*Reserved*	X
		8h	Selects Function GP2[11]	I/O
		9h-Fh	*Reserved*	X
15-12	PINMUX5_15_12		**$\overline{\text{SPI1_ENA}}$/GP2[12] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{\text{SPI1_ENA}}$	I/O
		2h-7h	*Reserved*	X
		8h	Selects Function GP2[12]	I/O
		9h-Fh	*Reserved*	X

(1) I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-27. Pin Multiplexing Control 5 Register (PINMUX5) Field Descriptions (continued)

Bit	Field	Value	Description	Type[1]
11-8	PINMUX5_11_8		**SPI1_CLK/GP2[13] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function SPI1_CLK	I/O
		2h-7h	*Reserved*	X
		8h	Selects Function GP2[13]	I/O
		9h-Fh	*Reserved*	X
7-4	PINMUX5_7_4		**$\overline{\text{SPI1_SCS[0]}}$/EPWM1B/PRU0_R30[7]/GP2[14]/TM64P3_IN12 Control**	
		0	Selects Function TM64P3_IN12	I
		1h	Selects Function $\overline{\text{SPI1_SCS[0]}}$	I/O
		2h	Selects Function EPWM1B	I/O
		3h	*Reserved*	X
		4h	Selects Function PRU0_R30[7]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP2[14]	I/O
		9h-Fh	*Reserved*	X
3-0	PINMUX5_3_0		**$\overline{\text{SPI1_SCS[1]}}$/EPWM1A/PRU0_R30[8]/GP2[15]/TM64P2_IN12 Control**	
		0	Selects Function TM64P2_IN12	I
		1h	Selects Function $\overline{\text{SPI1_SCS[1]}}$	I/O
		2h	Selects Function EPWM1A	I/O
		3h	*Reserved*	X
		4h	Selects Function PRU0_R30[8]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP2[15]	I/O
		9h-Fh	*Reserved*	X

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.10.7 Pin Multiplexing Control 6 Register (PINMUX6)

Figure 10-24. Pin Multiplexing Control 6 Register (PINMUX6)

31 28	27 24	23 20	19 16
PINMUX6_31_28	PINMUX6_27_24	PINMUX6_23_20	PINMUX6_19_16
R/W-0	R/W-0	R/W-0	R/W-0
15 12	**11 8**	**7 4**	**3 0**
PINMUX6_15_12	PINMUX6_11_8	PINMUX6_7_4	PINMUX6_3_0
R/W-0	R/W-0	R/W-0	R/W-0

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

Table 10-28. Pin Multiplexing Control 6 Register (PINMUX6) Field Descriptions

Bit	Field	Value	Description	Type[1]
31-28	PINMUX6_31_28		**$\overline{\text{EMA_CS[0]}}$/GP2[0] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{\text{EMA_CS[0]}}$	O
		2h-7h	*Reserved*	X
		8h	Selects Function GP2[0]	I/O
		9h-Fh	*Reserved*	X
27-24	PINMUX6_27_24		**EMA_WAIT[1]/PRU0_R30[1]/GP2[1]/PRU0_R31[1] Control**	
		0	Selects Function PRU0_R31[1]	I
		1h	Selects Function EMA_WAIT[1]	I
		2h-3h	*Reserved*	X
		4h	Selects Function PRU0_R30[1]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP2[1]	I/O
		9h-Fh	*Reserved*	X
23-20	PINMUX6_23_20		**$\overline{\text{EMA_WE_DQM[1]}}$/GP2[2] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{\text{EMA_WE_DQM[1]}}$	O
		2h-7h	*Reserved*	X
		8h	Selects Function GP2[2]	I/O
		9h-Fh	*Reserved*	X
19-16	PINMUX6_19_16		**$\overline{\text{EMA_WE_DQM[0]}}$/GP2[3] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{\text{EMA_WE_DQM[0]}}$	O
		2h-7h	*Reserved*	X
		8h	Selects Function GP2[3]	I/O
		9h-Fh	*Reserved*	X
15-12	PINMUX6_15_12		**$\overline{\text{EMA_CAS}}$/PRU0_R30[2]/GP2[4]/PRU0_R31[2] Control**	
		0	Selects Function PRU0_R31[2]	I
		1h	Selects Function $\overline{\text{EMA_CAS}}$	O
		2h-3h	*Reserved*	X
		4h	Selects Function PRU0_R30[2]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP2[4]	I/O
		9h-Fh	*Reserved*	X

(1) I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-28. Pin Multiplexing Control 6 Register (PINMUX6) Field Descriptions (continued)

Bit	Field	Value	Description	Type[1]
11-8	PINMUX6_11_8		**$\overline{\text{EMA_RAS}}$/PRU0_R30[3]/GP2[5]/PRU0_R31[3] Control**	
		0	Selects Function PRU0_R31[3]	I
		1h	Selects Function $\overline{\text{EMA_RAS}}$	O
		2h-3h	*Reserved*	X
		4h	Selects Function PRU0_R30[3]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP2[5]	I/O
		9h-Fh	*Reserved*	X
7-4	PINMUX6_7_4		**EMA_SDCKE/PRU0_R30[4]/GP2[6]/PRU0_R31[4] Control**	
		0	Selects Function PRU0_R31[4]	I
		1h	Selects Function EMA_SDCKE	O
		2h-3h	*Reserved*	X
		4h	Selects Function PRU0_R30[4]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP2[6]	I/O
		9h-Fh	*Reserved*	X
3-0	PINMUX6_3_0		**EMA_CLK/PRU0_R30[5]/GP2[7]/PRU0_R31[5] Control**	
		0	Selects Function PRU0_R31[5]	I
		1h	Selects Function EMA_CLK	O
		2h-3h	*Reserved*	X
		4h	Selects Function PRU0_R30[5]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP2[7]	I/O
		9h-Fh	*Reserved*	X

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.10.8 Pin Multiplexing Control 7 Register (PINMUX7)

Figure 10-25. Pin Multiplexing Control 7 Register (PINMUX7)

31 28	27 24	23 20	19 16
PINMUX7_31_28	PINMUX7_27_24	PINMUX7_23_20	PINMUX7_19_16
R/W-0	R/W-0	R/W-0	R/W-0
15 12	**11 8**	**7 4**	**3 0**
PINMUX7_15_12	PINMUX7_11_8	PINMUX7_7_4	PINMUX7_3_0
R/W-0	R/W-0	R/W-0	R/W-0

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

Table 10-29. Pin Multiplexing Control 7 Register (PINMUX7) Field Descriptions

Bit	Field	Value	Description	Type[1]
31-28	PINMUX7_31_28		**EMA_WAIT[0]/PRU0_R30[0]/GP3[8]/PRU0_R31[0] Control**	
		0	Selects Function PRU0_R31[0]	I
		1h	Selects Function EMA_WAIT[0]	I
		2h-3h	*Reserved*	X
		4h	Selects Function PRU0_R30[0]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP3[8]	I/O
		9h-Fh	*Reserved*	X
27-24	PINMUX7_27_24		**EMA_A_$\overline{RW}$/GP3[9] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A_$\overline{RW}$	O
		2h-7h	*Reserved*	X
		8h	Selects Function GP3[9]	I/O
		9h-Fh	*Reserved*	X
23-20	PINMUX7_23_20		**$\overline{EMA_OE}$/GP3[10] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{EMA_OE}$	O
		2h-7h	*Reserved*	X
		8h	Selects Function GP3[10]	I/O
		9h-Fh	*Reserved*	X
19-16	PINMUX7_19_16		**$\overline{EMA_WE}$/GP3[11] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{EMA_WE}$	O
		2h-7h	*Reserved*	X
		8h	Selects Function GP3[11]	I/O
		9h-Fh	*Reserved*	X
15-12	PINMUX7_15_12		**$\overline{EMA_CS[5]}$/GP3[12] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{EMA_CS[5]}$	O
		2h-7h	*Reserved*	X
		8h	Selects Function GP3[12]	I/O
		9h-Fh	*Reserved*	X

(1) I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-29. Pin Multiplexing Control 7 Register (PINMUX7) Field Descriptions (continued)

Bit	Field	Value	Description	Type[1]
11-8	PINMUX7_11_8		**$\overline{\text{EMA_CS[4]}}$/GP3[13] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{\text{EMA_CS[4]}}$	O
		2h-7h	*Reserved*	X
		8h	Selects Function GP3[13]	I/O
		9h-Fh	*Reserved*	X
7-4	PINMUX7_7_4		**$\overline{\text{EMA_CS[3]}}$/GP3[14] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{\text{EMA_CS[3]}}$	O
		2h-7h	*Reserved*	X
		8h	Selects Function GP3[14]	I/O
		9h-Fh	*Reserved*	X
3-0	PINMUX7_3_0		**$\overline{\text{EMA_CS[2]}}$/GP3[15] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function $\overline{\text{EMA_CS[2]}}$	O
		2h-7h	*Reserved*	X
		8h	Selects Function GP3[15]	I/O
		9h-Fh	*Reserved*	X

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.10.9 Pin Multiplexing Control 8 Register (PINMUX8)

Figure 10-26. Pin Multiplexing Control 8 Register (PINMUX8)

31 28	27 24	23 20	19 16
PINMUX8_31_28	PINMUX8_27_24	PINMUX8_23_20	PINMUX8_19_16
R/W-0	R/W-0	R/W-0	R/W-0
15 12	**11 8**	**7 4**	**3 0**
PINMUX8_15_12	PINMUX8_11_8	PINMUX8_7_4	PINMUX8_3_0
R/W-0	R/W-0	R/W-0	R/W-0

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

Table 10-30. Pin Multiplexing Control 8 Register (PINMUX8) Field Descriptions

Bit	Field	Value	Description	Type[1]
31-28	PINMUX8_31_28		**EMA_D[8]/GP3[0] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_D[8]	I/O
		2h-7h	*Reserved*	X
		8h	Selects Function GP3[0]	I/O
		9h-Fh	*Reserved*	X
27-24	PINMUX8_27_24		**EMA_D[9]/GP3[1] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_D[9]	I/O
		2h-7h	*Reserved*	X
		8h	Selects Function GP3[1]	I/O
		9h-Fh	*Reserved*	X
23-20	PINMUX8_23_20		**EMA_D[10]/GP3[2] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_D[10]	I/O
		2h-7h	*Reserved*	X
		8h	Selects Function GP3[2]	I/O
		9h-Fh	*Reserved*	X
19-16	PINMUX8_19_16		**EMA_D[11]/GP3[3] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_D[11]	I/O
		2h-7h	*Reserved*	X
		8h	Selects Function GP3[3]	I/O
		9h-Fh	*Reserved*	X
15-12	PINMUX8_15_12		**EMA_D[12]/GP3[4] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_D[12]	I/O
		2h-7h	*Reserved*	X
		8h	Selects Function GP3[4]	I/O
		9h-Fh	*Reserved*	X

(1) I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-30. Pin Multiplexing Control 8 Register (PINMUX8) Field Descriptions (continued)

Bit	Field	Value	Description	Type[1]
11-8	PINMUX8_11_8		**EMA_D[13]/GP3[5] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_D[13]	I/O
		2h-7h	*Reserved*	X
		8h	Selects Function GP3[5]	I/O
		9h-Fh	*Reserved*	X
7-4	PINMUX8_7_4		**EMA_D[14]/GP3[6] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_D[14]	I/O
		2h-7h	*Reserved*	X
		8h	Selects Function GP3[6]	I/O
		9h-Fh	*Reserved*	X
3-0	PINMUX8_3_0		**EMA_D[15]/GP3[7] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_D[15]	I/O
		2h-7h	*Reserved*	X
		8h	Selects Function GP3[7]	I/O
		9h-Fh	*Reserved*	X

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.10.10 Pin Multiplexing Control 9 Register (PINMUX9)

Figure 10-27. Pin Multiplexing Control 9 Register (PINMUX9)

31 28	27 24	23 20	19 16
PINMUX9_31_28	PINMUX9_27_24	PINMUX9_23_20	PINMUX9_19_16
R/W-0	R/W-0	R/W-0	R/W-0
15 12	**11 8**	**7 4**	**3 0**
PINMUX9_15_12	PINMUX9_11_8	PINMUX9_7_4	PINMUX9_3_0
R/W-0	R/W-0	R/W-0	R/W-0

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

Table 10-31. Pin Multiplexing Control 9 Register (PINMUX9) Field Descriptions

Bit	Field	Value	Description	Type[1]
31-28	PINMUX9_31_28		**EMA_D[0]/GP4[8] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_D[0]	I/O
		2h-7h	*Reserved*	X
		8h	Selects Function GP4[8]	I/O
		9h-Fh	*Reserved*	X
27-24	PINMUX9_27_24		**EMA_D[1]/GP4[9] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_D[1]	I/O
		2h-7h	*Reserved*	X
		8h	Selects Function GP4[9]	I/O
		9h-Fh	*Reserved*	X
23-20	PINMUX9_23_20		**EMA_D[2]/GP4[10] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_D[2]	I/O
		2h-7h	*Reserved*	X
		8h	Selects Function GP4[10]	I/O
		9h-Fh	*Reserved*	X
19-16	PINMUX9_19_16		**EMA_D[3]/GP4[11] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_D[3]	I/O
		2h-7h	*Reserved*	X
		8h	Selects Function GP4[11]	I/O
		9h-Fh	*Reserved*	X
15-12	PINMUX9_15_12		**EMA_D[4]/GP4[12] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_D[4]	I/O
		2h-7h	*Reserved*	X
		8h	Selects Function GP4[12]	I/O
		9h-Fh	*Reserved*	X

(1) I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-31. Pin Multiplexing Control 9 Register (PINMUX9) Field Descriptions (continued)

Bit	Field	Value	Description	Type[1]
11-8	PINMUX9_11_8		**EMA_D[5]/GP4[13] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_D[5]	I/O
		2h-7h	*Reserved*	X
		8h	Selects Function GP4[13]	I/O
		9h-Fh	*Reserved*	X
7-4	PINMUX9_7_4		**EMA_D[6]/GP4[14] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_D[6]	I/O
		2h-7h	*Reserved*	X
		8h	Selects Function GP4[14]	I/O
		9h-Fh	*Reserved*	X
3-0	PINMUX9_3_0		**EMA_D[7]/GP4[15] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_D[7]	I/O
		2h-7h	*Reserved*	X
		8h	Selects Function GP4[15]	I/O
		9h-Fh	*Reserved*	X

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.10.11 Pin Multiplexing Control 10 Register (PINMUX10)

Figure 10-28. Pin Multiplexing Control 10 Register (PINMUX10)

31 28	27 24	23 20	19 16
PINMUX10_31_28	PINMUX10_27_24	PINMUX10_23_20	PINMUX10_19_16
R/W-0	R/W-0	R/W-0	R/W-0
15 12	**11 8**	**7 4**	**3 0**
PINMUX10_15_12	PINMUX10_11_8	PINMUX10_7_4	PINMUX10_3_0
R/W-0	R/W-0	R/W-0	R/W-0

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

Table 10-32. Pin Multiplexing Control 10 Register (PINMUX10) Field Descriptions

Bit	Field	Value	Description	Type[1]
31-28	PINMUX10_31_28		**EMA_A[16]/MMCSD0_DAT[5]/PRU1_R30[24]/GP4[0] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[16]	O
		2h	Selects Function MMCSD0_DAT[5]	I/O
		3h	*Reserved*	X
		4h	Selects Function PRU1_R30[24]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP4[0]	I/O
		9h-Fh	*Reserved*	X
27-24	PINMUX10_27_24		**EMA_A[17]/MMCSD0_DAT[4]/PRU1_R30[25]/GP4[1] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[17]	O
		2h	Selects Function MMCSD0_DAT[4]	I/O
		3h	*Reserved*	X
		4h	Selects Function PRU1_R30[25]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP4[1]	I/O
		9h-Fh	*Reserved*	X
23-20	PINMUX10_23_20		**EMA_A[18]/MMCSD0_DAT[3]/PRU1_R30[26]/GP4[2] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[18]	O
		2h	Selects Function MMCSD0_DAT[3]	I/O
		3h	*Reserved*	X
		4h	Selects Function PRU1_R30[26]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP4[2]	I/O
		9h-Fh	*Reserved*	X

(1) I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-32. Pin Multiplexing Control 10 Register (PINMUX10) Field Descriptions (continued)

Bit	Field	Value	Description	Type[1]
19-16	PINMUX10_19_16		**EMA_A[19]/MMCSD0_DAT[2]/PRU1_R30[27]/GP4[3] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[19]	O
		2h	Selects Function MMCSD0_DAT[2]	I/O
		3h	*Reserved*	X
		4h	Selects Function PRU1_R30[27]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP4[3]	I/O
		9h-Fh	*Reserved*	X
15-12	PINMUX10_15_12		**EMA_A[20]/MMCSD0_DAT[1]/PRU1_R30[28]/GP4[4] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[20]	O
		2h	Selects Function MMCSD0_DAT[1]	I/O
		3h	*Reserved*	X
		4h	Selects Function PRU1_R30[28]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP4[4]	I/O
		9h-Fh	*Reserved*	X
11-8	PINMUX10_11_8		**EMA_A[21]/MMCSD0_DAT[0]/PRU1_R30[29]/GP4[5] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[21]	O
		2h	Selects Function MMCSD0_DAT[0]	I/O
		3h	*Reserved*	X
		4h	Selects Function PRU1_R30[29]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP4[5]	I/O
		9h-Fh	*Reserved*	X
7-4	PINMUX10_7_4		**EMA_A[22]/MMCSD0_CMD/PRU1_R30[30]/GP4[6] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[22]	O
		2h	Selects Function MMCSD0_CMD	I/O
		3h	*Reserved*	X
		4h	Selects Function PRU1_R30[30]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP4[6]	I/O
		9h-Fh	*Reserved*	X
3-0	PINMUX10_3_0		**MMCSD0_CLK/PRU1_R30[31]/GP4[7] Control**	
		0	Pin is 3-stated.	Z
		1h	*Reserved*	X
		2h	Selects Function MMCSD0_CLK	O
		3h	*Reserved*	X
		4h	Selects Function PRU1_R30[31]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP4[7]	I/O
		9h-Fh	*Reserved*	X

10.5.10.12 Pin Multiplexing Control 11 Register (PINMUX11)

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 10-29. Pin Multiplexing Control 11 Register (PINMUX11)

31 28	27 24	23 20	19 16
PINMUX11_31_28	PINMUX11_27_24	PINMUX11_23_20	PINMUX11_19_16
R/W-0	R/W-0	R/W-0	R/W-0
15 12	**11 8**	**7 4**	**3 0**
PINMUX11_15_12	PINMUX11_11_8	PINMUX11_7_4	PINMUX11_3_0
R/W-0	R/W-0	R/W-0	R/W-0

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

Table 10-33. Pin Multiplexing Control 11 Register (PINMUX11) Field Descriptions

Bit	Field	Value	Description	Type[1]
31-28	PINMUX11_31_28		**EMA_A[8]/PRU1_R30[16]/GP5[8] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[8]	O
		2h-3h	*Reserved*	X
		4h	Selects Function PRU1_R30[16]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP5[8]	I/O
		9h-Fh	*Reserved*	X
27-24	PINMUX11_27_24		**EMA_A[9]/PRU1_R30[17]/GP5[9] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[9]	O
		2h-3h	*Reserved*	X
		4h	Selects Function PRU1_R30[17]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP5[9]	I/O
		9h-Fh	*Reserved*	X
23-20	PINMUX11_23_20		**EMA_A[10]/PRU1_R30[18]/GP5[10]/PRU1_R31[18] Control**	
		0	Selects Function PRU1_R31[18]	I
		1h	Selects Function EMA_A[10]	O
		2h-3h	*Reserved*	X
		4h	Selects Function PRU1_R30[18]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP5[10]	I/O
		9h-Fh	*Reserved*	X
19-16	PINMUX11_19_16		**EMA_A[11]/PRU1_R30[19]/GP5[11]/PRU1_R31[19] Control**	
		0	Selects Function PRU1_R31[19]	I
		1h	Selects Function EMA_A[11]	O
		2h-3h	*Reserved*	X
		4h	Selects Function PRU1_R30[19]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP5[11]	I/O
		9h-Fh	*Reserved*	X

(1) I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-33. Pin Multiplexing Control 11 Register (PINMUX11) Field Descriptions (continued)

Bit	Field	Value	Description	Type[1]
15-12	PINMUX11_15_12		**EMA_A[12]/PRU1_R30[20]/GP5[12]/PRU1_R31[20] Control**	
		0	Selects Function PRU1_R31[20]	I
		1h	Selects Function EMA_A[12]	O
		2h-3h	*Reserved*	X
		4h	Selects Function PRU1_R30[20]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP5[12]	I/O
		9h-Fh	*Reserved*	X
11-8	PINMUX11_11_8		**EMA_A[13]/PRU0_R30[21]/PRU1_R30[21]/GP5[13]/PRU1_R31[21] Control**	
		0	Selects Function PRU1_R31[21]	I
		1h	Selects Function EMA_A[13]	O
		2h	Selects Function PRU0_R30[21]	O
		3h	*Reserved*	X
		4h	Selects Function PRU1_R30[21]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP5[13]	I/O
		9h-Fh	*Reserved*	X
7-4	PINMUX11_7_4		**EMA_A[14]/MMCSD0_DAT[7]/PRU1_R30[22]/GP5[14]/PRU1_R31[22] Control**	
		0	Selects Function PRU1_R31[22]	I
		1h	Selects Function EMA_A[14]	O
		2h	Selects Function MMCSD0_DAT[7]	I/O
		3h	*Reserved*	X
		4h	Selects Function PRU1_R30[22]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP5[14]	I/O
		9h-Fh	*Reserved*	X
3-0	PINMUX11_3_0		**EMA_A[15]/MMCSD0_DAT[6]/PRU1_R30[23]/GP5[15]/PRU1_R31[23] Control**	
		0	Selects Function PRU1_R31[23]	I
		1h	Selects Function EMA_A[15]	O
		2h	Selects Function MMCSD0_DAT[6]	I/O
		3h	*Reserved*	X
		4h	Selects Function PRU1_R30[23]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP5[15]	I/O
		9h-Fh	*Reserved*	X

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.10.13 Pin Multiplexing Control 12 Register (PINMUX12)

Figure 10-30. Pin Multiplexing Control 12 Register (PINMUX12)

31 28	27 24	23 20	19 16
PINMUX12_31_28	PINMUX12_27_24	PINMUX12_23_20	PINMUX12_19_16
R/W-0	R/W-0	R/W-0	R/W-0
15 12	**11 8**	**7 4**	**3 0**
PINMUX12_15_12	PINMUX12_11_8	PINMUX12_7_4	PINMUX12_3_0
R/W-0	R/W-0	R/W-0	R/W-0

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

Table 10-34. Pin Multiplexing Control 12 Register (PINMUX12) Field Descriptions

Bit	Field	Value	Description	Type[1]
31-28	PINMUX12_31_28		**EMA_A[0]/GP5[0] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[0]	O
		2h-7h	*Reserved*	X
		8h	Selects Function GP5[0]	I/O
		9h-Fh	*Reserved*	X
27-24	PINMUX12_27_24		**EMA_A[1]/GP5[1] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[1]	O
		2h-7h	*Reserved*	X
		8h	Selects Function GP5[1]	I/O
		9h-Fh	*Reserved*	X
23-20	PINMUX12_23_20		**EMA_A[2]/GP5[2] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[2]	O
		2h-7h	*Reserved*	X
		8h	Selects Function GP5[2]	I/O
		9h-Fh	*Reserved*	X
19-16	PINMUX12_19_16		**EMA_A[3]/GP5[3] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[3]	O
		2h-7h	*Reserved*	X
		8h	Selects Function GP5[3]	I/O
		9h-Fh	*Reserved*	X
15-12	PINMUX12_15_12		**EMA_A[4]/GP5[4] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[4]	O
		2h-7h	*Reserved*	X
		8h	Selects Function GP5[4]	I/O
		9h-Fh	*Reserved*	X

(1) I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-34. Pin Multiplexing Control 12 Register (PINMUX12) Field Descriptions (continued)

Bit	Field	Value	Description	Type[1]
11-8	PINMUX12_11_8		**EMA_A[5]/GP5[5] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[5]	O
		2h-7h	*Reserved*	X
		8h	Selects Function GP5[5]	I/O
		9h-Fh	*Reserved*	X
7-4	PINMUX12_7_4		**EMA_A[6]/GP5[6] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[6]	O
		2h-7h	*Reserved*	X
		8h	Selects Function GP5[6]	I/O
		9h-Fh	*Reserved*	X
3-0	PINMUX12_3_0		**EMA_A[7]/PRU1_R30[15]/GP5[7] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function EMA_A[7]	O
		2h-3h	*Reserved*	X
		4h	Selects Function PRU1_R30[15]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP5[7]	I/O
		9h-Fh	*Reserved*	X

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.10.14 Pin Multiplexing Control 13 Register (PINMUX13)

Figure 10-31. Pin Multiplexing Control 13 Register (PINMUX13)

31 28	27 24	23 20	19 16
PINMUX13_31_28	PINMUX13_27_24	PINMUX13_23_20	PINMUX13_19_16
R/W-0	R/W-0	R/W-0	R/W-0
15 12	**11 8**	**7 4**	**3 0**
PINMUX13_15_12	PINMUX13_11_8	PINMUX13_7_4	PINMUX13_3_0
R/W-0	R/W-0	R/W-0	R/W-0

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

Table 10-35. Pin Multiplexing Control 13 Register (PINMUX13) Field Descriptions

Bit	Field	Value	Description	Type[1]
31-28	PINMUX13_31_28		**PRU0_R30[26]/ UHPI_HR$\overline{W}$/UPP_CHA_WAIT/GP6[8]/PRU1_R31[17] Control**	
		0	Selects Function PRU1_R31[17]	I
		1h	Selects Function PRU0_R30[26]	O
		2h	Selects Function UHPI_HR$\overline{W}$	I
		3h	*Reserved*	X
		4h	Selects Function UPP_CHA_WAIT	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP6[8]	I/O
		9h-Fh	*Reserved*	X
27-24	PINMUX13_27_24		**PRU0_R30[27]/UHPI_HHWIL/UPP_CHA_ENABLE/GP6[9] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function PRU0_R30[27]	O
		2h	Selects Function UHPI_HHWIL	I
		3h	*Reserved*	X
		4h	Selects Function UPP_CHA_ENABLE	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP6[9]	I/O
		9h-Fh	*Reserved*	X
23-20	PINMUX13_23_20		**PRU0_R30[28]/UHPI_HCNTL1/UPP_CHA_START/GP6[10] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function PRU0_R30[28]	O
		2h	Selects Function UHPI_HCNTL1	I
		3h	*Reserved*	X
		4h	Selects Function UPP_CHA_START	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP6[10]	I/O
		9h-Fh	*Reserved*	X

(1) I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-35. Pin Multiplexing Control 13 Register (PINMUX13) Field Descriptions (continued)

Bit	Field	Value	Description	Type[1]
19-16	PINMUX13_19_16		**PRU0_R30[29]/UHPI_HCNTL0/UPP_CHA_CLOCK/GP6[11] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function PRU0_R30[29]	O
		2h	Selects Function UHPI_HCNTL0	I
		3h	*Reserved*	X
		4h	Selects Function UPP_CHA_CLOCK	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP6[11]	I/O
		9h-Fh	*Reserved*	X
15-12	PINMUX13_15_12		**PRU0_R30[30]/$\overline{\text{UHPI_HINT}}$/PRU1_R30[11]/GP6[12] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function PRU0_R30[30]	O
		2h	Selects Function $\overline{\text{UHPI_HINT}}$	O
		3h	*Reserved*	X
		4h	Selects Function PRU1_R30[11]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP6[12]	I/O
		9h-Fh	*Reserved*	X
11-8	PINMUX13_11_8		**PRU0_R30[31]/$\overline{\text{UHPI_HRDY}}$/PRU1_R30[12]/GP6[13] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function PRU0_R30[31]	O
		2h	Selects Function $\overline{\text{UHPI_HRDY}}$	O
		3h	*Reserved*	X
		4h	Selects Function PRU1_R30[12]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP6[13]	I/O
		9h-Fh	*Reserved*	X
7-4	PINMUX13_7_4		**CLKOUT/$\overline{\text{UHPI_HDS2}}$/PRU1_R30[13]/GP6[14] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function CLKOUT	O
		2h	Selects Function $\overline{\text{UHPI_HDS2}}$	I
		3h	*Reserved*	X
		4h	Selects Function PRU1_R30[13]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP6[14]	I/O
		9h-Fh	*Reserved*	X
3-0	PINMUX13_3_0		**$\overline{\text{RESETOUT}}$/$\overline{\text{UHPI_HAS}}$/PRU1_R30[14]/GP6[15] Control**	
		0	Selects Function $\overline{\text{RESETOUT}}$	O
		1h	Selects Function $\overline{\text{RESETOUT}}$	O
		2h	Selects Function $\overline{\text{UHPI_HAS}}$	I
		3h	*Reserved*	X
		4h	Selects Function PRU1_R30[14]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP6[15]	I/O
		9h-Fh	*Reserved*	X

10.5.10.15 Pin Multiplexing Control 14 Register (PINMUX14)

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 10-32. Pin Multiplexing Control 14 Register (PINMUX14)

31 28	27 24	23 20	19 16
PINMUX14_31_28	PINMUX14_27_24	PINMUX14_23_20	PINMUX14_19_16
R/W-0	R/W-0	R/W-0	R/W-0
15 12	**11 8**	**7 4**	**3 0**
PINMUX14_15_12	PINMUX14_11_8	PINMUX14_7_4	PINMUX14_3_0
R/W-0	R/W-0	R/W-0	R/W-0

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

Table 10-36. Pin Multiplexing Control 14 Register (PINMUX14) Field Descriptions

Bit	Field	Value	Description	Type[1]
31-28	PINMUX14_31_28		**VP_DIN[2]/UHPI_HD[10]/UPP_D[10]/PRU0_R31[24] Control**	
		0	Selects Function PRU0_R31[24]	I
		1h	Selects Function VP_DIN[2]	I
		2h	Selects Function UHPI_HD[10]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_D[10]	I/O
		5h-Fh	*Reserved*	X
27-24	PINMUX14_27_24		**VP_DIN[3]/UHPI_HD[11]/UPP_D[11]/PRU0_R31[25] Control**	
		0	Selects Function PRU0_R31[25]	I
		1h	Selects Function VP_DIN[3]	I
		2h	Selects Function UHPI_HD[11]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_D[11]	I/O
		5h-Fh	*Reserved*	X
23-20	PINMUX14_23_20		**VP_DIN[4]/UHPI_HD[12]/UPP_D[12]/PRU0_R31[26] Control**	
		0	Selects Function PRU0_R31[26]	I
		1h	Selects Function VP_DIN[4]	I
		2h	Selects Function UHPI_HD[12]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_D[12]	I/O
		5h-Fh	*Reserved*	X
19-16	PINMUX14_19_16		**VP_DIN[5]/UHPI_HD[13]/UPP_D[13]/PRU0_R31[27] Control**	
		0	Selects Function PRU0_R31[27]	I
		1h	Selects Function VP_DIN[5]	I
		2h	Selects Function UHPI_HD[13]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_D[13]	I/O
		5h-Fh	*Reserved*	X
15-12	PINMUX14_15_12		**VP_DIN[6]/UHPI_HD[14]/UPP_D[14]/PRU0_R31[28] Control**	
		0	Selects Function PRU0_R31[28]	I
		1h	Selects Function VP_DIN[6]	I
		2h	Selects Function UHPI_HD[14]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_D[14]	I/O
		5h-Fh	*Reserved*	X

(1) I = Input, O = Output, I/O = Bidirectional, X = Undefined

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-36. Pin Multiplexing Control 14 Register (PINMUX14) Field Descriptions (continued)

Bit	Field	Value	Description	Type[1]
11-8	PINMUX14_11_8		**VP_DIN[7]/UHPI_HD[15]/UPP_D[15]/PRU0_R31[29] Control**	
		0	Selects Function PRU0_R31[29]	I
		1h	Selects Function VP_DIN[7]	I
		2h	Selects Function UHPI_HD[15]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_D[15]	I/O
		5h-Fh	*Reserved*	X
7-4	PINMUX14_7_4		**VP_CLKIN1/$\overline{\text{UHPI_HDS1}}$/PRU1_R30[9]/GP6[6]/PRU1_R31[16] Control**	
		0	Selects Function PRU1_R31[16]	I
		1h	Selects Function VP_CLKIN1	I
		2h	Selects Function $\overline{\text{UHPI_HDS1}}$	I
		3h	*Reserved*	X
		4h	Selects Function PRU1_R30[9]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP6[6]	I/O
		9h-Fh	*Reserved*	X
3-0	PINMUX14_3_0		**VP_CLKIN0/$\overline{\text{UHPI_HCS}}$/PRU1_R30[10]/GP6[7]/UPP_2xTXCLK Control**	
		0	Selects Function UPP_2xTXCLK	I
		1h	Selects Function VP_CLKIN0	I
		2h	Selects Function $\overline{\text{UHPI_HCS}}$	I
		3h	*Reserved*	X
		4h	Selects Function PRU1_R30[10]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP6[7]	I/O
		9h-Fh	*Reserved*	X

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.10.16 Pin Multiplexing Control 15 Register (PINMUX15)

Figure 10-33. Pin Multiplexing Control 15 Register (PINMUX15)

31 28	27 24	23 20	19 16
PINMUX15_31_28	PINMUX15_27_24	PINMUX15_23_20	PINMUX15_19_16
R/W-0	R/W-0	R/W-0	R/W-0
15 12	**11 8**	**7 4**	**3 0**
PINMUX15_15_12	PINMUX15_11_8	PINMUX15_7_4	PINMUX15_3_0
R/W-0	R/W-0	R/W-0	R/W-0

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

Table 10-37. Pin Multiplexing Control 15 Register (PINMUX15) Field Descriptions

Bit	Field	Value	Description	Type[1]
31-28	PINMUX15_31_28		**VP_DIN[10]/UHPI_HD[2]/UPP_D[2]/PRU0_R30[10]/PRU0_R31[10] Control**	
		0	Selects Function PRU0_R31[10]	I
		1h	Selects Function VP_DIN[10]	I
		2h	Selects Function UHPI_HD[2]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_D[2]	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function PRU0_R30[10]	O
		9h-Fh	*Reserved*	X
27-24	PINMUX15_27_24		**VP_DIN[11]/UHPI_HD[3]/UPP_D[3]/PRU0_R30[11]/PRU0_R31[11] Control**	
		0	Selects Function PRU0_R31[11]	I
		1h	Selects Function VP_DIN[11]	I
		2h	Selects Function UHPI_HD[3]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_D[3]	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function PRU0_R30[11]	O
		9h-Fh	*Reserved*	X
23-20	PINMUX15_23_20		**VP_DIN[12]/UHPI_HD[4]/UPP_D[4]/PRU0_R30[12]/PRU0_R31[12] Control**	
		0	Selects Function PRU0_R31[12]	I
		1h	Selects Function VP_DIN[12]	I
		2h	Selects Function UHPI_HD[4]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_D[4]	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function PRU0_R30[12]	O
		9h-Fh	*Reserved*	X

(1) I = Input, O = Output, I/O = Bidirectional, X = Undefined

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-37. Pin Multiplexing Control 15 Register (PINMUX15) Field Descriptions (continued)

Bit	Field	Value	Description	Type[1]
19-16	PINMUX15_19_16		**VP_DIN[13]_FIELD/UHPI_HD[5]/UPP_D[5]/PRU0_R30[13]/PRU0_R31[13] Control**	
		0	Selects Function PRU0_R31[13]	I
		1h	Selects Function VP_DIN[13]_FIELD	I
		2h	Selects Function UHPI_HD[5]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_D[5]	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function PRU0_R30[13]	O
		9h-Fh	*Reserved*	X
15-12	PINMUX15_15_12		**VP_DIN[14]_HSYNC/UHPI_HD[6]/UPP_D[6]/PRU0_R30[14]/PRU0_R31[14] Control**	
		0	Selects Function PRU0_R31[14]	I
		1h	Selects Function VP_DIN[14]_HSYNC	I
		2h	Selects Function UHPI_HD[6]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_D[6]	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function PRU0_R30[14]	O
		9h-Fh	*Reserved*	X
11-8	PINMUX15_11_8		**VP_DIN[15]_VSYNC/UHPI_HD[7]/UPP_D[7]/PRU0_R30[15]/PRU0_R31[15] Control**	
		0	Selects Function PRU0_R31[15]	I
		1h	Selects Function VP_DIN[15]_VSYNC	I
		2h	Selects Function UHPI_HD[7]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_D[7]	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function PRU0_R30[15]	O
		9h-Fh	*Reserved*	X
7-4	PINMUX15_7_4		**VP_DIN[0]/UHPI_HD[8]/UPP_D[8]/PRU1_R31[29] Control**	
		0	Selects Function PRU1_R31[29]	I
		1h	Selects Function VP_DIN[0]	I
		2h	Selects Function UHPI_HD[8]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_D[8]	I/O
		5h-Fh	*Reserved*	X
3-0	PINMUX15_3_0		**VP_DIN[1]/UHPI_HD[9]/UPP_D[9]/PRU0_R31[23] Control**	
		0	Selects Function PRU0_R31[23].	I
		1h	Selects Function VP_DIN[1]	I
		2h	Selects Function UHPI_HD[9]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_D[9]	I/O
		5h-Fh	*Reserved*	X

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.10.17 Pin Multiplexing Control 16 Register (PINMUX16)

Figure 10-34. Pin Multiplexing Control 16 Register (PINMUX16)

31 28	27 24	23 20	19 16
PINMUX16_31_28	PINMUX16_27_24	PINMUX16_23_20	PINMUX16_19_16
R/W-0	R/W-0	R/W-0	R/W-0
15 12	**11 8**	**7 4**	**3 0**
PINMUX16_15_12	PINMUX16_11_8	PINMUX16_7_4	PINMUX16_3_0
R/W-0	R/W-0	R/W-0	R/W-0

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

Table 10-38. Pin Multiplexing Control 16 Register (PINMUX16) Field Descriptions

Bit	Field	Value	Description	Type[(1)]
31-28	PINMUX16_31_28		**VP_DOUT[2]/LCD_D[2]/UPP_XD[10]/GP7[10]/PRU1_R31[10] Control**	
		0	Selects Function PRU1_R31[10]	I
		1h	Selects Function VP_DOUT[2]	O
		2h	Selects Function LCD_D[2]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_XD[10]	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP7[10]	I/O
		9h-Fh	*Reserved*	X
27-24	PINMUX16_27_24		**VP_DOUT[3]/LCD_D[3]/UPP_XD[11]/GP7[11]/PRU1_R31[11] Control**	
		0	Selects Function PRU1_R31[11]	I
		1h	Selects Function VP_DOUT[3]	O
		2h	Selects Function LCD_D[3]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_XD[11]	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP7[11]	I/O
		9h-Fh	*Reserved*	X
23-20	PINMUX16_23_20		**VP_DOUT[4]/LCD_D[4]/UPP_XD[12]/GP7[12]/PRU1_R31[12] Control**	
		0	Selects Function PRU1_R31[12]	I
		1h	Selects Function VP_DOUT[4]	O
		2h	Selects Function LCD_D[4]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_XD[12]	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP7[12]	I/O
		9h-Fh	*Reserved*	X

(1) I = Input, O = Output, I/O = Bidirectional, X = Undefined

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-38. Pin Multiplexing Control 16 Register (PINMUX16) Field Descriptions (continued)

Bit	Field	Value	Description	Type[1]
19-16	PINMUX16_19_16		**VP_DOUT[5]/LCD_D[5]/UPP_XD[13]/GP7[13]/PRU1_R31[13] Control**	
		0	Selects Function PRU1_R31[13]	I
		1h	Selects Function VP_DOUT[5]	O
		2h	Selects Function LCD_D[5]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_XD[13]	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP7[13]	I/O
		9h-Fh	*Reserved*	X
15-12	PINMUX16_15_12		**VP_DOUT[6]/LCD_D[6]/UPP_XD[14]/GP7[14]/PRU1_R31[14] Control**	
		0	Selects Function PRU1_R31[14]	I
		1h	Selects Function VP_DOUT[6]	O
		2h	Selects Function LCD_D[6]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_XD[14]	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP7[14]	I/O
		9h-Fh	*Reserved*	X
11-8	PINMUX16_11_8		**VP_DOUT[7]/LCD_D[7]/UPP_XD[15]/GP7[15]/PRU1_R31[15] Control**	
		0	Selects Function PRU1_R31[15]	I
		1h	Selects Function VP_DOUT[7]	O
		2h	Selects Function LCD_D[7]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_XD[15]	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP7[15]	I/O
		9h-Fh	*Reserved*	X
7-4	PINMUX16_7_4		**VP_DIN[8]/UHPI_HD[0]/UPP_D0/GP6[5]/PRU1_R31[0] Control**	
		0	Selects Function PRU1_R31[0]	I
		1h	Selects Function VP_DIN[8]	I
		2h	Selects Function UHPI_HD[0]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_D[0]	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP6[5]	I/O
		9h-Fh	*Reserved*	X
3-0	PINMUX16_3_0		**VP_DIN[9]/UHPI_HD[1]/UPP_D[1]/PRU0_R30[9]/PRU0_R31[9] Control**	
		0	Selects Function PRU0_R31[9]	I
		1h	Selects Function VP_DIN[9]	I
		2h	Selects Function UHPI_HD[1]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_D[1]	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function PRU0_R30[9]	O
		9h-Fh	*Reserved*	X

10.5.10.18 Pin Multiplexing Control 17 Register (PINMUX17)

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 10-35. Pin Multiplexing Control 17 Register (PINMUX17)

31 28	27 24	23 20	19 16
PINMUX17_31_28	PINMUX17_27_24	PINMUX17_23_20	PINMUX17_19_16
R/W-0	R/W-0	R/W-0	R/W-0

15 12	11 8	7 4	3 0
PINMUX17_15_12	PINMUX17_11_8	PINMUX17_7_4	PINMUX17_3_0
R/W-0	R/W-0	R/W-0	R/W-0

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

Table 10-39. Pin Multiplexing Control 17 Register (PINMUX17) Field Descriptions

Bit	Field	Value	Description	Type[1]
31-28	PINMUX17_31_28		**VP_DOUT[10]/LCD_D[10]/UPP_XD[2]/GP7[2]/BOOT[2] Control**	
		0	Selects Function BOOT[2]	I
		1h	Selects Function VP_DOUT[10]	O
		2h	Selects Function LCD_D[10]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_XD[2]	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP7[2]	I/O
		9h-Fh	*Reserved*	X
27-24	PINMUX17_27_24		**VP_DOUT[11]/LCD_D[11]/UPP_XD[3]/GP7[3]/BOOT[3] Control**	
		0	Selects Function BOOT[3]	I
		1h	Selects Function VP_DOUT[11]	O
		2h	Selects Function LCD_D[11]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_XD[3]	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP7[3]	I/O
		9h-Fh	*Reserved*	X
23-20	PINMUX17_23_20		**VP_DOUT[12]/LCD_D[12]/UPP_XD[4]/GP7[4]/BOOT[4] Control**	
		0	Selects Function BOOT[4]	I
		1h	Selects Function VP_DOUT[12]	O
		2h	Selects Function LCD_D[12]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_XD[4]	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP7[4]	I/O
		9h-Fh	*Reserved*	X

(1) I = Input, O = Output, I/O = Bidirectional, X = Undefined

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-39. Pin Multiplexing Control 17 Register (PINMUX17) Field Descriptions (continued)

Bit	Field	Value	Description	Type[1]
19-16	PINMUX17_19_16		**VP_DOUT[13]/LCD_D[13]/UPP_XD[5]/GP7[5]/BOOT[5] Control**	
		0	Selects Function BOOT[5]	I
		1h	Selects Function VP_DOUT[13]	O
		2h	Selects Function LCD_D[13]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_XD[5]	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP7[5]	I/O
		9h-Fh	*Reserved*	X
15-12	PINMUX17_15_12		**VP_DOUT[14]/LCD_D[14]/UPP_XD[6]/GP7[6]/BOOT[6] Control**	
		0	Selects Function BOOT[6]	I
		1h	Selects Function VP_DOUT[14]	O
		2h	Selects Function LCD_D[14]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_XD[6]	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP7[6]	I/O
		9h-Fh	*Reserved*	X
11-8	PINMUX17_11_8		**VP_DOUT[15]/LCD_D[15]/UPP_XD[7]/GP7[7]/BOOT[7] Control**	
		0	Selects Function BOOT[7]	I
		1h	Selects Function VP_DOUT[15]	O
		2h	Selects Function LCD_D[15]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_XD[7]	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP7[7]	I/O
		9h-Fh	*Reserved*	X
7-4	PINMUX17_7_4		**VP_DOUT[0]/LCD_D[0]/UPP_XD[8]/GP7[8]/PRU1_R31[8] Control**	
		0	Selects Function PRU1_R31[8]	I
		1h	Selects Function VP_DOUT[0]	O
		2h	Selects Function LCD_D[0]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_XD[8]	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP7[8]	I/O
		9h-Fh	*Reserved*	X
3-0	PINMUX17_3_0		**VP_DOUT[1]/LCD_D[1]/UPP_XD[9]/GP7[9]/PRU1_R31[9] Control**	
		0	Selects Function PRU1_R31[9]	I
		1h	Selects Function VP_DOUT[1]	O
		2h	Selects Function LCD_D[1]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_XD[9]	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP7[9]	I/O
		9h-Fh	*Reserved*	X

10.5.10.19 Pin Multiplexing Control 18 Register (PINMUX18)

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 10-36. Pin Multiplexing Control 18 Register (PINMUX18)

31 28	27 24	23 20	19 16
PINMUX18_31_28	PINMUX18_27_24	PINMUX18_23_20	PINMUX18_19_16
R/W-0	R/W-0	R/W-0	R/W-0
15 12	**11 8**	**7 4**	**3 0**
PINMUX18_15_12	PINMUX18_11_8	PINMUX18_7_4	PINMUX18_3_0
R/W-0	R/W-0	R/W-0	R/W-0

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

Table 10-40. Pin Multiplexing Control 18 Register (PINMUX18) Field Descriptions

Bit	Field	Value	Description	Type[1]
31-28	PINMUX18_31_28		**MMCSD1_DAT[6]/LCD_MCLK/PRU1_R30[6]/GP8[10]/PRU1_R31[7] Control**	
		0	Selects Function PRU1_R31[7]	I
		1h	Selects Function MMCSD1_DAT[6]	I/O
		2h	Selects Function LCD_MCLK	O
		3h	*Reserved*	X
		4h	Selects Function PRU1_R30[6]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP8[10]	I/O
		9h-Fh	*Reserved*	X
27-24	PINMUX18_27_24		**MMCSD1_DAT[7]/LCD_PCLK/PRU1_R30[7]/GP8[11] Control**	
		0	Pin is 3-stated.	Z
		1h	Selects Function MMCSD1_DAT[7]	I/O
		2h	Selects Function LCD_PCLK	O
		3h	*Reserved*	X
		4h	Selects Function PRU1_R30[7]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP8[11]	I/O
		9h-Fh	*Reserved*	X
23-20	PINMUX18_23_20		**PRU0_R30[22]/PRU1_R30[8]/UPP_CHB_WAIT/GP8[12]/PRU1_R31[24] Control**	
		0	Selects Function PRU1_R31[24]	I
		1h	Selects Function PRU0_R30[22]	O
		2h	Selects Function PRU1_R30[8]	O
		3h	*Reserved*	X
		4h	Selects Function UPP_CHB_WAIT	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP8[12]	I/O
		9h-Fh	*Reserved*	X

(1) I = Input, O = Output, I/O = Bidirectional, X = Undefined, Z = High-impedance state

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-40. Pin Multiplexing Control 18 Register (PINMUX18) Field Descriptions (continued)

Bit	Field	Value	Description	Type[1]
19-16	PINMUX18_19_16		**PRU0_R30[23]/MMCSD1_CMD/UPP_CHB_ENABLE/GP8[13]/PRU1_R31[25] Control**	
		0	Selects Function PRU1_R31[25]	I
		1h	Selects Function PRU0_R30[23]	O
		2h	Selects Function MMCSD1_CMD	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_CHB_ENABLE	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP8[13]	I/O
		9h-Fh	*Reserved*	X
15-12	PINMUX18_15_12		**PRU0_R30[24]/MMCSD1_CLK/UPP_CHB_START/GP8[14]/PRU1_R31[26] Control**	
		0	Selects Function PRU1_R31[26]	I
		1h	Selects Function PRU0_R30[24]	O
		2h	Selects Function MMCSD1_CLK	O
		3h	*Reserved*	X
		4h	Selects Function UPP_CHB_START	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP8[14]	I/O
		9h-Fh	*Reserved*	X
11-8	PINMUX18_11_8		**PRU0_R30[25]/MMCSD1_DAT[0]/UPP_CHB_CLOCK/GP8[15]/PRU1_R31[27] Control**	
		0	Selects Function PRU1_R31[27]	I
		1h	Selects Function PRU0_R30[25]	O
		2h	Selects Function MMCSD1_DAT[0]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_CHB_CLOCK	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP8[15]	I/O
		9h-Fh	*Reserved*	X
7-4	PINMUX18_7_4		**VP_DOUT[8]/LCD_D[8]/UPP_XD[0]/GP7[0]/BOOT[0] Control**	
		0	Selects Function BOOT[0]	I
		1h	Selects Function VP_DOUT[8]	O
		2h	Selects Function LCD_D[8]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_XD[0]	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP7[0]	I/O
		9h-Fh	*Reserved*	X
3-0	PINMUX18_3_0		**VP_DOUT[9]/LCD_D[9]/UPP_XD[1]/GP7[1]/BOOT[1] Control**	
		0	Selects Function BOOT[1]	I
		1h	Selects Function VP_DOUT[9]	O
		2h	Selects Function LCD_D[9]	I/O
		3h	*Reserved*	X
		4h	Selects Function UPP_XD[1]	I/O
		5h-7h	*Reserved*	X
		8h	Selects Function GP7[1]	I/O
		9h-Fh	*Reserved*	X

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.10.20 Pin Multiplexing Control 19 Register (PINMUX19)

Figure 10-37. Pin Multiplexing Control 19 Register (PINMUX19)

31 28	27 24	23 20	19 16
PINMUX19_31_28	PINMUX19_27_24	PINMUX19_23_20	PINMUX19_19_16
R/W-0	R/W-0	R/W-0	R/W-0
15 12	**11 8**	**7 4**	**3 0**
PINMUX19_15_12	PINMUX19_11_8	PINMUX19_7_4	PINMUX19_3_0
R/W-0	R/W-0	R/W-0	R/W-0

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

Table 10-41. Pin Multiplexing Control 19 Register (PINMUX19) Field Descriptions

Bit	Field	Value	Description	Type[1]
31-28	PINMUX19_31_28		**RTCK/GP8[0] Control**	
		0	Selects Function RTCK	O
		1h	Selects Function RTCK	O
		2h-7h	*Reserved*	X
		8h	Selects Function GP8[0]	I/O
		9h-Fh	*Reserved*	X
27-24	PINMUX19_27_24		**$\overline{\text{LCD_AC_ENB_CS}}$/GP6[0]/PRU1_R31[28] Control**	
		0	Selects Function PRU1_R31[28]	I
		1h	*Reserved*	X
		2h	Selects Function $\overline{\text{LCD_AC_ENB_CS}}$	O
		3h-7h	*Reserved*	X
		8h	Selects Function GP6[0]	I/O
		9h-Fh	*Reserved*	X
23-20	PINMUX19_23_20		**VP_CLKOUT3/PRU1_R30[0]/GP6[1]/PRU1_R31[1] Control**	
		0	Selects Function PRU1_R31[1]	I
		1h	Selects Function VP_CLKOUT3	O
		2h-3h	*Reserved*	X
		4h	Selects Function PRU1_R30[0]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP6[1]	I/O
		9h-Fh	*Reserved*	X
19-16	PINMUX19_19_16		**VP_CLKIN3/MMCSD1_DAT[1]/PRU1_R30[1]/GP6[2]/PRU1_R31[2] Control**	
		0	Selects Function PRU1_R31[2]	I
		1h	Selects Function VP_CLKIN3	I
		2h	Selects Function MMCSD1_DAT[1]	I/O
		3h	*Reserved*	X
		4h	Selects Function PRU1_R30[1]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP6[2]	I/O
		9h-Fh	*Reserved*	X

(1) I = Input, O = Output, I/O = Bidirectional, X = Undefined

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-41. Pin Multiplexing Control 19 Register (PINMUX19) Field Descriptions (continued)

Bit	Field	Value	Description	Type[1]
15-12	PINMUX19_15_12		**VP_CLKOUT2/MMCSD1_DAT[2]/PRU1_R30[2]/GP6[3]/PRU1_R31[3] Control**	
		0	Selects Function PRU1_R31[3]	I
		1h	Selects Function VP_CLKOUT2	O
		2h	Selects Function MMCSD1_DAT[2]	I/O
		3h	*Reserved*	X
		4h	Selects Function PRU1_R30[2]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP6[3]	I/O
		9h-Fh	*Reserved*	X
11-8	PINMUX19_11_8		**VP_CLKIN2/MMCSD1_DAT[3]/PRU1_R30[3]/GP6[4]/PRU1_R31[4] Control**	
		0	Selects Function PRU1_R31[4]	I
		1h	Selects Function VP_CLKIN2	I
		2h	Selects Function MMCSD1_DAT[3]	I/O
		3h	*Reserved*	X
		4h	Selects Function PRU1_R30[3]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP6[4]	I/O
		9h-Fh	*Reserved*	X
7-4	PINMUX19_7_4		**MMCSD1_DAT[4]/LCD_VSYNC/PRU1_R30[4]/GP8[8]/PRU1_R31[5] Control**	
		0	Selects Function PRU1_R31[5]	I
		1h	Selects Function MMCSD1_DAT[4]	I/O
		2h	Selects Function LCD_VSYNC	O
		3h	*Reserved*	X
		4h	Selects Function PRU1_R30[4]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP8[8]	I/O
		9h-Fh	*Reserved*	X
3-0	PINMUX19_3_0		**MMCSD1_DAT[5]/LCD_HSYNC/PRU1_R30[5]/GP8[9]/PRU1_R31[6] Control**	
		0	Selects Function PRU1_R31[6]	I
		1h	Selects Function MMCSD1_DAT[5]	I/O
		2h	Selects Function LCD_HSYNC	O
		3h	*Reserved*	X
		4h	Selects Function PRU1_R30[5]	O
		5h-7h	*Reserved*	X
		8h	Selects Function GP8[9]	I/O
		9h-Fh	*Reserved*	X

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.11 Suspend Source Register (SUSPSRC)

The suspend source register (SUSPSRC) indicates the emulation suspend source for those peripherals that support emulation suspend. A value of 0 for a SUSPSRC bit corresponding to the peripheral, indicates that the ARM emulator controls the peripheral's emulation suspend signal.

The SUSPSRC is shown in Figure 10-38 and described in Table 10-42.

Figure 10-38. Suspend Source Register (SUSPSRC)

31	30	29	28	27	26	25	24
Reserved	Reserved	TIMER64P_2SRC	TIMER64P_1SRC	TIMER64P_0SRC	Reserved	Reserved	EPWM1SRC
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
23	22	21	20	19	18	17	16
EPWM0SRC	SPI1SRC	SPI0SRC	UART2SRC	UART1SRC	UART0SRC	I2C1SRC	I2C0SRC
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
15	14	13	12	11	10	9	8
Reserved	VPIFSRC	Reserved	HPISRC	Reserved	Reserved	USB0SRC	MCBSP1SRC
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
7	6	5	4	3	2	1	0
MCBSP0SRC	PRUSRC	Reserved	UPPSRC	TIMER64P_3SRC	ECAP2SRC	ECAP1SRC	ECAP0SRC
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; -*n* = value after reset

Table 10-42. Suspend Source Register (SUSPSRC) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	1	Reserved. Write the default value to all bits when modifying this register.
29	TIMER64P_2SRC		**Timer2 64 Emulation Suspend Source.**
		0	ARM is the source of the emulation suspend.
		1	No emulation suspend.
28	TIMER64P_1SRC		**Timer1 64 Emulation Suspend Source.**
		0	ARM is the source of the emulation suspend.
		1	No emulation suspend.
27	TIMER64P_0SRC		**Timer0 64 Emulation Suspend Source.**
		0	ARM is the source of the emulation suspend.
		1	No emulation suspend.
26-25	Reserved	1	Reserved. Write the default value to all bits when modifying this register.
24	EPWM1SRC		**EPWM1 Emulation Suspend Source.**
		0	ARM is the source of the emulation suspend.
		1	No emulation suspend.
23	EPWM0SRC		**EPWM0 Emulation Suspend Source.**
		0	ARM is the source of the emulation suspend.
		1	No emulation suspend.
22	SPI1SRC		**SPI1 Emulation Suspend Source.**
		0	ARM is the source of the emulation suspend.
		1	No emulation suspend.
21	SPI0SRC		**SPI0 Emulation Suspend Source.**
		0	ARM is the source of the emulation suspend.
		1	No emulation suspend.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-42. Suspend Source Register (SUSPSRC) Field Descriptions (continued)

Bit	Field	Value	Description
20	UART2SRC		**UART2 Emulation Suspend Source.**
		0	ARM is the source of the emulation suspend.
		1	No emulation suspend.
19	UART1SRC		**UART1 Emulation Suspend Source.**
		0	ARM is the source of the emulation suspend.
		1	No emulation suspend.
18	UART0SRC		**UART0 Emulation Suspend Source.**
		0	ARM is the source of the emulation suspend.
		1	No emulation suspend.
17	I2C1SRC		**I2C1 Emulation Suspend Source.**
		0	ARM is the source of the emulation suspend.
		1	No emulation suspend.
16	I2C0SRC		**I2C0 Emulation Suspend Source.**
		0	ARM is the source of the emulation suspend.
		1	No emulation suspend.
15	Reserved	1	Reserved. Write the default value to all bits when modifying this register.
14	VPIFSRC		**VPIF Emulation Suspend Source.**
		0	ARM is the source of the emulation suspend.
		1	No emulation suspend.
13	Reserved	1	Reserved. Write the default value to all bits when modifying this register.
12	HPISRC		**HPI Emulation Suspend Source.**
		0	ARM is the source of the emulation suspend.
		1	No emulation suspend.
11-10	Reserved	1	Reserved. Write the default value to all bits when modifying this register.
9	USB0SRC		**USB0 (USB 2.0) Emulation Suspend Source.**
		0	ARM is the source of the emulation suspend.
		1	No emulation suspend.
8	MCBSP1SRC		**McBSP1 Emulation Suspend Source.**
		0	ARM is the source of the emulation suspend.
		1	No emulation suspend.
7	MCBSP0SRC		**McBSP0 Emulation Suspend Source.**
		0	ARM is the source of the emulation suspend.
		1	No emulation suspend.
6	PRUSRC		**PRU Emulation Suspend Source.**
		0	ARM is the source of the emulation suspend.
		1	No emulation suspend.
5	Reserved	1	Reserved. Write the default value to all bits when modifying this register.
4	UPPSRC		**uPP Emulation Suspend Source.**
		0	ARM is the source of the emulation suspend.
		1	No emulation suspend.
3	TIMER64P_3SRC		**Timer3 64 Emulation Suspend Source.**
		0	ARM is the source of the emulation suspend.
		1	No emulation suspend.
2	ECAP2SRC		**ECAP2 Emulation Suspend Source.**
		0	ARM is the source of the emulation suspend.
		1	No emulation suspend.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-42. Suspend Source Register (SUSPSRC) Field Descriptions (continued)

Bit	Field	Value	Description
1	ECAP1SRC		**ECAP1 Emulation Suspend Source.**
		0	ARM is the source of the emulation suspend.
		1	No emulation suspend.
0	ECAP0SRC		**ECAP0 Emulation Suspend Source.**
		0	ARM is the source of the emulation suspend.
		1	No emulation suspend.

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.12 Chip Signal Register (CHIPSIG)

Interrupts to the ARM may be generated by setting one of the four CHIPSIG[3-0] bits in the chip signal register (CHIPSIG). Writing a 1 to these bits sets the interrupts, writing a 0 has no effect. Reads return the value of these bits and can also be used as status bits. The CHIPSIG is shown in Figure 10-39 and described in Table 10-43.

Figure 10-39. Chip Signal Register (CHIPSIG)

31	16
Reserved	
R-0	

15	5	4	3	2	1	0
Reserved		Rsvd	CHIPSIG3	CHIPSIG2	CHIPSIG1	CHIPSIG0
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 10-43. Chip Signal Register (CHIPSIG) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4	Reserved	0	Reserved. Write the default value when modifying this register.
3	CHIPSIG3		**Asserts SYSCFG_CHIPINT3 interrupt.**
		0	No effect
		1	Asserts interrupt
2	CHIPSIG2		**Asserts SYSCFG_CHIPINT2 interrupt.**
		0	No effect
		1	Asserts interrupt
1	CHIPSIG1		**Asserts SYSCFG_CHIPINT1 interrupt.**
		0	No effect
		1	Asserts interrupt
0	CHIPSIG0		**Asserts SYSCFG_CHIPINT0 interrupt.**
		0	No effect
		1	Asserts interrupt

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.13 Chip Signal Clear Register (CHIPSIG_CLR)

The chip signal clear register (CHIPSIG_CLR) is used to clear the bits set in the chip signal register (CHIPSIG). Writing a 1 to a CHIPSIG[*n*] bit in CHIPSIG_CLR clears the corresponding CHIPSIG[*n*] bit in CHIPSIG; writing a 0 has no effect. After servicing the interrupt, the interrupted processor can clear the bits set in CHIPSIG by writing 1 to the corresponding bits in CHIPSIG_CLR. The other processor may poll the CHIPSIG[*n*] bit to determine when the interrupted processor has completed the interrupt service. The CHIPSIG_CLR is shown in Figure 10-40 and described in Table 10-44.

For more information on ARM interrupts, see the *ARM Interrupt Controller (AINTC)* chapter.

Figure 10-40. Chip Signal Clear Register (CHIPSIG_CLR)

31	16
Reserved	
R-0	

15	5	4	3	2	1	0
Reserved		Rsvd	CHIPSIG3	CHIPSIG2	CHIPSIG1	CHIPSIG0
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 10-44. Chip Signal Clear Register (CHIPSIG_CLR) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4	Reserved	0	Reserved. Write the default value when modifying this register.
3	CHIPSIG3		**Clears SYSCFG_CHIPINT3 interrupt.**
		0	No effect
		1	Clears interrupt
2	CHIPSIG2		**Clears SYSCFG_CHIPINT2 interrupt.**
		0	No effect
		1	Clears interrupt
1	CHIPSIG1		**Clears SYSCFG_CHIPINT1 interrupt.**
		0	No effect
		1	Clears interrupt
0	CHIPSIG0		**Clears SYSCFG_CHIPINT0 interrupt.**
		0	No effect
		1	Clears interrupt

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.14 Chip Configuration 0 Register (CFGCHIP0)

The chip configuration 0 register (CFGCHIP0) controls the following functions:

- PLL Controller 0 memory-mapped register lock: Used to lock out writes to the PLLC0 memory-mapped registers (MMRs) to prevent any erroneous writes in software to the PLLC0 register space.
- EDMA3_0 Transfer Controller Default Burst Size (DBS) Control: This controls the maximum number of bytes issued per read/write command or the burst size for the individual transfer controllers (TCs) on the device. By default for all transfer controllers, the burst size is set to 16 bytes. However, CFGCHIP0 allows configurability of this parameter so that the TC can have a burst size of 16, 32, or 64 bytes. The burst size determines the intra packet efficiency for the EDMA3_0 transfers. Additionally, it also facilitates preemption at a system level, as all transfer requests are internally broken down by the transfer controller up to DBS size byte chunks and on a system level, each master's priority (configured by the MSTPRI register) is evaluated at burst size boundaries. The DBS value can significantly impact the standalone throughput performance depending on the source and destination (bus width/frequency/burst support etc) and the TC FIFO size, etc. Therefore, the DBS size configuration should be carefully analyzed to meet the system's throughput/performance requirements.

The CFGCHIP0 is shown in Figure 10-41 and described in Table 10-45.

Figure 10-41. Chip Configuration 0 Register (CFGCHIP0)

31	16
Reserved	
R-0	

15	5	4	3	2	1	0
Reserved		PLL_MASTER_LOCK	EDMA30TC1DBS		EDMA30TC0DBS	
R-0		R/W-0	R/W-0		R/W-0	

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

Table 10-45. Chip Configuration 0 Register (CFGCHIP0) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved.
4	PLL_MASTER_LOCK		**PLLC0 MMRs lock.**
		0	PLLC0 MMRs are freely accessible.
		1	All PLLC0 MMRs are locked.
3-2	EDMA30TC1DBS		**EDMA3_0_TC1 Default Burst Size (DBS).**
		0	16 bytes
		1h	32 bytes
		2h	64 bytes
		3h	*Reserved*
1-0	EDMA30TC0DBS		**EDMA3_0_TC0 Default Burst Size (DBS).**
		0	16 bytes
		1h	32 bytes
		2h	64 bytes
		3h	*Reserved*

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.15 Chip Configuration 1 Register (CFGCHIP1)

The chip configuration 1 register (CFGCHIP1) controls the following functions:

- eCAP0/1/2 event input source: Allows using McASP0 TX/RX events as eCAP event input sources.
- HPI Control: Allows HPIEN bit control that determines whether or not the HPI module has control over the HPI pins (multiplexed with other peripheral pins). It also provides configurability to select whether the host address is a word address or a byte address mode.
- EDMA3_1 Transfer Controller Default Burst Size (DBS) Control: This controls the maximum number of bytes issued per read/write command or the burst size for the individual transfer controllers (TCs) on the device. By default for all transfer controllers, the burst size is set to 16 bytes. However, CFGCHIP1 allows configurability of this parameter so that the TC can have a burst size of 16, 32, or 64 bytes. The burst size determines the intra packet efficiency for the EDMA3_1 transfers. Additionally, it also facilitates preemption at a system level, as all transfer requests are internally broken down by the transfer controller up to DBS size byte chunks and on a system level, each master's priority (configured by the MSTPRI register) is evaluated at burst size boundaries. The DBS value can significantly impact the standalone throughput performance depending on the source and destination (bus width/frequency/burst support etc) and the TC FIFO size, etc. Therefore, the DBS size configuration should be carefully analyzed to meet the system's throughput/performance requirements.
- eHRPWM Time Base Clock (TBCLK) Synchronization: Allows the software to globally synchronize all enabled eHRPWM modules to the time base clock (TBCLK).
- McASP0 AMUTEIN signal source control: Allows selecting GPIO interrupt from different banks as source for the McASP0 AMUTEIN signal.

The CFGCHIP1 is shown in Figure 10-42 and described in Table 10-46.

Figure 10-42. Chip Configuration 1 Register (CFGCHIP1)

31	27	26	22	21	17	16
CAP2SRC		CAP1SRC		CAP0SRC		HPIBYTEAD
R/W-0		R/W-0		R/W-0		R/W-0

15	14	13	12	11	8
HPIENA	EDMA31TC0DBS		TBCLKSYNC	Reserved	
R/W-0	R/W-0		R/W-0	R/W-0	

7	4	3	0
Reserved		AMUTESEL0	
R/W-0		R/W-0	

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

Table 10-46. Chip Configuration 1 Register (CFGCHIP1) Field Descriptions

Bit	Field	Value	Description
31-27	CAP2SRC		**Selects the eCAP2 module event input.**
		0	eCAP2 Pin input
		1h	McASP0 TX DMA Event
		2h	McASP0 RX DMA Event
		3h-1Fh	*Reserved*
26-22	CAP1SRC		**Selects the eCAP1 module event input.**
		0	eCAP1 Pin input
		1h	McASP0 TX DMA Event
		2h	McASP0 RX DMA Event
		3h-1Fh	*Reserved*

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-46. Chip Configuration 1 Register (CFGCHIP1) Field Descriptions (continued)

Bit	Field	Value	Description
21-17	CAP0SRC		**Selects the eCAP0 module event input.**
		0	eCAP0 Pin input
		1h	McASP0 TX DMA Event
		2h	McASP0 RX DMA Event
		3h-1Fh	*Reserved*
16	HPIBYTEAD		**HPI Byte/Word Address Mode select.**
		0	Host address is a word address.
		1	Host address is a byte address.
15	HPIENA		**HPI Enable Bit.**
		0	HPI is disabled.
		1	HPI is enabled.
14-13	EDMA31TC0DBS		**EDMA3_1_TC0 Default Burst Size.**
		0	16 bytes
		1h	32 bytes
		2h	64 bytes
		3h	*Reserved*
12	TBCLKSYNC		**eHRPWM Module Time Base Clock Synchronization.** Allows you to globally synchronize all enabled eHRPWM modules to the time base clock (TBCLK).
		0	Time base clock (TBCLK) within each enabled eHRPWM module is stopped.
		1	All enabled eHRPWM module clocks are started with the first rising edge of TBCLK aligned. For perfectly synchronized TBCLKs, the prescaler bits in the TBCTL register of each eHRPWM module must be set identically.
11-4	Reserved	0	Reserved. Write the default value to all bits when modifying this register.
3-0	AMUTESEL0		**Selects the source of McASP0 AMUTEIN signal.**
		0	Drive McASP0 AMUTEIN signal low.
		1h	GPIO Interrupt from Bank 0
		2h	GPIO Interrupt from Bank 1
		3h	GPIO Interrupt from Bank 2
		4h	GPIO Interrupt from Bank 3
		5h	GPIO Interrupt from Bank 4
		6h	GPIO Interrupt from Bank 5
		7h	GPIO Interrupt from Bank 6
		8h	GPIO Interrupt from Bank 7
		9h-Fh	*Reserved*

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.16 Chip Configuration 2 Register (CFGCHIP2)

The chip configuration 2 register (CFGCHIP2) controls the following functions:

- USB2.0 OTG PHY

The CFGCHIP2 is shown in Figure 10-43 and described in Table 10-47.

Figure 10-43. Chip Configuration 2 Register (CFGCHIP2)

31							24
Reserved							
R-0							

23						18	17	16
Reserved							USB0PHYCLKGD	USB0VBUSSENSE
R-0							R-0	R-0

15	14	13	12	11	10	9	8
RESET	USB0OTGMODE		Reserved	USB0PHYCLKMUX	USB0PHYPWDN	USB0OTGPWRDN	USB0DATPOL
R/W-1	R/W-3h		R/W-0	R/W-1	R/W-1	R/W-1	R/W-1

7	6	5	4	3			0
Reserved	USB0PHY_PLLON	USB0SESNDEN	USB0VBDTCTEN	USB0REF_FREQ			
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 10-47. Chip Configuration 2 Register (CFGCHIP2) Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17	USB0PHYCLKGD		**Status of USB2.0 PHY.**
		0	Clock is not present, power is not good, and PLL has not locked.
		1	Clock is present, power is good, and PLL has locked.
16	USB0VBUSSENSE		**Status of USB2.0 PHY VBUS sense.**
		0	PHY is not sensing voltage presence on the VBUS pin.
		1	PHY is sensing voltage presence on the VBUS pin.
15	RESET		**USB2.0 PHY reset.**
		0	Not in reset.
		1	USB2.0 PHY in reset.
14-13	USB0OTGMODE		**USB2.0 OTG subsystem mode.**
		0	No override. PHY drive signals to controller based on its comparators for VBUS and ID pins.
		1h	Override phy values to force USB host operation.
		2h	Override phy values to force USB device operation.
		3h	Override phy values to force USB host operation with VBUS low.
12	Reserved	0	Reserved. Write the default value when modifying this register.
11	USB0PHYCLKMUX		**USB2.0 PHY reference clock input mux.**
		0	USB2.0 PHY reference clock (USB_REFCLKIN) is sourced by an external pin.
		1	USB2.0 PHY reference clock (AUXCLK) is internally generated from the PLL.
10	USB0PHYPWDN		**USB2.0 PHY operation state control.**
		0	USB2.0 PHY is enabled and is in operating state (normal operation).
		1	USB2.0 PHY is disabled and powered down.
9	USB0OTGPWRDN		**USB2.0 OTG subsystem (SS) operation state control.**
		0	OTG SS is enabled and is in operating state (normal operation).
		1	OTG SS is disabled and is powered down.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-47. Chip Configuration 2 Register (CFGCHIP2) Field Descriptions (continued)

Bit	Field	Value	Description
8	USB0DATPOL		**USB2.0 differential data lines polarity selector.**
		0	Differential data polarities are inverted (USB_DP is connected to D- and USB_DM is connected to D+).
		1	Differential data polarity are not altered (USB_DP is connected to D+ and USB_DM is connected to D-).
7	Reserved	0	Reserved. Write the default value when modifying this register.
6	USB0PHY_PLLON		**Drives USB2.0 PHY, allowing or preventing it from stopping the 48 MHz clock during USB SUSPEND.**
		0	USB2.0 PHY is allowed to stop the 48 MHz clock during USB SUSPEND.
		1	USB2.0 PHY is prevented from stopping the 48 MHz clock during USB SUSPEND
5	USB0SESNDEN		**USB2.0 Session End comparator enable.**
		0	Session End comparator is disabled.
		1	Session End comparator is enabled.
4	USB0VBDTCTEN		**USB2.0 VBUS line comparators enable.**
		0	All VBUS line comparators are disabled.
		1	All VBUS line comparators are enabled.
3-0	USB0REF_FREQ		**USB2.0 PHY reference clock input frequencies.**
		0	*Reserved*
		1h	12 MHz
		2h	24 MHz
		3h	48 MHz
		4h	19.2 MHz
		5h	38.4 MHz
		6h	13 MHz
		7h	26 MHz
		8h	20 MHz
		9h	40 MHz
		Ah-Fh	*Reserved*

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.17 Chip Configuration 3 Register (CFGCHIP3)

The chip configuration 3 register (CFGCHIP3) controls the following peripheral/module functions:

- uPP Clock Source Control: Allows control for the source of the uPP 2x transmit clock.
- PLL Controller 1 memory-mapped register lock: Used to lock out writes to the PLLC1 memory-mapped registers (MMRs) to prevent any erroneous writes in software to the PLLC1 register space.
- ASYNC3 Clock Source Control: Allows control for the source of the ASYNC3 clock.
- PRU Event Input Select.
- DIV4p5 Clock Enable/Disable: The DIV4p5 (/4.5) hardware clock divider is provided to generate 133 MHz from the 600 MHz PLL clock for use as clocks to the EMIFs. Allows enabling/disabling this clock divider.
- EMIFA Module Clock Source Control: Allows control for the source of the EMIFA module clock.

The CFGCHIP3 is shown in Figure 10-44 and described in Table 10-48.

Figure 10-44. Chip Configuration 3 Register (CFGCHIP3)

31						16	15						8
Reserved							Reserved						
R-0							R/W-FFh						
7	6	5	4	3	2	1	0						
Reserved	UPP_TX_CLKSRC	PLL1_MASTER_LOCK	ASYNC3_CLKSRC	PRUEVTSEL	DIV45PENA	EMA_CLKSRC	Reserved						
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 10-48. Chip Configuration 3 Register (CFGCHIP3) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-8	Reserved	FFh	Reserved. Write the default value to all bits when modifying this register.
7	Reserved	0	Reserved. Write the default value when modifying this register.
6	UPP_TX_CLKSRC		**Clock source for uPP 2x transmit clock.**
		0	Clock driven by ASYNC3.
		1	Clock driven by external signal, 2xTXCLK.
5	PLL1_MASTER_LOCK		**PLLC1 MMRs lock.**
		0	PLLC1 MMRs are freely accessible.
		1	All PLLC1 MMRs are locked.
4	ASYNC3_CLKSRC		**Clock source for ASYNC3.**
		0	Clock driven by PLL0_SYSCLK2.
		1	Clock driven by PLL1_SYSCLK2.
3	PRUEVTSEL		**PRU event input select.**
		0	Normal mode
		1	Alternate mode
2	DIV45PENA		**Controls the fixed DIV4.5 divider in the PLL controller.**
		0	Divide by 4.5 is disabled.
		1	Divide by 4.5 is enabled.
1	EMA_CLKSRC		**Clock source for EMIFA clock domain.**
		0	Clock driven by PLL0_SYSCLK3
		1	Clock driven by DIV4.5 PLL output
0	Reserved	0	Reserved. Write the default value when modifying this register.

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.18 Chip Configuration 4 Register (CFGCHIP4)

The chip configuration 4 register (CFGCHIP4) is used for clearing the AMUNTEIN signal for McASP0. Writing a 1 causes a single pulse that clears the latched GPIO interrupt for AMUTEIN of McASP0, if it was previously set; reads always return a value of 0. The CFGCHIP4 is shown in Figure 10-45 and described in Table 10-49.

Figure 10-45. Chip Configuration 4 Register (CFGCHIP4)

31	16
Reserved	
R-0	

15	8	7	1	0
Reserved		Reserved		AMUTECLR0
R/W-FFh		R/W-0		R/W-0

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

Table 10-49. Chip Configuration 4 Register (CFGCHIP4) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-8	Reserved	FFh	Reserved. Write the default value to all bits when modifying this register.
7-1	Reserved	0	Reserved. Write the default value to all bits when modifying this register.
0	AMUTECLR0		**Clears the latched GPIO interrupt for AMUTEIN of McASP0 when set to 1.**
		0	No effect
		1	Clears interrupt

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.19 VTP I/O Control Register (VTPIO_CTL)

The VTP I/O control register (VTPIO_CTL) is used to control the calibration of the DDR2/mDDR memory controller I/Os with respect to voltage, temperature, and process (VTP). The voltage, temperature, and process information is used to control the IO's output impedance. The VTPIO_CTL is shown in Figure 10-46 and described in Table 10-50.

Figure 10-46. VTP I/O Control Register (VTPIO_CTL)

31	30	29	28	27	26	25	24
Reserved							
R-0							

23	22	21	20	19	18	17	16
Reserved					VREFEN	VREFTAP	
R-0					R/W-0	R/W-0	

15	14	13	12	11	10	9	8
READY	IOPWRDN	CLKRZ	Reserved				PWRSAVE
R-0	R/W-0	R/W-0	R/W-0				R/W-0

7	6	5	4	3	2	1	0
LOCK	POWERDN	D			F		
R/W-0	R/W-1	R/W-6h			R/W-7h		

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

Table 10-50. VTP I/O Control Register (VTPIO_CTL) Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reserved
18	VREFEN		**Internal DDR I/O Vref enable.**
		0	Connected to pad, external reference.
		1	*Reserved*
17-16	VREFTAP		**Selection for internal reference voltage level.**
		0	Vref = 50.0% of VDDS
		1h-3h	*Reserved*
15	READY		**VTP Ready status.**
		0	VTP is not ready.
		1	VTP is ready.
14	IOPWRDN		**Power down enable for DDR input buffer.**
		0	Disable power down control by the PWRDNEN bit in the DDR PHY control register 1 (DRPYC1R).
		1	Enable power down control by the PWRDNEN bit in the DDR PHY control register 1 (DRPYC1R).
13	CLKRZ	0	**VTP clear.** Write 0 to clear VTP flops.
12-9	Reserved	0	Reserved. Write the default value to all bits when modifying this register.
8	PWRSAVE		**VTP power save mode.** Turn off power to the external resistor when it is not needed. The PWRSAVE bit setting is only valid when the POWERDN bit is cleared to 0.
		0	Disable power save mode.
		1	Enable power save mode.
7	LOCK		**VTP impedance lock.** Lock impedance value so that the VTP controller can be powered down.
		0	Unlock impedance.
		1	Lock impedance.
6	POWERDN		**VTP power down.** Power down the VTP controller. The PWRSAVE bit setting is only valid when the POWERDN bit is cleared to 0.
		0	Disable power down.
		1	Enable power down.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-50. VTP I/O Control Register (VTPIO_CTL) Field Descriptions (continued)

Bit	Field	Value	Description
5-3	D		**Drive strength control bit.**
		0-5h	*Reserved*
		6h	100% drive strength
		7h	*Reserved*
2-0	F		**Digital filter control bit.**
		0-6h	*Reserved*
		7h	Digital filter is enabled.

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.20 DDR Slew Register (DDR_SLEW)

The DDR slew register (DDR_SLEW) reflects the DDR I/O timing as programmed in the device eFuse. The CMOSEN field configures the DDR I/O cells into an LVCMOS buffer (this makes it mDDR compatible). The DDR_SLEW is shown in Figure 10-47 and described in Table 10-51.

Figure 10-47. DDR Slew Register (DDR_SLEW)

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				ODT_TERMON		ODT_TERMOFF	
R-0				R/W-0		R/W-0	

7	6	5	4	3	2	1	0
Reserved		DDR_PDENA	CMOSEN	DDR_DATASLEW		DDR_CMDSLEW	
R-0		R/W-0	R/W-0	R-0		R-0	

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

Table 10-51. DDR Slew Register (DDR_SLEW) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reserved
11-10	ODT_TERMON		**Controls Thevenin termination mode while I/O is in read or write mode.** Termination is not supported on this device.
		0	No termination
		1h-3h	*Reserved*
9-8	ODT_TERMOFF		**Controls Thevenin termination mode while I/O is not in read or write mode.** Termination is not supported on this device.
		0	No termination
		1h-3h	*Reserved*
7-6	Reserved	0	Reserved
5	DDR_PDENA		**Enables pull downs for mDDR mode (should be disabled for DDR2).**
		0	Pull downs are disabled. Disable pull downs when using DDR2.
		1	Pull downs are enabled. Enable pull downs when using mDDR.
4	CMOSEN		**Selects mDDR LVCMOS RX / SSTL18 differential RX.**
		0	SSTL Receiver. Select SSTL when using DDR2.
		1	LVCMOS Receiver. Select LVCMOS when using mDDR.
3-2	DDR_DATASLEW		**Slew rate mode control status for data macro.** Slew rate control is not supported on this device.
		0	Slew rate control is off.
		1h-3h	*Reserved*
1-0	DDR_CMDSLEW		**Slew rate mode control status for command macro.** Slew rate control is not supported on this device.
		0	Slew rate control is off.
		1h-3h	*Reserved*

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.21 Deep Sleep Register (DEEPSLEEP)

The deep sleep register (DEEPSLEEP) control the Deep Sleep logic. See your device-specific data manual and the *Boot Considerations* chapter for details on boot and configuration settings. The DEEPSLEEP is shown in Figure 10-48 and described in Table 10-52.

Figure 10-48. Deep Sleep Register (DEEPSLEEP)

31	30	29 16
SLEEPENABLE	SLEEPCOMPLETE	Reserved
R/W-0	R-0	R-0

15 0
SLEEPCOUNT
R/W-FFFFh

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

Table 10-52. Deep Sleep Register (DEEPSLEEP) Field Descriptions

Bit	Field	Value	Description
31	SLEEPENABLE		**Deep sleep enable.** The software must clear this bit to 0 when the device is awakened from deep sleep.
		0	Device is in normal operating mode; $\overline{\text{DEEPSLEEP}}$ pin has no effect.
		1	Deep sleep mode is enabled; setting $\overline{\text{DEEPSLEEP}}$ pin low initiates oscillator shut down.
30	SLEEPCOMPLETE		**Deep sleep complete.** Once the deep sleep process starts, the software must poll the SLEEPCOMPLETE bit; when the SLEEPCOMPLETE bit is read as 1, the software should clear the SLEEPENABLE bit and continue operation.
		0	SLEEPCOUNT delay is not complete.
		1	SLEEPCOUNT delay is complete.
29-16	Reserved	0	Reserved
15-0	SLEEPCOUNT	0-FFFFh	**Deep sleep counter.** Number of cycles to count prior to the oscillator being stable. All 16 bits are tied directly to the counter in the Deep Sleep logic.

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.22 Pullup/Pulldown Enable Register (PUPD_ENA)

The pullup/pulldown enable register (PUPD_ENA) enables the pull-up or pull-down functionality for the pin group *n* defined in your device-specific data manual. The PUPD_ENA is shown in Figure 10-49 and described in Table 10-53.

Figure 10-49. Pullup/Pulldown Enable Register (PUPD_ENA)

31	0
PUPDENA[*n*]	
R/W-FFFF FFFFh	

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

Table 10-53. Pullup/Pulldown Enable Register (PUPD_ENA) Field Descriptions

Bit	Field	Value	Description
31-0	PUPDENA[*n*]		**Enables internal pull-up or pull-down functionality for pin group CP[n].** See your device-specific data manual for pin group information. The internal pull-up or pull-down functionality selection for bit position *n* in PUPD_ENA is set in the same bit position *n* of the pullup/pulldown select register (PUPD_SEL).
		0	Internal pull-up or pull-down functionality for pin group *n* is disabled.
		1	Internal pull-up or pull-down functionality for pin group *n* is enabled.

10.5.23 Pullup/Pulldown Select Register (PUPD_SEL)

The pullup/pulldown select register (PUPD_SEL) selects between the pull-up or pull-down functionality for the pin group *n* defined in your device-specific data manual. The PUPD_SEL is shown in Figure 10-50 and described in Table 10-54 and Table 10-55.

NOTE: The PUPD_SEL settings are not active until the device is out of reset. During reset, all of the CP[*n*] pins are pulled down. If the application requires a pull-up during reset, an external pull-up should be used.

Figure 10-50. Pullup/Pulldown Select Register (PUPD_SEL)

31	0
PUPDSEL[*n*]	
R/W-C3FF FFFFh	

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

Table 10-54. Pullup/Pulldown Select Register (PUPD_SEL) Field Descriptions

Bit	Field	Value	Description
31-0	PUPDSEL[*n*]		**Selects between the internal pull-up or pull-down functionality for pin group CP[n].** See your device-specific data manual for pin group information. The selection for bit position *n* in PUPD_SEL is only valid when the same bit position *n* is set in the pullup/pulldown enable register (PUPD_ENA).
		0	Internal pull-down functionality for pin group *n* is disabled.
		1	Internal pull-up functionality for pin group *n* is enabled.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 10-55. Pullup/Pulldown Select Register (PUPD_SEL) Default Values

Bit	Field	Default Value	Description
31	PUPDSEL[31]	1	Pin Group CP[31] is configured for pull-up by default.
30	PUPDSEL[30]	1	Pin Group CP[30] is configured for pull-up by default.
29	PUPDSEL[29]	0	Pin Group CP[29] is configured for pull-down by default.
28	PUPDSEL[28]	0	Pin Group CP[28] is configured for pull-down by default.
27	PUPDSEL[27]	0	Pin Group CP[27] is configured for pull-down by default.
26	PUPDSEL[26]	0	Pin Group CP[26] is configured for pull-down by default.
25	PUPDSEL[25]	1	Pin Group CP[25] is configured for pull-up by default.
24	PUPDSEL[24]	1	Pin Group CP[24] is configured for pull-up by default.
23	PUPDSEL[23]	1	Pin Group CP[23] is configured for pull-up by default.
22	PUPDSEL[22]	1	Pin Group CP[22] is configured for pull-up by default.
21	PUPDSEL[21]	1	Pin Group CP[21] is configured for pull-up by default.
20	PUPDSEL[20]	1	Pin Group CP[20] is configured for pull-up by default.
19	PUPDSEL[19]	1	Pin Group CP[19] is configured for pull-up by default.
18	PUPDSEL[18]	1	Pin Group CP[18] is configured for pull-up by default.
17	PUPDSEL[17]	1	Pin Group CP[17] is configured for pull-up by default.
16	PUPDSEL[16]	1	Pin Group CP[16] is configured for pull-up by default.
15	PUPDSEL[15]	1	Pin Group CP[15] is configured for pull-up by default.
14	PUPDSEL[14]	1	Pin Group CP[14] is configured for pull-up by default.
13	PUPDSEL[13]	1	Pin Group CP[13] is configured for pull-up by default.
12	PUPDSEL[12]	1	Pin Group CP[12] is configured for pull-up by default.
11	PUPDSEL[11]	1	Pin Group CP[11] is configured for pull-up by default.
10	PUPDSEL[10]	1	Pin Group CP[10] is configured for pull-up by default.
9	PUPDSEL[9]	1	Pin Group CP[9] is configured for pull-up by default.
8	PUPDSEL[8]	1	Pin Group CP[8] is configured for pull-up by default.
7	PUPDSEL[7]	1	Pin Group CP[7] is configured for pull-up by default.
6	PUPDSEL[6]	1	Pin Group CP[6] is configured for pull-up by default.
5	PUPDSEL[5]	1	Pin Group CP[5] is configured for pull-up by default.
4	PUPDSEL[4]	1	Pin Group CP[4] is configured for pull-up by default.
3	PUPDSEL[3]	1	Pin Group CP[3] is configured for pull-up by default.
2	PUPDSEL[2]	1	Pin Group CP[2] is configured for pull-up by default.
1	PUPDSEL[1]	1	Pin Group CP[1] is configured for pull-up by default.
0	PUPDSEL[0]	1	Pin Group CP[0] is configured for pull-up by default.

Copyright © 2013–2016, Texas Instruments Incorporated

10.5.24 RXACTIVE Control Register (RXACTIVE)

The RXACTIVE control register (RXACTIVE) enables or disables the LVCMOS receivers for the pin group *n* defined in your device-specific data manual. The RXACTIVE is shown in Figure 10-51 and described in Table 10-56.

Figure 10-51. RXACTIVE Control Register (RXACTIVE)

31	0
RXACTIVE[*n*]	
R/W-FFFF FFFFh	

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

Table 10-56. RXACTIVE Control Register (RXACTIVE) Field Descriptions

Bit	Field	Value	Description
31-0	RXACTIVE[*n*]		**Enables the LVCMOS receivers on pin group *n***. See your device-specific data manual for pin group information. Receivers should only be disabled if the associated pin group is not being used.
		0	LVCMOS receivers for pin group *n* are disabled.
		1	LVCMOS receivers for pin group *n* are enabled.

Copyright © 2013–2016, Texas Instruments Incorporated

ARM Interrupt Controller (AINTC)

Topic **Page**

11.1 Introduction **269**
11.2 Interrupt Mapping **269**
11.3 AINTC Methodology **272**
11.4 AINTC Registers **276**

Copyright © 2013–2016, Texas Instruments Incorporated

11.1 Introduction

The ARM interrupt controller (AINTC) is an interface between interrupts coming from different parts of the system (these are referred to as system interrupts in this document), and the ARM9 interrupt interface. ARM9 supports two types of interrupts: FIQ and IRQ (these are referred to as host interrupts in this document). The AINTC has the following features:

- Supports up to 101 system interrupts.
- Supports up to 32 interrupt channels.
- Channels 0 and 1 are mapped (hard-wired) to the FIQ ARM interrupt and channels 2-31 are mapped to IRQ ARM interrupt.
- Each system interrupt can be enabled and disabled.
- Each host interrupt can be enabled and disabled.
- Hardware prioritization of interrupts.
- Combining of interrupts from IPs to a single system interrupt.
- Supports two active low debug interrupts.

See the ARM926EJ Technical Reference Manual for information about the ARM's FIQ and IRQ interrupts.

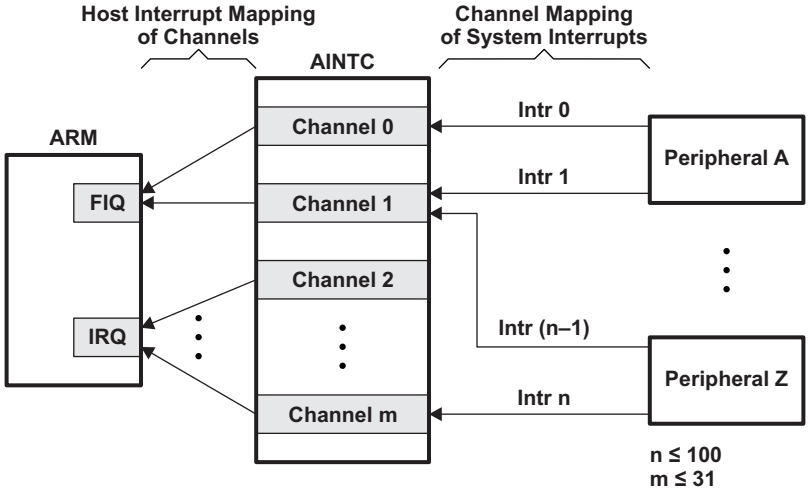
11.2 Interrupt Mapping

The AINTC supports up to 101 system interrupts from different peripherals to be mapped to 32 channels inside the AINTC (see Figure 11-1). Interrupts from these 32 channels are further mapped to either an ARM FIQ interrupt or an ARM IRQ interrupt.

- Any of the 101 system interrupts can be mapped to any of the 32 channels.
- Multiple interrupts can be mapped to a single channel.
- An interrupt should not be mapped to more than one channel.
- Interrupts from channels 0 and 1 are mapped to FIQ ARM interrupt on host side.
- Interrupts from channels 2 to 31 are mapped to IRQ ARM interrupt on host side.
- For I < k, interrupts on channel-I have higher priority than interrupts on channel-k.
- For interrupts on same channel, priority is determined by the hardware interrupt number. The lower the interrupt number, the higher the priority.

Table 11-1 shows the system interrupt assignments for the AINTC.

Figure 11-1. AINTC Interrupt Mapping



Copyright © 2013–2016, Texas Instruments Incorporated

Table 11-1. AINTC System Interrupt Assignments

Event	Interrupt Name	Source
0	COMMTX	ARM
1	COMMRX	ARM
2	NINT	ARM
3	PRU_EVTOUT0	PRUSS Interrupt
4	PRU_EVTOUT1	PRUSS Interrupt
5	PRU_EVTOUT2	PRUSS Interrupt
6	PRU_EVTOUT3	PRUSS Interrupt
7	PRU_EVTOUT4	PRUSS Interrupt
8	PRU_EVTOUT5	PRUSS Interrupt
9	PRU_EVTOUT6	PRUSS Interrupt
10	PRU_EVTOUT7	PRUSS Interrupt
11	EDMA3_0_CC0_INT0	EDMA3_0 Channel Controller 0 Shadow Region 0 Transfer Completion Interrupt
12	EDMA3_0_CC0_ERRINT	EDMA3_0 Channel Controller 0 Error Interrupt
13	EDMA3_0_TC0_ERRINT	EDMA3_0 Transfer Controller 0 Error Interrupt
14	EMIFA_INT	EMIFA Interrupt
15	IIC0_INT	I2C0 interrupt
16	MMCSD0_INT0	MMCSD0 MMC/SD Interrupt
17	MMCSD0_INT1	MMCSD0 SDIO Interrupt
18	PSC0_ALLINT	PSC0 Interrupt
19	RTC_IRQS[1:0]	RTC Interrupt
20	SPI0_INT	SPI0 Interrupt
21	T64P0_TINT12	Timer64P0 Interrupt (TINT12)
22	T64P0_TINT34	Timer64P0 Interrupt (TINT34)
23	T64P1_TINT12	Timer64P1 Interrupt (TINT12)
24	T64P1_TINT34	Timer64P1 Interrupt (TINT34)
25	UART0_INT	UART0 Interrupt
26	—	Reserved
27	PROTERR	SYSCFG Protection Shared Interrupt
28	SYSCFG_CHIPINT0	SYSCFG CHIPSIG Register
29	SYSCFG_CHIPINT1	SYSCFG CHIPSIG Register
30	SYSCFG_CHIPINT2	SYSCFG CHIPSIG Register
31	SYSCFG_CHIPINT3	SYSCFG CHIPSIG Register
32	EDMA3_0_TC1_ERRINT	EDMA3_0 Transfer Controller 1 Error Interrupt
33-40	—	Reserved
41	DDR2_MEMERR	DDR2 Controller Interrupt
42	GPIO_B0INT	GPIO Bank 0 Interrupt
43	GPIO_B1INT	GPIO Bank 1 Interrupt
44	GPIO_B2INT	GPIO Bank 2 Interrupt
45	GPIO_B3INT	GPIO Bank 3 Interrupt
46	GPIO_B4INT	GPIO Bank 4 Interrupt
47	GPIO_B5INT	GPIO Bank 5 Interrupt
48	GPIO_B6INT	GPIO Bank 6 Interrupt
49	GPIO_B7INT	GPIO Bank 7 Interrupt
50	GPIO_B8INT	GPIO Bank 8 Interrupt
51	IIC1_INT	I2C1 Interrupt
52	LCDC_INT	LCD Controller Interrupt
53	UART_INT1	UART1 Interrupt

Copyright © 2013–2016, Texas Instruments Incorporated

Table 11-1. AINTC System Interrupt Assignments (continued)

Event	Interrupt Name	Source
54	MCASP_INT	McASP0 Combined RX/TX Interrupt
55	PSC1_ALLINT	PSC1 Interrupt
56	SPI1_INT	SPI1 Interrupt
57	UHPI_ARMINT	HPI ARM Interrupt
58	USB0_INT	USB0 (USB2.0) Interrupt
59-60	—	Reserved
61	UART2_INT	UART2 Interrupt
62	—	Reserved
63	EHRPWM0	HiResTimer / PWM0 Interrupt
64	EHRPWM0TZ	HiResTimer / PWM0 Trip Zone Interrupt
65	EHRPWM1	HiResTimer / PWM1 Interrupt
66	EHRPWM1TZ	HiResTimer / PWM1 Trip Zone Interrupt
67	—	Reserved
68	T64P2_ALL	Timer64P2 Combined Interrupt (TINT12 and TINT34)
69	ECAP0	eCAP0 Interrupt
70	ECAP1	eCAP1 Interrupt
71	ECAP2	eCAP2 Interrupt
72	MMCSD1_INT0	MMCSD1 MMC/SD Interrupt
73	MMCSD1_INT1	MMCSD1 SDIO Interrupt
74	T64P2_CMPINT0	Timer64P2 - Compare Interrupt 0
75	T64P2_CMPINT1	Timer64P2 - Compare Interrupt 1
76	T64P2_CMPINT2	Timer64P2 - Compare Interrupt 2
77	T64P2_CMPINT3	Timer64P2 - Compare Interrupt 3
78	T64P2_CMPINT4	Timer64P2 - Compare Interrupt 4
79	T64P2_CMPINT5	Timer64P2 - Compare Interrupt 5
80	T64P2_CMPINT6	Timer64P2 - Compare Interrupt 6
81	T64P2_CMPINT7	Timer64P2 - Compare Interrupt 7
82	T64P3_CMPINT0	Timer64P3 - Compare Interrupt 0
83	T64P3_CMPINT1	Timer64P3 - Compare Interrupt 1
84	T64P3_CMPINT2	Timer64P3 - Compare Interrupt 2
85	T64P3_CMPINT3	Timer64P3 - Compare Interrupt 3
86	T64P3_CMPINT4	Timer64P3 - Compare Interrupt 4
87	T64P3_CMPINT5	Timer64P3 - Compare Interrupt 5
88	T64P3_CMPINT6	Timer64P3 - Compare Interrupt 6
89	T64P3_CMPINT7	Timer64P3 - Compare Interrupt 7
90	ARMCLKSTOPREQ	PSC0 Interrupt
91	uPP_ALLINT	uPP Combined Interrupt
92	VPIF_ALLINT	VPIF Combined Interrupt
93	EDMA3_1_CC0_INT0	EDMA3_1 Channel Controller 0 Shadow Region 0 Transfer Completion Interrupt
94	EDMA3_1_CC0_ERRINT	EDMA3_1 Channel Controller 0 Error Interrupt
95	EDMA3_1_TC0_ERRINT	EDMA3_1 Transfer Controller 0 Error Interrupt
96	T64P3_ALL	Timer64P3 Combined Interrupt (TINT12 and TINT34)
97	MCBSP0_RINT	McBSP0 Receive Interrupt
98	MCBSP0_XINT	McBSP0 Transmit Interrupt
99	MCBSP1_RINT	McBSP1 Receive Interrupt
100	MCBSP1_XINT	McBSP1 Transmit Interrupt

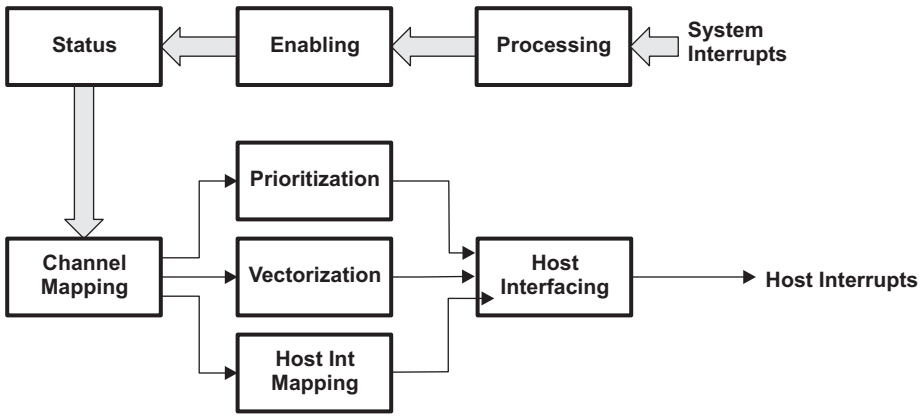
Copyright © 2013–2016, Texas Instruments Incorporated

11.3 AINTC Methodology

The AINTC module controls the system interrupt mapping to the host interrupt interface. System interrupts are generated by the device peripherals. The AINTC receives the system interrupts and maps them to internal channels. The channels are used to combine and prioritize system interrupts. These channels are then mapped onto the host interface that is typically a smaller number of host interrupts or a vector input. Interrupts from system side are active high in polarity. Also, they are pulse type of interrupts.

The AINTC encompasses many functions to process the system interrupts and prepare them for the host interface. These functions are: processing, enabling, status, channel mapping, host interrupt mapping, prioritization, vectorization, debug, and host interfacing. Figure 11-2 illustrates the flow of system interrupts through the functions to the host. The following subsections describe each part of the flow.

Figure 11-2. Flow of System Interrupts to Host



11.3.1 Interrupt Processing

The interrupt processing block does the following tasks:

- Synchronization of slower and asynchronous interrupts
- Conversion of polarity to active high
- Conversion of interrupt type to pulse interrupts

After the processing block, all interrupts will be active-high pulses.

Copyright © 2013–2016, Texas Instruments Incorporated

11.3.2 Interrupt Enabling

The AINTC interrupt enable system allows individual interrupts to be enabled or disabled. Use the following sequence to enable interrupts:

1. Enable global host interrupts. All host interrupts are enabled by setting the ENABLE bit in the global enable register (GER). Individual host interrupts are enabled or disabled from their individual enables and are not overridden by the global enable.
2. Enable host interrupt lines. Host interrupt lines (FIQ and IRQ) can be enabled through one of two methods:
   (a) Set the desired mapped bit(s) in the host interrupt enable register (HIER), or
   (b) Write the host interrupt index (0-1) to the host interrupt enable indexed set register (HIEISR) for every interrupt line to enable.
3. Enable system interrupts. System interrupts can be individually enabled through one of two methods:
   (a) Set the desired mapped bit(s) in the system interrupt enable set registers (ESR1-ESR4), or
   (b) Write the system interrupt index (0-100) to the system interrupt enable indexed set register (EISR) for every system interrupt to enable.

11.3.3 Interrupt Status Checking

The next stage is to capture which system interrupts are pending. There are two kinds of pending status: raw status and enabled status. Raw status is the pending status of the system interrupt without regards to the enable bit for the system interrupt. Enabled status is the pending status of the system interrupts with the enable bits active. When the enable bit is inactive, the enabled status will always be inactive.

The enabled status of system interrupts is captured in system interrupt status enabled/clear registers (SECR1-SECR4). Status of system interrupt 'N' is indicated by the Nth bit of SECR1-SECR4. Since there exists 101 system interrupts, four 32-bit registers are used to capture the enabled status of interrupts.

The pending status reflects whether the system interrupt occurred since the last time the status register bit was cleared. Each bit in the status register is individually clearable.

11.3.4 Interrupt Channel Mapping

The AINTC has 32 internal channels to which enabled system interrupts can be mapped. Higher priority interrupts should be mapped to channels 0 and 1. Other interrupts can be mapped to any of the channels from 2 to 31. Channel 0 has highest priority and channel 31 has the lowest priority. Channels 0 and 1 are connected to FIQ ARM interrupt. Channels 2 to 31 are connected to IRQ ARM interrupt. Channels are used to group the system interrupts into a smaller number of priorities that can be given to a host interface with a very small number of interrupt inputs. When multiple system interrupts are mapped to the same channel their interrupts are ORed together so that when either is active the output is active.

The channel map registers (CMR*m*) define the channel for each system interrupt. There is one register per 4 system interrupts; therefore, there are 26 channel map registers (CMR0-CMR25) for a system of 101 interrupts. Channel for each system interrupt can be set using these registers.

11.3.5 Host Interrupt Mapping Interrupts

The Host is ARM9, which has two lines: FIQ and IRQ. The 32 channels from the AINTC are mapped to these two lines. The AINTC has a fixed host interrupt mapping scheme. Channels 0 and 1 are mapped to FIQ and channels 2-31 are mapped to IRQ. Thus, system interrupts mapped to channels 0 and 1 are propagated as FIQ to the host and system interrupts mapped to channels 2-31 are propagated as IRQ to the host. When multiple channels are mapped to the same host interrupt, then prioritization is done to select which interrupt is in the highest-priority channel and which should be sent first to the host.

Copyright © 2013–2016, Texas Instruments Incorporated

11.3.6 Interrupt Prioritization

The next stage of the AINTC is prioritization. Since multiple interrupts feed into a single channel and multiple channels feed into a single host interrupt, it is necessary to prioritize between all the system interrupts/channels to decide on a single system interrupt to handle. The AINTC provides hardware to perform this prioritization with a given scheme so that software does not have to do this. There are two levels of prioritizations:

1. The first level of prioritization is between the active channels for a host interrupt. Channel 0 has the highest priority and channel 31 has the lowest. So the first level of prioritization picks the lowest numbered active channel.
2. The second level of prioritization is between the active system interrupts for the prioritized channel. The system interrupt in vector position 0 has the highest priority and system interrupt 100 has the lowest priority. So the second level of prioritization picks the lowest vector position active system interrupt.

The prioritized system interrupt for each host interrupt line (FIQ and IRQ) can be obtained from the host interrupt prioritized index registers (HIPIR1 and HIPIR2). The host interrupt prioritized index register values update dynamically as interrupts arrive at AINTC so care should be taken to avoid register race conditions.

The AINTC features a prioritization hold mode that is intended to prevent race conditions while servicing interrupts. This mode is enabled by setting the priority hold mode (PRHOLDMODE) bit in the control register (CR). When enabled, a read of either the host interrupt prioritized index register (HIPIR*n*) or the host interrupt prioritized vector register (HIPVR*n*) will freeze both the HIPIR*n* and HIPVR*n* values for the respective host interrupt *n*. The values are frozen until one of the following actions is taken to release the registers:

1. Write to the host interrupt prioritized index register (HIPIR*n*)
2. Write to the host interrupt prioritized vector register (HIPVR*n*)
3. Write-set bit *n* of the host interrupt enable register (HIER)
4. Write-set the active interrupt index to the host interrupt enable index set register (HIEISR)
5. Write-clear the active interrupt index to the host interrupt enable index clear register (HIEICR)

11.3.7 Interrupt Nesting

If interrupt service routines (ISRs) consume a large number of CPU cycles and may delay the servicing of other interrupts, the AINTC can perform a nesting function in its prioritization. Nesting is a method of disabling certain interrupts (usually lower-priority interrupts) when an interrupt is taken so that only those desired interrupts can trigger to the host while it is servicing the current interrupt. The typical usage is to nest on the current interrupt and disable all interrupts of the same or lower priority (or channel). Then the host will only be interrupted from a higher priority interrupt.

Nesting is available in 1 of 3 methods selectable by the NESTMODE bit in the control register (CR):

1. Nesting for all host interrupts, based on channel priority: When an interrupt is taken, the nesting level is set to its channel priority. From then, that channel priority and all lower priority channels will be disabled from generating host interrupts and only higher priority channels are allowed. When the interrupt is completely serviced, the nesting level is returned to its original value. When there is no interrupt being serviced, there are no channels disabled due to nesting. The global nesting level register (GNLR) allows the checking and setting of the global nesting level across all host interrupts. The nesting level is the channel (and all of lower priority channels) that are nested out because of a current interrupt.
2. Nesting for individual host interrupts, based on channel priority: Always nest based on channel priority for each host interrupt individually. When an interrupt is taken on a host interrupt, then, the nesting level is set to its channel priority for just that host interrupt, and other host interrupts do not have their nesting affected. Then for that host interrupt, equal or lower priority channels will not interrupt the host but may on other host interrupts if programmed. When the interrupt is completely serviced the nesting level for the host interrupt is returned to its original value. The host interrupt nesting level registers (HINLR1 and HINLR2) display and control the nesting level for each host interrupt. The nesting level controls which channel and lower priority channels are nested. There is one register per host interrupt.

Copyright © 2013–2016, Texas Instruments Incorporated

3. Software manually performs the nesting of interrupts. When an interrupt is taken, the software will disable all the host interrupts, manually update the enables for any or all the system interrupts, and then re-enable all the host interrupts. This now allows only the system interrupts that are still enabled to trigger to the host. When the interrupt is completely serviced the software must reverse the changes to re-enable the nested out system interrupts. This method requires the most software interaction but gives the most flexibility if simple channel based nesting mechanisms are not adequate.

The recommended approach is the automatic host interrupt nesting method (second method). Because higher priority interrupts can preempt lower priority interrupts in this method, a software stack is used to keep track of nest priorities. The base stack value should be initialized to the default nest priority of the application. Take the following steps within the ARM hardware interrupt service routine to handle interrupts using host interrupt priority nesting:

1. Disable the ARM hardware interrupt.
2. Clear the OVERRIDE bit in the host interrupt nesting level register *n* (HINLR*n*) to expose the priority level of the active interrupt.
3. Push the active (or desired) interrupt priority value into the nest priority stack.
4. Write the active (or desired) priority level into HINLR*n* by setting the OVERRIDE bit.
5. Calculate and store the ISR address for the active interrupt. Unfreeze the host interrupt prioritized index register *n* (HIPIR*n*) and the host interrupt prioritized vector register *n* (HIPVR*n*), if the PRHOLDMODE bit in the control register (CR) is set.
6. Clear the system interrupt status by setting the appropriate bit in the system interrupt status enabled/clear register *n* (SECR*n*) or by writing the appropriate index to the system interrupt status indexed clear register (SICR).
7. Acknowledge and enable the ARM hardware interrupt.
8. Execute the ISR at the address stored from step 5. During this step, interrupts enabled by the new nest priority level will be able to preempt the ISR.
9. Disable the ARM hardware interrupt.
10. Discard the most recent priority level in the nest priority stack and restore the previous priority level to HINLR*n* by setting the OVERRIDE bit.
11. Enable the ARM hardware interrupt.

11.3.8 Interrupt Vectorization

The next stage of the AINTC is vectorization. Vectorization is an advanced feature that allows the host to receive an interrupt service routine (ISR) address in addition to just the interrupt status. Without vectorization the host would receive the interrupt and enter a general ISR that gets the prioritized system interrupt to service from the AINTC, looks up the specific ISR address for that system interrupt, and then jumps to that address. With vectorization the host can read a register that has the ISR address already calculated and jump to that address immediately.

Vectorization uses a base and universal size where all the ISR code is placed in a contiguous memory region with each ISR code a standard size. For this calculation, the vector base register (VBR) is programmed by software to hold the base address of all the ISR code and the vector size register (VSR) is programmed for the size in words between ISR code for each system interrupt. The index number of each system interrupt is used to calculate the final offset. The specific system interrupt ISR address is then calculated as:

ISR address = base + (index × size)

There is also a special case when there is no interrupt pending and then the ISR address is the ISR Null address. This is in case the vector address is executed when there is no pending interrupt so that a Null handler can be in place to just return from the interrupt. The vector null address register (VNR) holds the address of the ISR null address. When there is a pending interrupt then the ISR address is calculated as *exact base + offset* for that interrupt number.

Copyright © 2013–2016, Texas Instruments Incorporated

11.3.9 Interrupt Status Clearing

After servicing the interrupt (after execution of the ISR), interrupt status is to be cleared. If a system interrupt status is not cleared, then another host interrupt may not be triggered or another host interrupt may be triggered incorrectly. For clearing the status of an interrupt, whose interrupt number is N, write a 1 to the Nth bit position in the system interrupt status enabled/clear registers (SECR1-SECR4). System interrupt N can also be cleared by writing the value N into the system interrupt status indexed clear register (SICR).

11.3.10 Interrupt Disabling

At any time, if any interrupt is not to be propagated to the host, then that interrupt should be disabled. For disabling an interrupt whose interrupt number is N, write a 1 to the Nth bit in the system interrupt enable clear registers (ECR1-ECR4). System interrupt N can also be disabled by writing the value N in the system interrupt enable indexed clear register (EICR).

11.4 AINTC Registers

Table 11-2 lists the memory-mapped registers for the AINTC.

Table 11-2. ARM Interrupt Controller (AINTC) Registers

Address	Acronym	Register Description	Section
FFFE E000h	REVID	Revision Identification Register	Section 11.4.1
FFFE E004h	CR	Control Register	Section 11.4.2
FFFE E010h	GER	Global Enable Register	Section 11.4.3
FFFE E01Ch	GNLR	Global Nesting Level Register	Section 11.4.4
FFFE E020h	SISR	System Interrupt Status Indexed Set Register	Section 11.4.5
FFFE E024h	SICR	System Interrupt Status Indexed Clear Register	Section 11.4.6
FFFE E028h	EISR	System Interrupt Enable Indexed Set Register	Section 11.4.7
FFFE E02Ch	EICR	System Interrupt Enable Indexed Clear Register	Section 11.4.8
FFFE E034h	HIEISR	Host Interrupt Enable Indexed Set Register	Section 11.4.9
FFFE E038h	HIEICR	Host Interrupt Enable Indexed Clear Register	Section 11.4.10
FFFE E050h	VBR	Vector Base Register	Section 11.4.11
FFFE E054h	VSR	Vector Size Register	Section 11.4.12
FFFE E058h	VNR	Vector Null Register	Section 11.4.13
FFFE E080h	GPIR	Global Prioritized Index Register	Section 11.4.14
FFFE E084h	GPVR	Global Prioritized Vector Register	Section 11.4.15
FFFE E200h	SRSR1	System Interrupt Status Raw/Set Register 1	Section 11.4.16
FFFE E204h	SRSR2	System Interrupt Status Raw/Set Register 2	Section 11.4.17
FFFE E208h	SRSR3	System Interrupt Status Raw/Set Register 3	Section 11.4.18
FFFE E20Ch	SRSR4	System Interrupt Status Raw/Set Register 4	Section 11.4.19
FFFE E280h	SECR1	System Interrupt Status Enabled/Clear Register 1	Section 11.4.20
FFFE E284h	SECR2	System Interrupt Status Enabled/Clear Register 2	Section 11.4.21
FFFE E288h	SECR3	System Interrupt Status Enabled/Clear Register 3	Section 11.4.22
FFFE E28Ch	SECR4	System Interrupt Status Enabled/Clear Register 4	Section 11.4.23
FFFE E300h	ESR1	System Interrupt Enable Set Register 1	Section 11.4.24
FFFE E304h	ESR2	System Interrupt Enable Set Register 2	Section 11.4.25
FFFE E308h	ESR3	System Interrupt Enable Set Register 3	Section 11.4.26
FFFE E30Ch	ESR4	System Interrupt Enable Set Register 4	Section 11.4.27
FFFE E380h	ECR1	System Interrupt Enable Clear Register 1	Section 11.4.28
FFFE E384h	ECR2	System Interrupt Enable Clear Register 2	Section 11.4.29
FFFE E388h	ECR3	System Interrupt Enable Clear Register 3	Section 11.4.30

Copyright © 2013–2016, Texas Instruments Incorporated

Table 11-2. ARM Interrupt Controller (AINTC) Registers (continued)

Address	Acronym	Register Description	Section
FFFE E38Ch	ECR4	System Interrupt Enable Clear Register 4	Section 11.4.31
FFFE E400h–FFFE E464h	CMR0-CMR25	Channel Map Registers 0-25	Section 11.4.32
FFFE E900h	HIPIR1	Host Interrupt Prioritized Index Register 1	Section 11.4.33
FFFE E904h	HIPIR2	Host Interrupt Prioritized Index Register 2	Section 11.4.34
FFFE F100h	HINLR1	Host Interrupt Nesting Level Register 1	Section 11.4.35
FFFE F104h	HINLR2	Host Interrupt Nesting Level Register 2	Section 11.4.36
FFFE F500h	HIER	Host Interrupt Enable Register	Section 11.4.37
FFFE F600h	HIPVR1	Host Interrupt Prioritized Vector Register 1	Section 11.4.38
FFFE F604h	HIPVR2	Host Interrupt Prioritized Vector Register 2	Section 11.4.39

11.4.1 Revision Identification Register (REVID)

The revision identification register (REVID) is shown in Figure 23-35 and described in Table 11-3.

Figure 11-3. Revision Identification Register (REVID)

31	0
REV	
R-4E82 A900h	

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

Table 11-3. Revision Identification Register (REVID) Field Descriptions

Bit	Field	Value	Description
31-0	REV	4E82 A900h	Revision ID of the AINTC.

Copyright © 2013–2016, Texas Instruments Incorporated

11.4.2 Control Register (CR)

The control register (CR) holds global control parameters. The CR is shown in Figure 11-4 and described in Table 11-4.

Figure 11-4. Control Register (CR)

31	16
Reserved	
R-0	

15	5	4	3	2	1	0
Reserved		PRHOLDMODE	NESTMODE		Reserved	
R-0		R/W-0	R/W-0		R-0	

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

Table 11-4. Control Register (CR) Field Descriptions

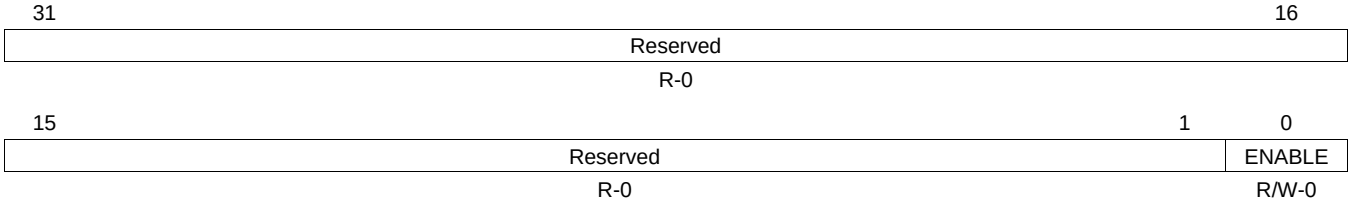
Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4	PRHOLDMODE		Enables priority holding mode.
		0	No priority holding. Prioritized MMRs will continually update.
		1	Priority holding enabled. Prioritized Index and Vector Address MMRs will hold their value after the first is read. See Section 11.3.6 for details.
3-2	NESTMODE	0-3h	Nesting mode.
		0	No nesting
		1h	Automatic individual nesting (per host interrupt)
		2h	Automatic global nesting (over all host interrupts)
		3h	Manual nesting
1-0	Reserved	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

11.4.3 Global Enable Register (GER)

The global enable register (GER) enables all the host interrupts. Individual host interrupts are still enabled or disabled from their individual enables and are not overridden by the global enable. The GER is shown in Figure 11-5 and described in Table 11-5.

Figure 11-5. Global Enable Register (GER)



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

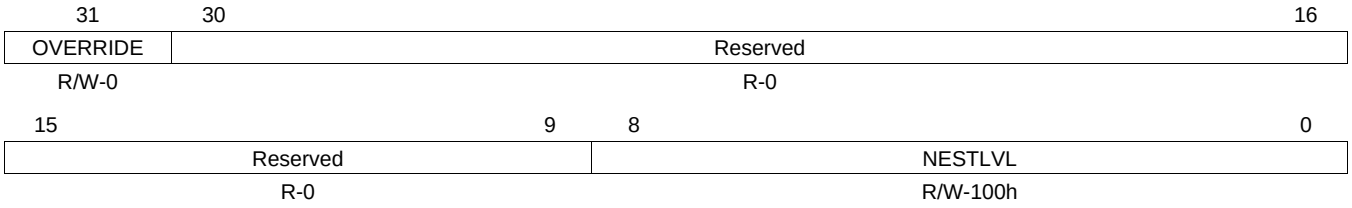
Table 11-5. Global Enable Register (GER) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	ENABLE	0-1	The current global enable value when read. Writes set the global enable.

11.4.4 Global Nesting Level Register (GNLR)

The global nesting level register (GNLR) allows the checking and setting of the global nesting level across all host interrupts when automatic global nesting mode is set. The nesting level is the channel (and all of lower priority) that are nested out because of a current interrupt. The GNLR is shown in Figure 11-6 and described in Table 11-6.

Figure 11-6. Global Nesting Level Register (GNLR)



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

Table 11-6. Global Nesting Level Register (GNLR) Field Descriptions

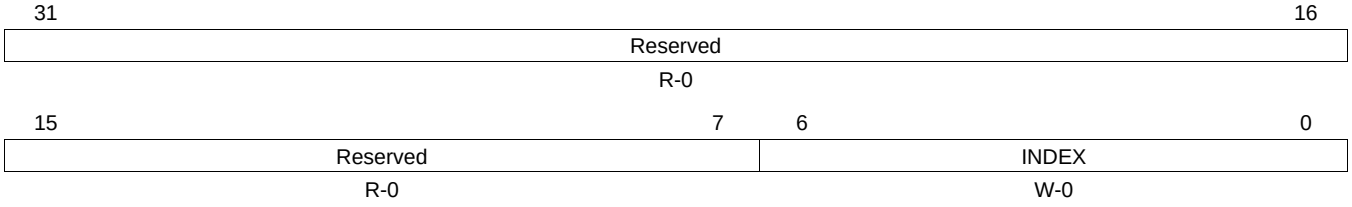
Bit	Field	Value	Description
31	OVERRIDE	0-1	Always read as 0. Writes of 1 override the automatic nesting and set the NESTLVL to the written data.
30-9	Reserved	0	Reserved
8-0	NESTLVL	0-1FFh	The current global nesting level (highest channel that is nested). Writes set the nesting level. In autonesting mode this value is updated internally, unless the auto_override bit is set.

Copyright © 2013–2016, Texas Instruments Incorporated

11.4.5 System Interrupt Status Indexed Set Register (SISR)

The system interrupt status indexed set register (SISR) allows setting the status of an interrupt. The interrupt to set is the INDEX value written. This sets the Raw Status Register bit of the given INDEX. The SISR is shown in Figure 11-7 and described in Table 11-7.

Figure 11-7. System Interrupt Status Indexed Set Register (SISR)



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

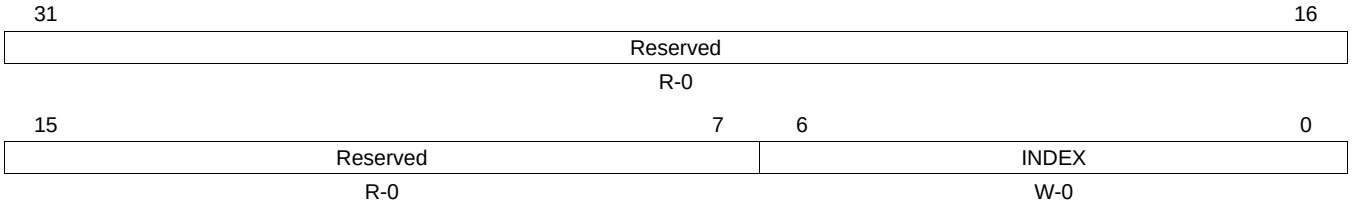
Table 11-7. System Interrupt Status Indexed Set Register (SISR) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reserved
6-0	INDEX	0-7Fh	Writes set the status of the interrupt given in the INDEX value. Reads return 0.

11.4.6 System Interrupt Status Indexed Clear Register (SICR)

The system interrupt status indexed clear register (SICR) allows clearing the status of an interrupt. The interrupt to clear is the INDEX value written. This clears the Raw Status Register bit of the given INDEX. The SICR is shown in Figure 11-8 and described in Table 11-8.

Figure 11-8. System Interrupt Status Indexed Clear Register (SICR)



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

Table 11-8. System Interrupt Status Indexed Clear Register (SICR) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reserved
6-0	INDEX	0-7Fh	Writes clear the status of the interrupt given in the INDEX value. Reads return 0.

Copyright © 2013–2016, Texas Instruments Incorporated

11.4.7 System Interrupt Enable Indexed Set Register (EISR)

The system interrupt enable indexed set register (EISR) allows enabling an interrupt. The interrupt to enable is the INDEX value written. This sets the Enable Register bit of the given INDEX. The EISR is shown in Figure 11-9 and described in Table 11-9.

Figure 11-9. System Interrupt Enable Indexed Set Register (EISR)

31	16
Reserved	
R-0	

15	7	6	0
Reserved		INDEX	
R-0		W-0	

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

Table 11-9. System Interrupt Enable Indexed Set Register (EISR) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reserved
6-0	INDEX	0-7Fh	Writes set the enable of the interrupt given in the INDEX value. Reads return 0.

11.4.8 System Interrupt Enable Indexed Clear Register (EICR)

The system interrupt enable indexed clear register (EICR) allows disabling an interrupt. The interrupt to disable is the INDEX value written. This clears the Enable Register bit of the given INDEX. The EICR is shown in Figure 11-10 and described in Table 11-10.

Figure 11-10. System Interrupt Enable Indexed Clear Register (EICR)

31	16
Reserved	
R-0	

15	7	6	0
Reserved		INDEX	
R-0		W-0	

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

Table 11-10. System Interrupt Enable Indexed Clear Register (EICR) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reserved
6-0	INDEX	0-7Fh	Writes clear the enable of the interrupt given in the INDEX value. Reads return 0.

Copyright © 2013–2016, Texas Instruments Incorporated

11.4.9 Host Interrupt Enable Indexed Set Register (HIEISR)

The host interrupt enable indexed set register (HIEISR) allows enabling a host interrupt output. The host interrupt to enable is the INDEX value written. This enables the host interrupt output or triggers the output again if already enabled. The HEISR is shown in Figure 11-11 and described in Table 11-11.

Figure 11-11. Host Interrupt Enable Indexed Set Register (HEISR)

31	16
Reserved	
R-0	

15	1	0
Reserved		INDEX
R-0		W-0

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

Table 11-11. Host Interrupt Enable Indexed Set Register (HEISR) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	INDEX		Writes set the enable of the host interrupt given in the INDEX value. Reads return 0.
		0	Writing a 0 sets FIQ.
		1	Writing a 1 sets IRQ.

11.4.10 Host Interrupt Enable Indexed Clear Register (HIEICR)

The host interrupt enable indexed clear register (HIEICR) allows disabling a host interrupt output. The host interrupt to disable is the INDEX value written. This disables the host interrupt output. The HIEICR is shown in Figure 11-12 and described in Table 11-12.

Figure 11-12. Host Interrupt Enable Indexed Clear Register (HIEICR)

31	16
Reserved	
R-0	

15	1	0
Reserved		INDEX
R-0		W-0

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

Table 11-12. Host Interrupt Enable Indexed Clear Register (HIEICR) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	INDEX		Writes clear the enable of the host interrupt given in the INDEX value. Reads return 0.
		0	Writing a 0 clears FIQ.
		1	Writing a 1 clears IRQ.

Copyright © 2013–2016, Texas Instruments Incorporated

11.4.11 Vector Base Register (VBR)

The vector base register (VBR) holds the base address of the ISR vector addresses. The VBR is shown in Figure 11-13 and described in Table 11-13.

Figure 11-13. Vector Base Register (VBR)

31	0
BASE	
R/W-0	

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

Table 11-13. Vector Base Register (VBR) Field Descriptions

Bit	Field	Value	Description
31-0	BASE	0-FFFF FFFFh	ISR Base Address.

11.4.12 Vector Size Register (VSR)

The vector size register (VSR) holds the sizes of the individual ISR routines in the vector table. This is only the sizes to space the calculated vector addresses for the initial ISR targets (the ISR targets could branch off to the full ISR routines). The VSR is shown in Figure 11-14 and described in Table 11-14.

NOTE: The VSR must be configured even if the desired value is equal to the default value.

Figure 11-14. Vector Size Register (VSR)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		SIZE	
R-0		R/W-0	

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

Table 11-14. Vector Size Register (VSR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	SIZE	0-FFh	Size of ISR address spaces.
		0	4 bytes
		1h	8 bytes
		2h	16 bytes
		3h	32 bytes
		4h	64 bytes
		5h-FFh	...

Copyright © 2013–2016, Texas Instruments Incorporated

11.4.13 Vector Null Register (VNR)

The vector null register (VNR) holds the address of the ISR null address that handles no pending interrupts (if accidentally branched to when no interrupts are pending). The VNR is shown in Figure 11-15 and described in Table 11-15.

Figure 11-15. Vector Null Register (VNR)

31	0
NULL	
R/W-0	

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

Table 11-15. Vector Null Register (VNR) Field Descriptions

Bit	Field	Value	Description
31-0	NULL	0-FFFF FFFFh	ISR Null Address.

11.4.14 Global Prioritized Index Register (GPIR)

The global prioritized index register (GPIR) shows the interrupt number of the highest priority interrupt pending across all the host interrupts. The GPIR is shown in Figure 11-16 and described in Table 11-16.

Figure 11-16. Global Prioritized Index Register (GPIR)

31	30	16
NONE	Reserved	
R-1	R-0	

15	10	9	0
Reserved		PRI_INDX	
R-0		R-0	

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

Table 11-16. Global Prioritized Index Register (GPIR) Field Descriptions

Bit	Field	Value	Description
31	NONE	0-1	No Interrupt is pending. Can be used by host to test for a negative value to see if no interrupts are pending.
30-10	Reserved	0	Reserved
9-0	PRI_INDX	0-3FFh	The currently highest priority interrupt index pending across all the host interrupts.

Copyright © 2013–2016, Texas Instruments Incorporated

11.4.15 Global Prioritized Vector Register (GPVR)

The global prioritized vector register (GPVR) shows the interrupt vector address of the highest priority interrupt pending across all the host interrupts. The GPVR is shown in Figure 11-17 and described in Table 11-17.

Figure 11-17. Global Prioritized Vector Register (GPVR)

31	0
ADDR	
R-0	

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

Table 11-17. Global Prioritized Vector Register (GPVR) Field Descriptions

Bit	Field	Value	Description
31-0	ADDR	0-FFFF FFFFh	The currently highest priority interrupts vector address across all the host interrupts.

11.4.16 System Interrupt Status Raw/Set Register 1 (SRSR1)

The system interrupt status raw/set register 1 (SRSR1) shows the pending enabled status of the system interrupts 0 to 31. Software can write to SRSR1 to set a system interrupt without a hardware trigger. There is one bit per system interrupt. The SRSR1 is shown in Figure 11-18 and described in Table 11-18.

Figure 11-18. System Interrupt Status Raw/Set Register 1 (SRSR1)

31	0
RAW_STATUS[*n*]	
W-0	

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

Table 11-18. System Interrupt Status Raw/Set Register 1 (SRSR1) Field Descriptions

Bit	Field	Value	Description
31-0	RAW_STATUS[*n*]		System interrupt raw status and setting of the system interrupts 0 to 31. Reads return the raw status.
		0	Writing a 0 has no effect.
		1	Write a 1 in bit position [*n*] to set the status of the system interrupt *n*.

Copyright © 2013–2016, Texas Instruments Incorporated

11.4.17 System Interrupt Status Raw/Set Register 2 (SRSR2)

The system interrupt status raw/set register 2 (SRSR2) shows the pending enabled status of the system interrupts 32 to 63. Software can write to SRSR2 to set a system interrupt without a hardware trigger. There is one bit per system interrupt. The SRSR2 is shown in Figure 11-19 and described in Table 11-19.

Figure 11-19. System Interrupt Status Raw/Set Register 2 (SRSR2)

31	0
RAW_STATUS[*n*]	
W-0	

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

Table 11-19. System Interrupt Status Raw/Set Register 2 (SRSR2) Field Descriptions

Bit	Field	Value	Description
31-0	RAW_STATUS[*n*]		System interrupt raw status and setting of the system interrupts 32 to 63. Reads return the raw status.
		0	Writing a 0 has no effect.
		1	Write a 1 in bit position [*n*] to set the status of the system interrupt *n* + 32.

11.4.18 System Interrupt Status Raw/Set Register 3 (SRSR3)

The system interrupt status raw/set register 3 (SRSR3) shows the pending enabled status of the system interrupts 64 to 95. Software can write to SRSR3 to set a system interrupt without a hardware trigger. There is one bit per system interrupt. The SRSR3 is shown in Figure 11-20 and described in Table 11-20.

Figure 11-20. System Interrupt Status Raw/Set Register 3 (SRSR3)

31	0
RAW_STATUS[*n*]	
W-0	

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

Table 11-20. System Interrupt Status Raw/Set Register 3 (SRSR3) Field Descriptions

Bit	Field	Value	Description
31-0	RAW_STATUS[*n*]		System interrupt raw status and setting of the system interrupts 64 to 95. Reads return the raw status.
		0	Writing a 0 has no effect.
		1	Write a 1 in bit position [*n*] to set the status of the system interrupt *n* + 64.

Copyright © 2013–2016, Texas Instruments Incorporated

11.4.19 System Interrupt Status Raw/Set Register 4 (SRSR4)

The system interrupt status raw/set register 4 (SRSR4) shows the pending enabled status of the system interrupts 96 to 100. Software can write to SRSR4 to set a system interrupt without a hardware trigger. There is one bit per system interrupt. The SRSR4 is shown in Figure 11-21 and described in Table 11-21.

Figure 11-21. System Interrupt Status Raw/Set Register 4 (SRSR4)

31	5	4	0
Reserved		RAW_STATUS[*n*]	
R-0		W-0	

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

Table 11-21. System Interrupt Status Raw/Set Register 4 (SRSR4) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4-0	RAW_STATUS[*n*]		System interrupt raw status and setting of the system interrupts 96 to 100. Reads return the raw status.
		0	Writing a 0 has no effect.
		1	Write a 1 in bit position [*n*] to set the status of the system interrupt *n* + 96.

11.4.20 System Interrupt Status Enabled/Clear Register 1 (SECR1)

The system interrupt status enabled/clear register 1 (SECR1) shows the pending enabled status of the system interrupts 0 to 31. Software can write to SECR1 to clear a system interrupt after it has been serviced. If a system interrupt status is not cleared then another host interrupt may not be triggered or another host interrupt may be triggered incorrectly. There is one bit per system interrupt. The SECR1 is shown in Figure 11-22 and described in Table 11-22.

Figure 11-22. System Interrupt Status Enabled/Clear Register 1 (SECR1)

31	0
ENBL_STATUS[*n*]	
W-0	

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

Table 11-22. System Interrupt Status Enabled/Clear Register 1 (SECR1) Field Descriptions

Bit	Field	Value	Description
31-0	ENBL_STATUS[*n*]		System interrupt enabled status and clearing of the system interrupts 0 to 31. Reads return the enabled status (before enabling with the Enable Registers).
		0	Writing a 0 has no effect.
		1	Write a 1 in bit position [*n*] to clear the status of the system interrupt *n*.

Copyright © 2013–2016, Texas Instruments Incorporated

11.4.21 System Interrupt Status Enabled/Clear Register 2 (SECR2)

The system interrupt status enabled/clear register 2 (SECR2) shows the pending enabled status of the system interrupts 32 to 63. Software can write to SECR2 to clear a system interrupt after it has been serviced. If a system interrupt status is not cleared then another host interrupt may not be triggered or another host interrupt may be triggered incorrectly. There is one bit per system interrupt. The SECR2 is shown in Figure 11-23 and described in Table 11-23.

Figure 11-23. System Interrupt Status Enabled/Clear Register 2 (SECR2)

31	0
ENBL_STATUS[*n*]	
W-0	

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

Table 11-23. System Interrupt Status Enabled/Clear Register 2 (SECR2) Field Descriptions

Bit	Field	Value	Description
31-0	ENBL_STATUS[*n*]		System interrupt enabled status and clearing of the system interrupts 32 to 63. Reads return the enabled status (before enabling with the Enable Registers).
		0	Writing a 0 has no effect.
		1	Write a 1 in bit position [*n*] to clear the status of the system interrupt *n* + 32.

11.4.22 System Interrupt Status Enabled/Clear Register 3 (SECR3)

The system interrupt status enabled/clear register 3 (SECR3) shows the pending enabled status of the system interrupts 64 to 95. Software can write to SECR3 to clear a system interrupt after it has been serviced. If a system interrupt status is not cleared then another host interrupt may not be triggered or another host interrupt may be triggered incorrectly. There is one bit per system interrupt. The SECR3 is shown in Figure 11-24 and described in Table 11-24.

Figure 11-24. System Interrupt Status Enabled/Clear Register 3 (SECR3)

31	0
ENBL_STATUS[*n*]	
W-0	

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

Table 11-24. System Interrupt Status Enabled/Clear Register 3 (SECR3) Field Descriptions

Bit	Field	Value	Description
31-0	ENBL_STATUS[*n*]		System interrupt enabled status and clearing of the system interrupts 64 to 95. Reads return the enabled status (before enabling with the Enable Registers).
		0	Writing a 0 has no effect.
		1	Write a 1 in bit position [*n*] to clear the status of the system interrupt *n* + 64.

Copyright © 2013–2016, Texas Instruments Incorporated

11.4.23 System Interrupt Status Enabled/Clear Register 4 (SECR4)

The system interrupt status enabled/clear register 4 (SECR4) shows the pending enabled status of the system interrupts 96 to 100. Software can write to SECR4 to clear a system interrupt after it has been serviced. If a system interrupt status is not cleared then another host interrupt may not be triggered or another host interrupt may be triggered incorrectly. There is one bit per system interrupt. The SECR4 is shown in Figure 11-25 and described in Table 11-25.

Figure 11-25. System Interrupt Status Enabled/Clear Register 4 (SECR4)

31	5	4	0
Reserved		ENBL_STATUS[*n*]	
R-0		W-0	

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

Table 11-25. System Interrupt Status Enabled/Clear Register 4 (SECR4) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4-0	ENBL_STATUS[*n*]		System interrupt enabled status and clearing of the system interrupts 96 to 100. Reads return the enabled status (before enabling with the Enable Registers).
		0	Writing a 0 has no effect.
		1	Write a 1 in bit position [*n*] to clear the status of the system interrupt *n* + 96.

11.4.24 System Interrupt Enable Set Register 1 (ESR1)

The system interrupt enable set register 1 (ESR1) enables system interrupts 0 to 31 to trigger outputs. System interrupts that are not enabled do not interrupt the host. There is one bit per system interrupt. The ESR1 is shown in Figure 11-26 and described in Table 11-26.

Figure 11-26. System Interrupt Enable Set Register 1 (ESR1)

31	0
ENABLE[*n*]	
W-0	

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

Table 11-26. System Interrupt Enable Set Register 1 (ESR1) Field Descriptions

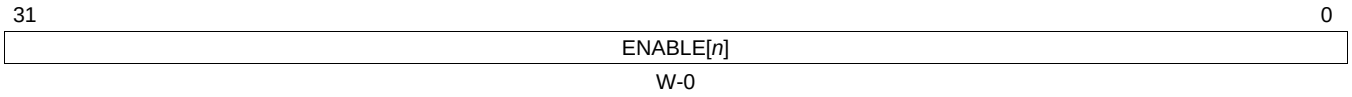
Bit	Field	Value	Description
31-0	ENABLE[*n*]		System interrupt 0 to 31 enable. Read returns the enable value (0 = disabled, 1 = enabled).
		0	Writing a 0 has no effect.
		1	Write a 1 in bit position [*n*] to set the enable for system interrupt *n*.

Copyright © 2013–2016, Texas Instruments Incorporated

11.4.25 System Interrupt Enable Set Register 2 (ESR2)

The system interrupt enable set register 2 (ESR2) enables system interrupts 32 to 63 to trigger outputs. System interrupts that are not enabled do not interrupt the host. There is one bit per system interrupt. The ESR2 is shown in Figure 11-27 and described in Table 11-27.

Figure 11-27. System Interrupt Enable Set Register 2 (ESR2)



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

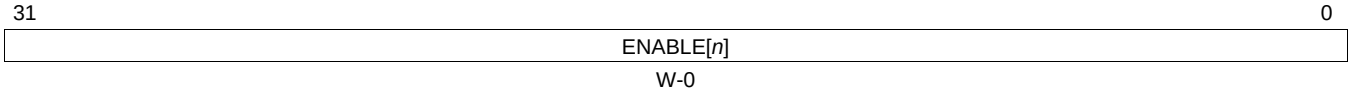
Table 11-27. System Interrupt Enable Set Register 2 (ESR2) Field Descriptions

Bit	Field	Value	Description
31-0	ENABLE[*n*]		System interrupt 32 to 63 enable. Read returns the enable value (0 = disabled, 1 = enabled).
		0	Writing a 0 has no effect.
		1	Write a 1 in bit position [*n*] to set the enable for system interrupt *n* + 32.

11.4.26 System Interrupt Enable Set Register 3 (ESR3)

The system interrupt enable set register 3 (ESR3) enables system interrupts 64 to 95 to trigger outputs. System interrupts that are not enabled do not interrupt the host. There is one bit per system interrupt. The ESR3 is shown in Figure 11-28 and described in Table 11-28.

Figure 11-28. System Interrupt Enable Set Register 3 (ESR3)



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

Table 11-28. System Interrupt Enable Set Register 3 (ESR3) Field Descriptions

Bit	Field	Value	Description
31-0	ENABLE[*n*]		System interrupt 64 to 95 enable. Read returns the enable value (0 = disabled, 1 = enabled).
		0	Writing a 0 has no effect.
		1	Write a 1 in bit position [*n*] to set the enable for system interrupt *n* + 64.

Copyright © 2013–2016, Texas Instruments Incorporated

11.4.27 System Interrupt Enable Set Register 4 (ESR4)

The system interrupt enable set register 4 (ESR4) enables system interrupts 96 to 100 to trigger outputs. System interrupts that are not enabled do not interrupt the host. There is one bit per system interrupt. The ESR4 is shown in Figure 11-29 and described in Table 11-29.

Figure 11-29. System Interrupt Enable Set Register 4 (ESR4)

31	5	4	0
Reserved		ENABLE[*n*]	
R-0		W-0	

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

Table 11-29. System Interrupt Enable Set Register 4 (ESR4) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4-0	ENABLE[*n*]		System interrupt 96 to 100 enable. Read returns the enable value (0 = disabled, 1 = enabled).
		0	Writing a 0 has no effect.
		1	Write a 1 in bit position [*n*] to set the enable for system interrupt *n* + 96.

11.4.28 System Interrupt Enable Clear Register 1 (ECR1)

The system interrupt enable clear register 1 (ECR1) disables system interrupts 0 to 31 to map to channels. System interrupts that are not enabled do not interrupt the host. There is one bit per system interrupt. The ECR1 is shown in Figure 11-30 and described in Table 11-30.

Figure 11-30. System Interrupt Enable Clear Register 1 (ECR1)

31	0
DISABLE[*n*]	
W-0	

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

Table 11-30. System Interrupt Enable Clear Register 1 (ECR1) Field Descriptions

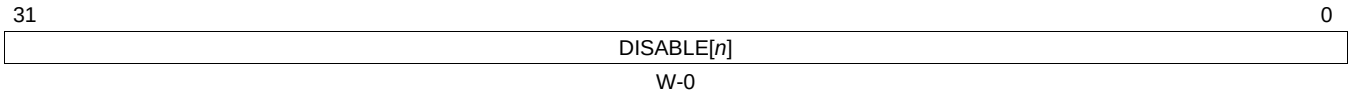
Bit	Field	Value	Description
31-0	DISABLE[*n*]		System interrupt 0 to 31 disable. Read returns the enable value (0 = disabled, 1 = enabled).
		0	Writing a 0 has no effect.
		1	Write a 1 in bit position [*n*] to clear the enable for system interrupt *n*.

Copyright © 2013–2016, Texas Instruments Incorporated

11.4.29 System Interrupt Enable Clear Register 2 (ECR2)

The system interrupt enable clear register 2 (ECR2) disables system interrupts 32 to 63 to map to channels. System interrupts that are not enabled do not interrupt the host. There is one bit per system interrupt. The ECR2 is shown in Figure 11-31 and described in Table 11-31.

Figure 11-31. System Interrupt Enable Clear Register 2 (ECR2)



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

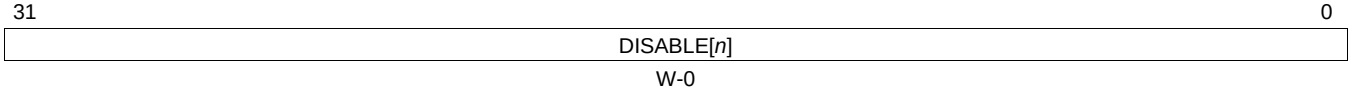
Table 11-31. System Interrupt Enable Clear Register 2 (ECR2) Field Descriptions

Bit	Field	Value	Description
31-0	DISABLE[*n*]		System interrupt 32 to 63 disable. Read returns the enable value (0 = disabled, 1 = enabled).
		0	Writing a 0 has no effect.
		1	Write a 1 in bit position [*n*] to clear the enable for system interrupt *n* + 32.

11.4.30 System Interrupt Enable Clear Register 3 (ECR3)

The system interrupt enable clear register 3 (ECR3) disables system interrupts 64 to 95 to map to channels. System interrupts that are not enabled do not interrupt the host. There is one bit per system interrupt. The ECR3 is shown in Figure 11-32 and described in Table 11-32.

Figure 11-32. System Interrupt Enable Clear Register 3 (ECR3)



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

Table 11-32. System Interrupt Enable Clear Register 3 (ECR3) Field Descriptions

Bit	Field	Value	Description
27-0	DISABLE[*n*]		System interrupt 64 to 95 disable. Read returns the enable value (0 = disabled, 1 = enabled).
		0	Writing a 0 has no effect.
		1	Write a 1 in bit position [*n*] to clear the enable for system interrupt *n* + 64.

Copyright © 2013–2016, Texas Instruments Incorporated

11.4.31 System Interrupt Enable Clear Register 4 (ECR4)

The system interrupt enable clear register 4 (ECR4) disables system interrupts 96 to 100 to map to channels. System interrupts that are not enabled do not interrupt the host. There is one bit per system interrupt. The ECR4 is shown in Figure 11-33 and described in Table 11-33.

Figure 11-33. System Interrupt Enable Clear Register 4 (ECR4)

31	5	4	0
Reserved		DISABLE[*n*]	
R-0		W-0	

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

Table 11-33. System Interrupt Enable Clear Register 4 (ECR4) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4-0	DISABLE[*n*]		System interrupt 96 to 100 disable. Read returns the enable value (0 = disabled, 1 = enabled).
		0	Writing a 0 has no effect.
		1	Write a 1 in bit position [*n*] to clear the enable for system interrupt *n* + 96.

11.4.32 Channel Map Registers (CMR0-CMR25)

The channel map registers (CMR0-CMR25) define the channel for each system interrupt. There is one register per 4 system interrupts. The CMR*n* is shown in Figure 11-34 and described in Table 11-34.

Figure 11-34. Channel Map Registers (CMR*n*)

31	24	23	16
CHNL_NPLUS3		CHNL_NPLUS2	
R/W-0		R/W-0	

15	8	7	0
CHNL_NPLUS1		CHNL_N	
R/W-0		R/W-0	

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

Table 11-34. Channel Map Registers (CMR*n*) Field Descriptions

Bit	Field	Value	Description
31-24	CHNL_NPLUS3	0-FFh	Sets the host interrupt for channel N + 3.
23-16	CHNL_NPLUS2	0-FFh	Sets the host interrupt for channel N + 2.
15-8	CHNL_NPLUS1	0-FFh	Sets the host interrupt for channel N + 1.
7-0	CHNL_N	0-FFh	Sets the channel for the system interrupt N. (N ranges from 0 to 100).

Copyright © 2013–2016, Texas Instruments Incorporated

11.4.33 Host Interrupt Prioritized Index Register 1 (HIPIR1)

The host interrupt prioritized index register 1 (HIPIR1) shows the highest priority current pending interrupt for the FIQ interrupt. The HIPIR1 is shown in Figure 11-35 and described in Table 11-35.

Figure 11-35. Host Interrupt Prioritized Index Register 1 (HIPIR1)

31	30	16
NONE	Reserved	
R-1	R-0	

15	10	9	0
Reserved		PRI_INDX	
R-0		R-0	

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

Table 11-35. Host Interrupt Prioritized Index Register 1 (HIPIR1) Field Descriptions

Bit	Field	Value	Description
31	NONE	0-1	No Interrupt is pending.
30-10	Reserved	0	Reserved
9-0	PRI_INDX	0-3FFh	Interrupt number of the highest priority pending interrupt for FIQ host interrupt.

11.4.34 Host Interrupt Prioritized Index Register 2 (HIPIR2)

The host interrupt prioritized index register 2 (HIPIR2) shows the highest priority current pending interrupt for the IRQ interrupt. The HIPIR2 is shown in Figure 11-36 and described in Table 11-36.

Figure 11-36. Host Interrupt Prioritized Index Register 2 (HIPIR2)

31	30	16
NONE	Reserved	
R-1	R-0	

15	10	9	0
Reserved		PRI_INDX	
R-0		R-0	

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

Table 11-36. Host Interrupt Prioritized Index Register 2 (HIPIR2) Field Descriptions

Bit	Field	Value	Description
31	NONE	0-1	No Interrupt is pending.
30-10	Reserved	0	Reserved
9-0	PRI_INDX	0-3FFh	Interrupt number of the highest priority pending interrupt for IRQ host interrupt.

Copyright © 2013–2016, Texas Instruments Incorporated

11.4.35 Host Interrupt Nesting Level Register 1 (HINLR1)

The host interrupt nesting level register 1 (HINLR1) displays and controls the nesting level for FIQ host interrupt. The nesting level controls which channel and lower priority channels are nested. The HINLR1 is shown in Figure 11-37 and described in Table 11-37.

Figure 11-37. Host Interrupt Nesting Level Register 1 (HINLR1)

31	30	16
OVERRIDE	Reserved	
W-0	R-0	

15	9	8	0
Reserved		NEST_LVL	
R-0		R/W-100h	

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

Table 11-37. Host Interrupt Nesting Level Register 1 (HINLR1) Field Descriptions

Bit	Field	Value	Description
31	OVERRIDE	0-1	Reads return 0. Writes of a 1 override the auto updating of the NEST_LVL and use the write data.
30-9	Reserved	0	Reserved
8-0	NEST_LVL	0-1FFh	Reads return the current nesting level for the FIQ host interrupt. Writes set the nesting level for the FIQ host interrupt. In auto mode the value is updated internally, unless the OVERRIDE is set and then the write data is used.

11.4.36 Host Interrupt Nesting Level Register 2 (HINLR2)

The host interrupt nesting level register 2 (HINLR2) displays and controls the nesting level for IRQ host interrupt. The nesting level controls which channel and lower priority channels are nested. The HINLR2 is shown in Figure 11-38 and described in Table 11-38.

Figure 11-38. Host Interrupt Nesting Level Register 2 (HINLR2)

31	30	16
OVERRIDE	Reserved	
W-0	R-0	

15	9	8	0
Reserved		NEST_LVL	
R-0		R/W-100h	

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

Table 11-38. Host Interrupt Nesting Level Register 2 (HINLR2) Field Descriptions

Bit	Field	Value	Description
31	OVERRIDE	0-1	Reads return 0. Writes of a 1 override the auto updating of the NEST_LVL and use the write data.
30-9	Reserved	0	Reserved
8-0	NEST_LVL	0-1FFh	Reads return the current nesting level for the IRQ host interrupt. Writes set the nesting level for the IRQ host interrupt. In auto mode the value is updated internally, unless the OVERRIDE is set and then the write data is used.

Copyright © 2013–2016, Texas Instruments Incorporated

11.4.37 Host Interrupt Enable Register (HIER)

The host interrupt enable register (HIER) enables or disables individual host interrupts (FIQ and IRQ). These work separately from the global enables. There is one bit per host interrupt. These bits are updated when writing to the host interrupt enable indexed set register (HIEISR) and the host interrupt disable indexed clear register (HIDISR). The HIER is shown in Figure 11-39 and described in Table 11-39.

Figure 11-39. Host Interrupt Enable Register (HIER)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		IRQ	FIQ
R-0		R/W-0	R/W-0

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

Table 11-39. Host Interrupt Enable Register (HIER) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	IRQ		Enable of IRQ
		0	IRQ is disabled.
		1	IRQ is enabled.
0	FIQ		Enable of FIQ
		0	FIQ is disabled.
		1	FIQ is enabled.

Copyright © 2013–2016, Texas Instruments Incorporated

11.4.38 Host Interrupt Prioritized Vector Register 1 (HIPVR1)

The host interrupt prioritized vector register 1 (HIPVR1) shows the interrupt vector address of the highest priority interrupt pending for FIQ host interrupt. The HIPVR1 is shown in Figure 11-40 and described in Table 11-40.

Figure 11-40. Host Interrupt Prioritized Vector Register 1 (HIPVR1)

31	0
ADDR	
R-0	

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

Table 11-40. Host Interrupt Prioritized Vector Register 1 (HIPVR1) Field Descriptions

Bit	Field	Value	Description
31-0	ADDR	0-FFFF FFFFh	The currently highest priority interrupt vector address across for the FIQ host interrupt.

11.4.39 Host Interrupt Prioritized Vector Register 2 (HIPVR2)

The host interrupt prioritized vector register 2 (HIPVR2) shows the interrupt vector address of the highest priority interrupt pending for IRQ host interrupt. The HIPVR2 is shown in Figure 11-41 and described in Table 11-41.

Figure 11-41. Host Interrupt Prioritized Vector Register 2 (HIPVR2)

31	0
ADDR	
R-0	

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

Table 11-41. Host Interrupt Prioritized Vector Register 2 (HIPVR2) Field Descriptions

Bit	Field	Value	Description
31-0	ADDR	0-FFFF FFFFh	The currently highest priority interrupt vector address across for the IRQ host interrupt.

Copyright © 2013–2016, Texas Instruments Incorporated

Boot Considerations

Topic Page

12.1 Introduction 299

Copyright © 2013–2016, Texas Instruments Incorporated

12.1 Introduction

This device supports a variety of boot modes through an internal ARM ROM bootloader. This device does not support dedicated hardware boot modes; therefore, all boot modes utilize the internal ARM ROM. The input states of the BOOT pins are sampled and latched into the BOOTCFG register, which is part of the system configuration (SYSCFG) module, when device reset is deasserted. Boot mode selection is determined by the values of the BOOT pins.

The following boot modes are supported:

- NAND Flash boot
  - 8-bit NAND
  - 16-bit NAND
- NOR Flash boot
  - NOR Direct boot (8-bit or 16-bit)
  - NOR Legacy boot (8-bit or 16-bit)
  - NOR AIS boot (8-bit or 16-bit)
- HPI boot
- I2C0/I2C1 boot
  - EEPROM (Master Mode)
  - External Host (Slave Mode)
- SPI0/SPI1 boot
  - Serial Flash (Master Mode)
  - Serial EEPROM (Master Mode)
  - External Host (Slave Mode)
- UART0/1/2 boot
  - External Host
- MMC/SD0 boot

See *Using the AM18xx Bootloader Application Report* (SPRABA5) for more details on the ROM Boot Loader, a list of boot pins used, and the complete list of supported boot modes.

Copyright © 2013–2016, Texas Instruments Incorporated

Programmable Real-Time Unit Subsystem (PRUSS)

Topic **Page**

13.1 Overview 301
13.2 Description 303
13.3 Constants Table 304
13.4 PRU Module Interface 305
13.5 Instruction Set 306
13.6 Instruction Formats 309
13.7 PRU Interrupt Controller 327
13.8 Registers 334

Copyright © 2013–2016, Texas Instruments Incorporated

The Programmable Real-Time Unit Subsystem (PRUSS) consists of:

- Two programmable real-time units (PRU0 and PRU1) and their associated memories.
- An interrupt controller (INTC) for handling system interrupt events. The INTC also supports posting events back to the device level host CPU.
- A Switched Central Resource (SCR) for connecting the various internal and external masters to the resources inside the PRUSS.

The two PRUs can operate completely independently or in coordination with each other. The two PRUs can also work in coordination with the device level host CPU. This is determined by the nature of the program that is loaded into the two PRUs instruction memory. Several different signaling mechanisms are available between the two PRUs and the device level host CPU.

The two PRUs are optimized for performing embedded tasks that require manipulation of packed memory-mapped data structures, handling of system events that have tight real-time constraints and interfacing with systems external to the device.

13.1 Overview

The PRU is a optimized for performing embedded tasks that require manipulation of packed memory mapped data structures, handling of system events that have tight realtime constraints and interfacing with systems external to the DSP/SoC. The PRU is both very small and very efficient at handling such tasks.

The major attributes of the PRU are as follows:

Attribute	Value
IO Architecture	Load / Store
Data Flow Architecture	Register to Register
Core Level Bus Architecture	
Type	4-Bus Harvard (1 Instruction, 3 Data)
Instruction I/F	32-Bit
Memory I/F 0	32-Bit
Memory I/F 1	32-Bit
Execution Model	
Issue Type	Scalar
Pipelining	None
Ordering	In-order
ALU Type	Unsigned Integer
Registers	
General Purpose (GP)	30 (R1 – R30)
External Status	1 (R31)
GP / Indexing	1 (R0)
Addressability	Bit, Byte (8-bit), Halfword (16-bit), Word (32-bit), Pointer

Copyright © 2013–2016, Texas Instruments Incorporated

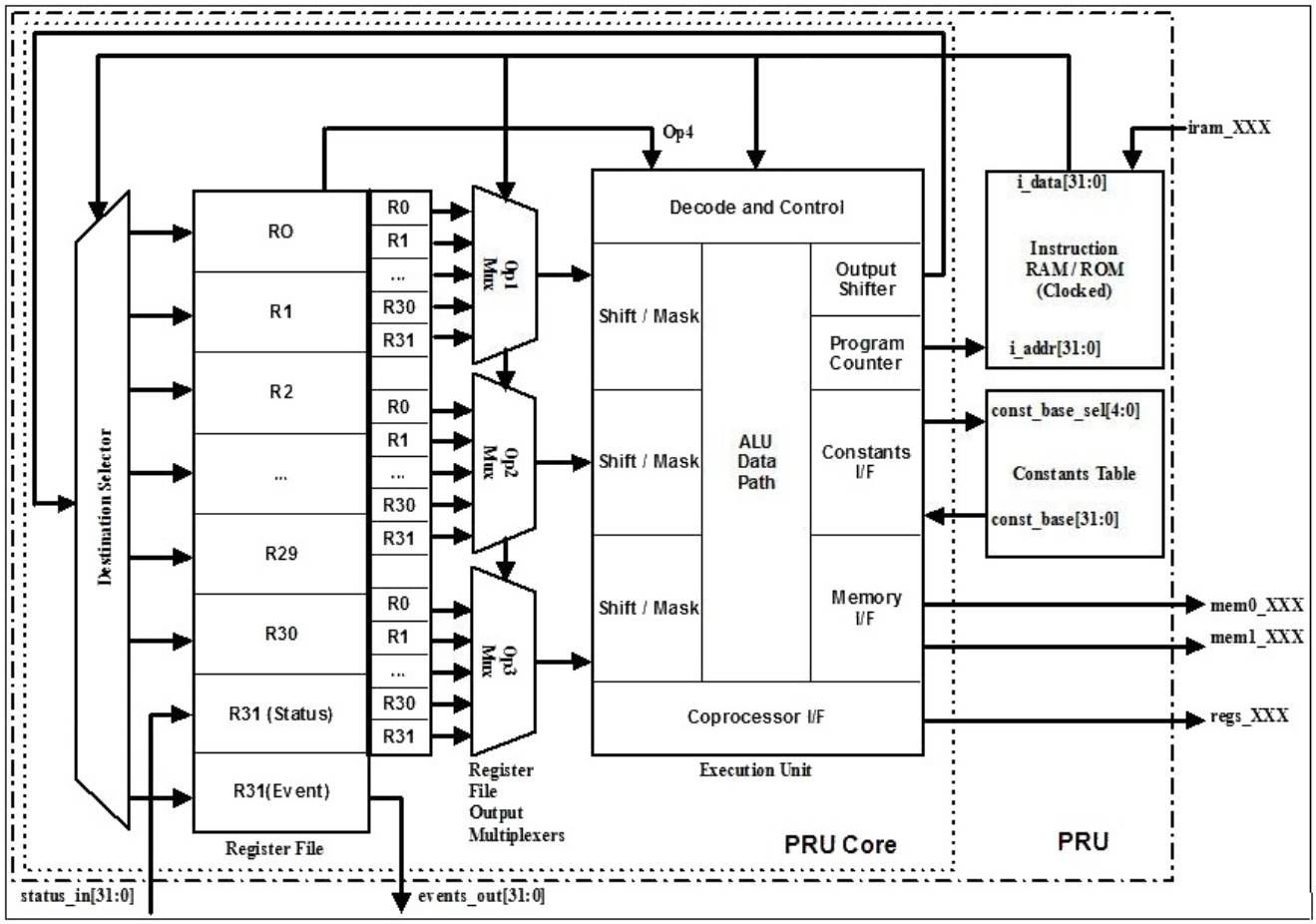
Attribute	Value
Addressing Modes	
Load Immediate	16-bit Immediate
Load / Store – Memory	Register Base + Register Offset
	Register Base + 8-bit Immediate Offset
	Register Base with auto increment / decrement
	Constant Table Base + Register Offset
	Constant Table Base + 8-bit Immediate Offset
	Constant Table Base with auto increment / decrement
Data Path Width	32-Bits
Instruction Width	32-Bits
Accessibility to Internal PRU Structures	Provides 32-bit Slave with 3 regions: • Instruction RAM • Control / Status registers • Debug access to internal registers (R0-R31) and constant table

Copyright © 2013–2016, Texas Instruments Incorporated

13.2 Description

The processor is based on a four bus architecture which allows instructions to be fetched and executed concurrently with data transfers. Additionally, an input is provided in order to allow external status information to be reflected in the internal processor status register. The figure below shows a block diagram of the processing element and the associated instruction RAM/ROM that contains the code that is to be executed.

Figure 13-1. PRU Block Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

13.3 Constants Table

The PRU Constants Table is a structure connected to a dedicated interface on the PRU core within the PRU that is used to provide the base address for the Load Burst Constant + Offset (LBCO) and Store Burst Constant + Offset (SBCO) instructions. The PRU Constants Table is provided in order to maximize the usage of the PRU register file for embedded processing applications by moving many of the commonly used constant or deterministically calculated base addresses from the internal register file to an external table. Since this table is accessed using a dedicated interface, no performance difference is realized between the LBCO and LBBO or SBCO and SBBO instructions. The constants in the table are provided in Table 2.

Table 13-1. Constants Table(1)

Entry #	Region Pointed To	Value [31:0]
0	PRU0/1 Local INTC	0x00004000
1	Timer64P0	0x01C20000
2	I2C0	0x01C22000
3	PRU0/1 Local Data	0x00000000
4	PRU1/0 Local Data	0x00002000
5	MMC/SD	0x01C40000
6	SPI0	0x01C41000
7	UART0	0x01C42000
8	McASP0 DMA	0x01D02000
9	Reserved	0x01D06000
10	Reserved	0x01D0A000
11	UART1	0x01D0C000
12	UART2	0x01D0D000
13	USB0	0x01E00000
14	USB1	0x01E25000
15	UHPI Config	0x01E10000
16	Reserved	0x01E12000
17	I2C1	0x01E28000
18	EPWM0	0x01F00000
19	EPWM1	0x01F02000
20	Reserved	0x01F04000
21	ECAP0	0x01F06000
22	ECAP1	0x01F07000
23	ECAP2	0x01F08000
24	PRU0/1 Local Data	0x00000n00, n = c24_blk_index[3:0]
25	McASP0 Control	0x01D00n00, n = c25_blk_index[3:0]
26	Reserved	0x01D04000
27	Reserved	0x01D08000
28	DSP Megamodule RAM/ROM	0x11nnnn00, nnnn = c28_pointer[15:0]
29	EMIFA SDRAM	0x40nnnn00, nnnn = c29_pointer[15:0]
30	Shared RAM	0x80nnnn00, nnnn = c30_pointer[15:0]
31	mDDR/DDR2 Data	0xC0nnnn00, nnnn = c31_pointer[15:0]

(1) These constants cannot be used due to memory map restrictions.

1. Constants not in this table can be created 'on the fly' by two consecutive LDI #16 instructions. These constants are just ones that are expected to be commonly used, enough so to be hard-coded into the PRU constants table.
2. Constants table entries 24 through 31 are not fully hard-coded, but contain a programmable bitfield (ex. c24_blk_index[3:0]) that is programmable through the PRU control register space (0x01C3_7000 - 0x01C3_73FF for PRU0 and 0x01C3_7800 - 0x01C3_7BFF for PRU1).

Copyright © 2013–2016, Texas Instruments Incorporated

13.4 PRU Module Interface

13.4.1 Event Out Mapping (R31): PRU System Events

PRU Event Interface directly feeds pulsed event information out of the PRU's internal ALU. This provides the interface logic between the PRU R31 event_out[5:0] and the system interrupts for the PRUSS Interrupt controller(INTC).

Bits	Name	Description
31-6	Reserved	
5	pruX_vec_valid	Valid strobe for vector output
4-0	pruX_vec[4:0]	Vector output

Writing a '1' to pruX_vec_valid (R31 bit 5) simultaneously with a channel number from 0 to 31 written to pruX_vec[4:0] (R31 bits 4:0) creates a pulse on the output of the appropriate de-multiplexer output. For example, writing '100000' will generate a pulse on demux channel 0, writing '100001' will generate a pulse on demux channel 1, ... writing '111111' will generate a pulse on demux channel 31, and writing '0xxxxx' will not generate any pulse on the demux output. The demultiplexed values from both of the PRUs are logically ORed together. The composite demultiplexed output channels 0 through 31 are connected to system interrupts 32 through 63 respectively.

This allows the PRU to assert one of the systems interrupts 32-63 by writing to its own R31 register. The system interrupt is used to either post a completion event to one of the host CPUs (ARM, DSP) or to signal the other PRU. The host to be signaled is determined by the system interrupt to interrupt channel mapping (programmable). Refer to Section 13.7 for more details.

13.4.2 Status Mapping (R31): Interrupt Events Input

The PRU Real Time Status Interface directly feeds information into register 31(R31) of the PRU's internal register file. The firmware on the PRU uses the status information to make decisions during execution. The status interface is comprised of signals from different modules inside of the PRUSS which require some level of interaction with the PRU. More details on the Host interrupts imported into bit 30 and 31 of register R31 of both the PRUs is provided in the chapter 3, PRUSS Interrupt Controller (INTC).

Bits	Name	Description
31	pru_intr_in[1]	PRU Interrupt 1 from INTC
30	pru_intr_in[0]	PRU Interrupt 0 from INTC
29-0	pruX_r31_status[29:0]	Status inputs from primary input

13.4.3 General Purpose Inputs (R31)

The pruX_r31_status[29:0] are mapped out of the PRUSS and are brought out as general purpose input pins. The values input to the pins "pruX_R31[29:0]" are reflected in the R31 register on bits [29:0] to be used by the program running on the PRU. Each PRU of the PRUSS has a separate mapping to pins, so that there are 60 total general purpose inputs to the PRUSS.

13.4.4 General Purpose Outputs (R30)

The pruX_r30[31:0] bits are exported out of the PRUSS and are brought out as general purpose output pins. The values written to register R30 will be reflected on the general purpose output pins "pruX_R30[31:0]" to be used by the program running on the PRU. Each PRU of the PRUSS has a separate mapping to pins, so that there are 64 total general purpose outputs from the PRUSS.

Copyright © 2013–2016, Texas Instruments Incorporated

13.5 Instruction Set

The instruction set is divided into four major categories:

1. Instructions which move data in or out of the processors internal registers
2. Instructions which perform an arithmetic operation
3. Instructions which perform a logical operation
4. Instructions which control program flow

The following sections give a complete list and short description of all of the supported instructions. In these descriptions, the following abbreviations are used:

Table 13-2. Abbreviations for Instruction Descriptions

Abbreviation	Description
Rs1	Source register 1 from the instruction
Op2	Operand 2 from the instruction – can be either a register (Rs2) or an 8-bit immediate value
Rd	Destination register from the instruction
BrOff	Branch offset from the instruction – a 10-bit 2's complement relative offset
WdCnt	Word count – the # of 32-bit data phases that occur in the burst on an external memory interface
CPI	Clock Cycles Per Instruction

Table 13-3. Load/Store Instructions

Mnemonic	Instruction	Description	CPI
LDI	Load Immediate	Load 16-bit immediate value into internal register	1
LBBO	Load Burst, Base + Offset	Load variable length burst of bytes through one of the memory interfaces into internal register(s) using a register as the base address and a register or an 8-bit immediate as the offset	1 + WdCnt (VBUS) 2 + WdCnt (VBUSP)
SBBO	Store Burst, Base + Offset	Store variable length burst of bytes through one of the memory interfaces from internal register(s) using a register as the base address and a register or 8-bit immediate as the offset	1 + WdCnt
LBCO	Load Burst, Constant + Offset	Load variable length burst of bytes through one of the memory interfaces into internal register(s) using an indexed constant as the base address and a register or an 8-bit immediate as the offset	1 + WdCnt (VBUS) 2 + WdCnt (VBUSP)
SBCO	Store Burst, Constant + Offset	Store variable length burst of bytes through one of the memory interfaces from internal register(s) using an indexed constant as the base address and a register or 8-bit immediate as the offset	1 + WdCnt

Table 13-4. Arithmetic Instructions

Mnemonic	Instruction	Description	CPI
ADD	Integer Add	Adds Rs1 and Op2, writes result to Rd, and saves carry.	1
ADC	Integer Add With Carry	Adds Rs1, Op2 and the saved carry, writes result to Rd, and saves carry.	1
SUB	Integer Subtract	Subtracts Op2 from Rs1 and writes result to Rd and saves carry (borrow).	1
SUC	Integer Subtract With Carry	Subtracts Op2 from Rs1 then subtracts the saved carry (borrow) and writes result to Rd and saves carry (borrow).	1
RSB	Integer Reverse Subtract	Subtracts Rs1 from Op2 and writes result to Rd and saves carry (borrow).	1
RSC	Interger Reverse Subtract With Carry	Subtracts Rs1 from Op2 then subtracts the saved carry (borrow) and writes result to Rd and saves carry (borrow).	1

Copyright © 2013–2016, Texas Instruments Incorporated

Table 13-5. Logical Instructions

Mnemonic	Instruction	Description	CPI
AND	Bitwise And	Bitwise ANDs Rs1 with Op2 and stores to Rd.	1
OR	Bitwise Or	Bitwise ORs Rs1 with Op2 and stores to Rd.	1
XOR	Bitwise Exclusive Or	Bitwise exclusive ORs Rs1 with Op2 and stores to Rd.	1
NOT	Bitwise Invert	Bitwise inverts Rs1 and stores to Rd.	1
LSR	Logical Shift Right	Shifts Rs1 right (with zero fill) by the value given in the 5 LSBs of Op2 and stores to Rd.	1
LSL	Logical Shift Left	Shifts Rs1 left (with zero fill) by the value given in the 5 LSBs of Op2 and stores to Rd.	1
MIN	Minimum	Compares Rs1 and Op2 and the smaller value is copied to Rd.	1
MAX	Maximum	Compares Rs1 and Op2 and the larger value is copied to Rd.	1
CLR	Clear Bit	Copies Rs1 to Rd but with a bit specied by the 5 LSBs of Op2 cleared during the copy.	1
SET	Set Bit	Copies Rs1 to Rd but with a bit specied by the 5 LSBs of Op2 set during the copy.	1
LMBD	Left-most Bit Detect	Scans Rs1 from the leftmost bit for a bit equal to bit 0 of Rs2. When found, the bit number (0 to 31) is written to Rd. If not found, the value 32 is written to Rd.	1
SCAN	Scan Register File	Scans the register file for a byte pattern of a programmable length (up to 4 bytes) with a programmable field count and field stride. The Op1 register contains 4 fields: Rn.b0 = offset in the register file from R0.b0 to start scanning Rn.b1 = fc (field count), the number of fields to scan Rn.b2 = fw (field width), the size in bytes of the field to scan for (1, 2, or 4 bytes) Rn.b3 = fs (field stride), the number of bytes to advance to the next field in the register file (1 to 4 bytes) Op1 is updated after the scan with the offset of the match (or 0xFF if no match) in Rn.b0, and the fields remaining in the scan (including the matching field) in Rn.b1. The Op2 is the field to scan for.	IF fw = fs 2+((fc*fw+3)/4) ELSE 2+fc This is a worst case cycle count. Matching scans could take fewer cycles.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 13-6. Program Flow Control Instructions

Mnemonic	Instruction	Description	CPI
QBBS	Quick Branch – Bit Set	Adds BrOff to the program counter if the bit in Rs1 specified by the 5 LSBs of Op2 is a 1	1
QBBC	Quick Branch – Bit Clear	Adds BrOff to the program counter if the bit in Rs1 specified by the 5 LSBs of Op2 is a 0	1
QBGT	Quick Branch – Greater Than	Compares Op2 to Rs1 and adds BrOff to the program counter if Op2 is greater than Rs1.	1
QBGE	Quick Branch – Greater Than or Equal	Compares Op2 to Rs1 and adds BrOff to the program counter if Op2 is greater than or equal to Rs1.	1
QBLT	Quick Branch – Less Than	Compares Op2 to Rs1 and adds BrOff to the program counter if Op2 is less than Rs1.	1
QBLE	Quick Branch – Less Than or Equal	Compares Op2 to Rs1 and adds BrOff to the program counter if Op2 is less than or equal to Rs1.	1
QBEQ	Quick Branch – Equal	Compares Op2 to Rs1 and adds BrOff to the program counter if Op2 is equal to Rs1.	1
QBNE	Quick Branch – Not Equal	Compares Op2 to Rs1 and adds BrOff to the program counter if Op2 is not equal to Rs1.	1
JMP	Unconditional Jump	Sets the program counter equal to either the contents of a register or to a 16-bit immediate value	1
JAL	Unconditional Jump and Link	Saves the current program counter into Rd and sets the program counter equal to either the contents of a register or to a 16-bit immediate value.	1
HALT	Halt Processor	Disables the PRU and does not increment the program counter. When the PRU is re-enabled, it will continue processing at this instruction.	1
SLP	Sleep	Pauses execution of the current program and disables the clock for the majority of the core until a specified external event (un-masked status bit) is asserted.	1 to infinity

Copyright © 2013–2016, Texas Instruments Incorporated

13.6 Instruction Formats

A total of 7 different instruction formats are supported for the various operations. The following sections describe the position, size, and function of each of the fields within the various formats.

Figure 13-2. Format 1a: (All Arithmetic and Logical Functions – Register Op2)

31	29	28	24	23	21	20	16
OP		ALUOP	IO	Rs2Sel		Rs2	
15	13	12	8	7	5	4	0
Rs1Sel		Rs1		RdSel		Rd	

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

Table 13-7. Format 1a: (All Arithmetic and Logical Functions – Register Op2)

Bit	Field	Description
31-29	OP	0b000 = Specifies Format 1
28-25	ALUOP	0 = ADD 1 = ADC 2 = SUB 3 = SUC 4 = LSL 5 = LSR 6 = RSB 7 = RSC 8 = AND 9 = OR 10 = XOR 11 = NOT 12 = MIN 13 = MAX 14 = CLR 15 = SET
24	IO	0 = Op2 is a register
23-21	Rs2Sel	0 = Select bits 7:0 from the source register 2 1 = Select bits 15:8 from the source register 2 2 = Select bits 23:16 from the source register 2 3 = Select bits 31:24 from the source register 2 4 = Select bits 15:0 from the source register 2 5 = Select bits 23:8 from the source register 2 6 = Select bits 31:16 from the source register 2 7 = Select bits 31:0 from the source register 2
20-16	Rs2	0 – 31 = This field selects the register number which contains the second source operand
15-13	Rs1Sel	0 = Select bits 7:0 from the source register 1 1 = Select bits 15:8 from the source register 1 2 = Select bits 23:16 from the source register 1 3 = Select bits 31:24 from the source register 1 4 = Select bits 15:0 from the source register 1 5 = Select bits 23:8 from the source register 1 6 = Select bits 31:16 from the source register 1 7 = Select bits 31:0 from the source register 1
12-8	Rs1	0-31 = This field selects the register number which contains the first source operand

Copyright © 2013–2016, Texas Instruments Incorporated

Table 13-7. Format 1a: (All Arithmetic and Logical Functions – Register Op2) (continued)

Bit	Field	Description
7-5	RdSel	0 = Select bits 7:0 of the destination register 1 = Select bits 15:8 of the destination register 2 = Select bits 23:16 of the destination register 3 = Select bits 31:24 of the destination register 4 = Select bits 15:0 of the destination register 5 = Select bits 23:8 of the destination register 6 = Select bits 31:16 of the destination register 7 = Select bits 31:0 of the destination register
4-0	Rd	0-31 = This field selects the destination register number to which the result should be written.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 13-3. Format 1b: (All Arithmetic and Logical Functions – Immediate Op2)

31	29	28	24	23	16
OP		ALUOP	IO	Imm2	
15	13	12	8	7 5	4 0
Rs1Sel		Rs1		RdSel	Rd

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

Table 13-8. Format 1b: (All Arithmetic and Logical Functions – Immediate Op2)

Bit	Field	Description
31-29	OP	0b000 = Specifies Format 1
28-25	ALUOP	0 = ADD 1 = ADC 2 = SUB 3 = SUC 4 = LSL 5 = LSR 6 = RSB 7 = RSC 8 = AND 9 = OR 10 = XOR 11 = NOT 12 = MIN 13 = MAX 14 = CLR 15 = SET
24	IO	1 = Op2 is an 8-bit immediate
23-16	Imm2	0-255 = This field is the 8-bit immediate value to be used as the source operand 2.
15-13	Rs1Sel	0 = Select bits 7:0 from the source register 1 1 = Select bits 15:8 from the source register 1 2 = Select bits 23:16 from the source register 1 3 = Select bits 31:24 from the source register 1 4 = Select bits 15:0 from the source register 1 5 = Select bits 23:8 from the source register 1 6 = Select bits 31:16 from the source register 1 7 = Select bits 31:0 from the source register 1
12-8	Rs1	0-31 = This field selects the register number which contains the first source operand
7-5	RdSel	0 = Select bits 7:0 of the destination register 1 = Select bits 15:8 of the destination register 2 = Select bits 23:16 of the destination register 3 = Select bits 31:24 of the destination register 4 = Select bits 15:0 of the destination register 5 = Select bits 23:8 of the destination register 6 = Select bits 31:16 of the destination register 7 = Select bits 31:0 of the destination register
4-0	Rd	0-31 = This field selects the destination register number to which the result should be written.

Copyright © 2013–2016, Texas Instruments Incorporated

The following represents the 7 most significant bits of all format 2 instructions:

Figure 13-4. Format 2

31 29	28 25	24 0
OP	SUBOB	RESERVED

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

Table 13-9. Format 2

Bit	Field	Description
31-29	OP	0b001 = Specifies Format 2
28-25	SUBOB	0 = JMP 1 = JAL 2 = LDI 3 = LMBD 4 = SCAN 5 = HALT 6 = currently reserved for MVIx 7-13 = RESERVED 14 = currently reserved for RFI – Return From Interrupt 15 = SLP – Sleep

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 13-5. Format 2a: (JMP,JAL – Register Op2)

31	29	28	25	24	23	21	20	16
OP		SUBOP		IO	Rs2Sel		Rs2	
15			8	7		5	4	0
RESERVED					RdSel		Rd	

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

Table 13-10. Format 2a: (JMP,JAL – Register Op2)

Bit	Field	Description
31-29	OP	0b001 = Specifies Format 2
28-25	SUBOP	0 = JMP 1 = JAL
24	IO	0 = Op2 is a register
23-21	Rs2Sel	0 = Select bits 7:0 from the source register 2 1 = Select bits 15:8 from the source register 2 2 = Select bits 23:16 from the source register 2 3 = Select bits 31:24 from the source register 2 4 = Select bits 15:0 from the source register 2 5 = Select bits 23:8 from the source register 2 6 = Select bits 31:16 from the source register 2 7 = Select bits 31:0 from the source register 2
20-16	Rs2	0-31 = This field selects the register number which contains the value to be copied to the program counter.
15-8	RESERVED	
7-5	RdSel	0 = Select bits 7:0 of the destination register 1 = Select bits 15:8 of the destination register 2 = Select bits 23:16 of the destination register 3 = Select bits 31:24 of the destination register 4 = Select bits 15:0 of the destination register 5 = Select bits 23:8 of the destination register 6 = Select bits 31:16 of the destination register 7 = Select bits 31:0 of the destination register
4-0	Rd	0-30 = This field selects the destination register number to which the pre-incremented program counter should be written. Only written for JAL.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 13-6. Format 2b: (JMP, JAL – Immediate Op2)

31	29	28	25	24	23	16
OP		SUBOP		IO	Imm	
15			9	8	7 5	4 0
Imm					RdSel	Rd

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

Table 13-11. Format 2b: (JMP, JAL – Immediate Op2)

Bit	Field	Description
31-29	OP	0b001 = Specifies Format 2
28-25	SUBOP	0 = JMP - Jump 1 = JAL – Jump and Link
24	IO	1 = Jump operand is a 16-bit immediate value
23-8	Imm	0h–FFFFh = This field is the 16-bit immediate value to be copied to the program counter.
7-5	RdSel	0 = Select bits 7:0 of the destination register 1 = Select bits 15:8 of the destination register 2 = Select bits 23:16 of the destination register 3 = Select bits 31:24 of the destination register 4 = Select bits 15:0 of the destination register 5 = Select bits 23:8 of the destination register 6 = Select bits 31:16 of the destination register 7 = Select bits 31:0 of the destination register
4-0	Rd	0-30 = This field selects the destination register number to which the pre-incremented program counter should be written. Only written for JAL.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 13-7. Format 2c: (LDI)

31	29	28	25	24	23	16
OP		SUBOP		IO	Imm	
15			8	7	5	4 0
Imm				RdSel	Rd	

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

Table 13-12. Format 2c: (LDI)

Bit	Field	Description
31-29	OP	0b001 = Specifies Format 2
28-25	SUBOP	2 = Specifies LDI
24	RESERVED	
23-8	Imm	0 = Select bits 7:0 of the destination register 1 = Select bits 15:8 of the destination register 2 = Select bits 23:16 of the destination register 3 = Select bits 31:24 of the destination register 4 = Select bits 15:0 of the destination register 5 = Select bits 23:8 of the destination register 6 = Select bits 31:16 of the destination register 7 = Select bits 31:0 of the destination register
7-5	RdSel	0 = Select bits 7:0 of the destination register 1 = Select bits 15:8 of the destination register 2 = Select bits 23:16 of the destination register 3 = Select bits 31:24 of the destination register 4 = Select bits 15:0 of the destination register 5 = Select bits 23:8 of the destination register 6 = Select bits 31:16 of the destination register 7 = Select bits 31:0 of the destination register
4-0	Rd	0-31 = This field selects the destination register number to which the result should be written.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 13-8. Format 2d: (LMBD - Leftmost Bit Detect - Register Op2)

31	29	28	25	24	23	21	20	16
OP		SUBOB		IO	Rs2Sel		Rs2	
15	13	12	8	7	5	4	0	
Rs1Sel		Rs1		RdSel		Rd		

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

Table 13-13. Format 2d: (LMBD - Leftmost Bit Detect - Register Op2)

Bit	Field	Description
31-29	OP	0b001 = Specifies Format 2
28-25	SUBOB	3 = Specifies LMBD
24	IO	0 = Op2 is a register
23-21	Rs2Sel	0 = Select bits 7:0 from Rs2 1 = Select bits 15:8 from Rs2 2 = Select bits 23:16 from Rs2 3 = Select bits 31:24 from Rs2 4 = Select bits 15:0 from Rs2 5 = Select bits 23:8 from Rs2 6 = Select bits 31:16 from Rs2 7 = Select bits 31:0 from Rs2
20-16	Rs2	0-31 = Rs2 register number 0-31
15-13	Rs1Sel	0 = Select bits 7:0 from Rs1 1 = Select bits 15:8 from Rs1 2 = Select bits 23:16 from Rs1 3 = Select bits 31:24 from Rs1 4 = Select bits 15:0 from Rs1 5 = Select bits 23:8 from Rs1 6 = Select bits 31:16 from Rs1 7 = Select bits 31:0 from Rs1
12-8	Rs1	0-31 = Rs1 register number 0-31
7-5	RdSel	0 = Select bits 7:0 from Rd 1 = Select bits 15:8 from Rd 2 = Select bits 23:16 from Rd 3 = Select bits 31:24 from Rd 4 = Select bits 15:0 from Rd 5 = Select bits 23:8 from Rd 6 = Select bits 31:16 from Rd 7 = Select bits 31:0 from Rd
4-0	Rd	0-31 = Rd register number 0-31

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 13-9. Format 2e: (LMBD - Immediate Op2)

31	29	28	25	24	23	16
OP		SUBOB		IO	Imm2	
15	13	12	8	7	5	4 0
Rs2Sel		Rs2		RdSel		Rd

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

Table 13-14. Format 2e: (LMBD - Immediate Op2)

Bit	Field	Description
31-29	OP	0b001 = Specifies Format 2
28-25	SUBOB	3 = Specifies LMBD
24	IO	1 = Op2 is an 8 bit immediate
23-16	Imm2	0-255 = Immediate for src2
15-13	Rs2Sel	0 = Select bits 7:0 from Rs1 1 = Select bits 15:8 from Rs1 2 = Select bits 23:16 from Rs1 3 = Select bits 31:24 from Rs1 4 = Select bits 15:0 from Rs1 5 = Select bits 23:8 from Rs1 6 = Select bits 31:16 from Rs1 7 = Select bits 31:0 from Rs1
12-8	Rs1	0-31 = Rd register number
7-5	RdSel	0 S= elect bits 7:0 from Rd 1 = Select bits 15:8 from Rd 2 = Select bits 23:16 from Rd 3 = Select bits 31:24 from Rd 4 = Select bits 15:0 from Rd 5 = Select bits 23:8 from Rd 6 = Select bits 31:16 from Rd 7 = Select bits 31:0 from Rd
4-0	Rd	0-31 = Rd register number 0-31

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 13-10. Format 2f: (SCAN - Register Op2)

31	29	28	25	24	23	21	20	16
OP		SUBOB		IO	Rs2Sel		Rs2	
15	13	12	8	7	5	4	0	
Rs1Sel		Rs1		RdSel		Rd		

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

Table 13-15. Format 2f: (SCAN - Register Op2)

Bit	Field	Description
31-29	OP	0b001 = Specifies Format 2
28-25	SUBOB	4 = Specifies SCAN
24	IO	0 = Op2 is a register
23-21	Rs2Sel	0 = Select bits 7:0 from Rs2 1 = Select bits 15:8 from Rs2 2 = Select bits 23:16 from Rs2 3 = Select bits 31:24 from Rs2 4 = Select bits 15:0 from Rs2 5 = Select bits 23:8 from Rs2 6 = Select bits 31:16 from Rs2 7 = Select bits 31:0 from Rs2
20-16	Rs2	0-31 = Rs2 register number 0-31
15-13	Rs1Sel	7 = Select bits 31:0 from Rs1
12-8	Rs1	*0-31 = Must be identical to Rd*
7-5	RdSel	7 = Select bits 31:0 from Rd
4-0	Rd	0-31 = Rd register number 0-31

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 13-11. Format 2g: (SCAN - Immediate Op2)

31	29	28	25	24	23	16
OP		SUBOB		IO	Imm2	
15	13	12	8	7	5	4 0
Rs1Sel		Rs1		RdSel		Rd

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

Table 13-16. Format 2g: (SCAN - Immediate Op2)

Bit	Field	Description
31-29	OP	0b001 = Specifies Format 2
28-25	SUBOB	4 = Specifies SCAN
24	IO	1 = Op2 is an 8 bit immediate
23-16	Imm2	0-255 = Immediate for src2
15-13	Rs1Sel	7 = Select bits 31:0 from Rs1
12-8	Rs1	*0-31 = Must be identical to Rd*
7-5	RdSel	7 = Select bits 31:0 from Rd
4-0	Rd	0-31 = Rd register number 0-31

Figure 13-12. Format 2h: (HALT)

31	29	28	25	24	0
OP		SUBOP		RESERVED	

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

Table 13-17. Format 2h: (HALT)

Bit	Field	Description
31-29	OP	0b001 = Specifies Format 2
28-25	SUBOB	5 = Specifies HALT
24-0	RESERVED	

Figure 13-13. Format 2i: (SLP)

31	29	28	25	24	23	22	16
OP		SUBOP		RESERVED	Wake OnStatus	RESERVED	
15							0
RESERVED							

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

Table 13-18. Format 2i: (SLP)

Bit	Field	Description
31-29	OP	0b001 = Specifies Format 2
28-25	SUBOP	15 = SLP – Sleep
24	RESERVED	
23	WakeOnStatus	0 = Do not wake when non-masked status is asserted 1 = Wake when non-masked status is asserted
22-0	RESERVED	

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 13-14. Format 4a: (Quick Arithmetic Test and Branch – Register Op2)

31	30	29	28	27	26	25	24	23	21	20	16
OP		GT	EQ	LT	BrOff[9:8]		IO	Rs2Sel		Rs2	
15		13	12				8	7			0
Rs1Sel			Rs1					BrOff[7:0]			

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

Table 13-19. Format 4a: (Quick Arithmetic Test and Branch – Register Op2)

Bit	Field	Description
31-30	OP	0b01 = Specifies Format 4
29	GT	0-1 = Greater Than: If set specifies that branch should be taken if Op2 > Rs1[(1)]
28	EQ	0-1 = Equal: If set specifies that branch should be taken if Op2 == Rs1[(1)]
27	LT	0-1 = Less Than: If set specifies that branch should be taken if Op2 < Rs1[(1)]
26-25	BrOff[9:8]	0-3 = This field contains the 2-MSBs of the 2s complement signed offset for the branch
24	IO	0 = Op2 is a register
23-21	Rs2Sel	0 = Select bits 7:0 from the source register 2 1 = Select bits 15:8 from the source register 2 2 = Select bits 23:16 from the source register 2 3 = Select bits 31:24 from the source register 2 4 = Select bits 15:0 from the source register 2 5 = Select bits 23:8 from the source register 2 6 = Select bits 31:16 from the source register 2 7 = Select bits 31:0 from the source register 2
20-16	Rs2	0-31 = This field selects the register number which contains the second source operand that is to be compared
15-13	Rs1Sel	0 = Select bits 7:0 from the source register 1 1 = Select bits 15:8 from the source register 1 2 = Select bits 23:16 from the source register 1 3 = Select bits 31:24 from the source register 1 4 = Select bits 15:0 from the source register 1 5 = Select bits 23:8 from the source register 1 6 = Select bits 31:16 from the source register 1 7 = Select bits 31:0 from the source register 1
12-8	Rs1	0-31 = This field selects the register number which contains the first source operand that is to be compared.
7-0	BrOff[7:0]	0-255 = This field contains the 8-LSBs of the 2-s complement signed offset for the branch

(1) Branch is taken if any of the 3 conditions are satisfied: GT, EQ or LT.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 13-15. Format 4b: (Quick Arithmetic Test and Branch – Immediate Op2)

31	30	29	28	27	26	25	24	23	16
OP		GT	EQ	LT	BrOff[9:8]		IO	Imm	
15		13	12				8	7	0
Rs1Sel			Rs1					BrOff[7:0]	

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

Table 13-20. Format 4b: (Quick Arithmetic Test and Branch – Immediate Op2)

Bit	Field	Description
31-30	OP	0b01 = Specifies Format 4
29	GT	0-1 = Greater Than: If set specifies that branch should be taken if Op2 > Rs1 (1)
28	EQ	0-1 = Equal: If set specifies that branch should be taken if Op2 == Rs1 (1)
27	LT	0-1 = Less Than: If set specifies that branch should be taken if Op2 < Rs1 (1)
26-25	BrOff[9:8]	0-3 = This field contains the 2-MSBs of the 2s complement signed offset for the branch
24	IO	1 = Op2 is an 8-bit immediate value
23-16	Imm	0-255 = 8-bit immediate value to be used as second operand to be compared.
15-13	Rs1Sel	0 = Select bits 7:0 from the source register 1 1 = Select bits 15:8 from the source register 1 2 = Select bits 23:16 from the source register 1 3 = Select bits 31:24 from the source register 1 4 = Select bits 15:0 from the source register 1 5 = Select bits 23:8 from the source register 1 6 = Select bits 31:16 from the source register 1 7 = Select bits 31:0 from the source register 1
12-8	Rs1	0-31 = This field selects the register number which contains the first source operand that is to be compared.
7-0	BrOff[7:0]	0-255 = This field contains the 8-LSBs of the 2-s complement signed offset for the branch

(1) Branch is taken if any of the 3 conditions are satisfied: GT, EQ or LT.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 13-16. Format 5a: (Quick Bit Test and Branch – Register Op2)

31	29	28	27	26	25	24	23	21	20	16
OP		BS	BC	BrOff[[9:8]		IO	Rs2Sel		Rs2	
15	13	12				8	7			0
Rs1Sel		Rs1					BrOff[7:0]			

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

Table 13-21. Format 5a: (Quick Bit Test and Branch – Register Op2)

Bit	Field	Description
31-29	OP	0b110 = Specifies Format 5
28	BS	0-1 = Bit Set: If set specifies that branch should be taken if Rs1[Op2[4:0]] == 1[(1)]
27	BC	0-1 = Bit Clear: If set specifies that branch should be taken if Rs1[Op2[4:0]] == 0[(1)]
26-25	BrOff[9:8]	0-3 = This field contains the 2-MSBs of the 2s complement signed offset for the branch
24	IO	0 = Op2 is a register
23-21	Rs2Sel	0 = Select bits 7:0 from the source register 2 1 = Select bits 15:8 from the source register 2 2 = Select bits 23:16 from the source register 2 3 = Select bits 31:24 from the source register 2 4 = Select bits 15:0 from the source register 2 5 = Select bits 23:8 from the source register 2 6 = Select bits 31:16 from the source register 2 7 = Select bits 31:0 from the source register 2
20-16	Rs2	0-31 = This field selects the register number which contains the bit number of the first operand which is to be compared.
15-13	Rs1Sel	0 = Select bits 7:0 from the source register 1 1 = Select bits 15:8 from the source register 1 2 = Select bits 23:16 from the source register 1 3 = Select bits 31:24 from the source register 1 4 = Select bits 15:0 from the source register 1 5 = Select bits 23:8 from the source register 1 6 = Select bits 31:16 from the source register 1 7 = Select bits 31:0 from the source register 1
12-8	Rs1	0-31 = This field selects the register number which contains the first source operand that is to be compared.
7-0	BrOff[7:0]	0-255 = This field contains the 8-LSBs of the 2-s complement signed offset for the branch

(1) Branch is taken if any of the 2 conditions are satisfied: BS or BC.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 13-17. Format 5b: (Quick Bit Test and Branch – Immediate Op2)

31	29	28	27	26	25	24	23	21	20	16
OP		BS	BC	BrOff[[9:8]		IO	RESERVED		Imm	
15	13	12				8	7			0
Rs1Sel		Rs1					BrOff[7:0]			

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

Table 13-22. Format 5b: (Quick Bit Test and Branch – Immediate Op2)

Bit	Field	Description
31-29	OP	0b110 = Specifies Format 5
28	BS	0-1 = Bit Set: If set specifies that branch should be taken if Rs1[Op2[4:0]] == 1[(1)]
27	BC	0-1 = Bit Clear: If set specifies that branch should be taken if Rs1[Op2[4:0]] == 0[(1)]
26-25	BrOff[9:8]	0-3 = This field contains the 2-MSBs of the 2s complement signed offset for the branch
24	IO	1 = Op2 is a 5-bit immediate value
23-21	RESERVED	
20-16	Imm	0-31 = This field selects the bit number of the first operand which is to be compared.
15-13	Rs1Sel	0 = Select bits 7:0 from the source register 1 1 = Select bits 15:8 from the source register 1 2 = Select bits 23:16 from the source register 1 3 = Select bits 31:24 from the source register 1 4 = Select bits 15:0 from the source register 1 5 = Select bits 23:8 from the source register 1 6 = Select bits 31:16 from the source register 1 7 = Select bits 31:0 from the source register 1
12-8	Rs1	0-31 = This field selects the register number which contains the first source operand that is to be compared
7-0	BrOff[7:0]	0-255 = This field contains the 8-LSBs of the 2-s complement signed offset for the branch

(1) Branch is taken if any of the 2 conditions are satisfied: BS or BC.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 13-18. Format 6a: (LBBO/SBBO - Register Offset)

31	29	28	27	25	24	23	21	20	16
OP		LoadStore	BurstLen[6:4]		IO	RoSel		Ro	
15	13	12		8	7	6	5	4	0
BurstLen[3:1]		Rb			BurstLen[0]	RxByteAddr		Rx	

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

Table 13-23. Format 6a: (LBBO/SBBO - Register Offset)

Bit	Field	Description
31-29	OP	0b111 = Specifies Format 6a / 6b
28	LoadStore	0 = SBBO 1 = LBBO
27-25	BurstLen[6:4]	The following 3 fields specify the burst length (in bytes) for the transfer
24	IO	0 = The offset is to be taken from a register
23-21	RoSel	0 = Select bits 7:0 from the offset register 1 = Select bits 15:8 from the offset register 2 = Select bits 23:16 from the offset register 3 = Select bits 31:24 from the offset register 4 = Select bits 15:0 from the offset register 5 = Select bits 23:8 from the offset register 6 = Select bits 31:16 from the offset register 7 = Select bits 31:0 from the offset register
20-16	Ro	0-31 = This field selects the register number which contains the beginning offset for the transfer.
15-13	BurstLen[3:1]	
12-8	Rb	0-31 = This field selects the register number which contains the base address for the transfer
7	BurstLen[0]	0-123 = byte count = BurstLen + 1 (1 – 124 Bytes) 124 = byte count = R0 bits 7:0 125 = byte count = R0 bits 15:8 126 = byte count = R0 bits 23:16 127 = byte count = R0 bits 31:24
6-5	RxByteAddr	0-3 = This field selects the beginning byte number in the source / destination register for the data transfer
4-0	Rx	0-30 = This field selects the beginning source / destination register number for the data transfer.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 13-19. Format 6b: (LBBO/SBBO - Immediate Offset)

31	29	28	27	25	24	23	16
OP		LoadStore	BurstLen[6:4]		IO	Imm	
15	13	12	9	8	7	6 5	4 0
BurstLen[3:1]		Rb			BurstLen[0]	RxByteAddr	Rx

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

Table 13-24. Format 6b: (LBBO/SBBO - Immediate Offset)

Bit	Field	Description
31-29	OP	0b111 = Specifies Format 6a / 6b
28	LoadStore	0 = SBBO 1 = LBBO
27-25	BurstLen[6:4]	The following 3 fields specify the burst length (in bytes) for the transfer.
24	IO	1 = The offset is an immediate 8-bit value
23-16	Imm	0-255 = Immediate 8-bit offset value
15-13	BurstLen[3:1]	
12-8	Rb	0-31 = This field selects the register number which contains the base address for the transfer
7	BurstLen[0]	0-123 = byte count = BurstLen +1 (1 – 124 Bytes) 124 = byte count = R0 bits 7:0 125 = byte count = R0 bits 15:8 126 = byte count = R0 bits 23:16 127 = byte count = R0 bits 31:24
6-5	RxByteAddr	0-3 = This field selects the beginning byte number in the source / destination register for the data transfer
4-0	Rx	0-30 = This field selects the beginning source / destination register number for the data transfer.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 13-20. Format 6c: (LBCO/SBCO - Register Offset)

31	29	28	27	25	24	23	21	20	16
OP		LoadStore	BurstLen[6:4]		IO	RoSel		Ro	
15	13	12		8	7	6	5	4	0
BurstLen[3:1]		Cb			BurstLen[0]	RxByteAddr		Rx	

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

Table 13-25. Format 6c: (LBCO/SBCO - Register Offset)

Bit	Field	Description
31-29	OP	0b100 = Specifies Format 6c / 6d
28	LoadStore	0 = SBCO 1 = LBCO
27-25	BurstLen[6:4]	The following 3 fields specify the burst length (in bytes) for the transfer.
24	IO	0 = The offset is to be taken from a register
23-21	RoSel	0 = Select bits 7:0 from the offset register 1 = Select bits 15:8 from the offset register 2 = Select bits 23:16 from the offset register 3 = Select bits 31:24 from the offset register 4 = Select bits 15:0 from the offset register 5 = Select bits 23:8 from the offset register 6 = Select bits 31:16 from the offset register 7 = Select bits 31:0 from the offset register
20-16	Ro	0-31 = This field selects the register number which contains the beginning offset for the transfer.
15-13	BurstLen[3:1]	
12-8	Cb	0-31 = This field selects the constant table entry number which contains the base address for the transfer
7	BurstLen[0]	0-123 = byte count = BurstLen + 1 (1 – 124 Bytes) 124 = byte count = R0 bits 7:0 125 = byte count = R0 bits 15:8 126 = byte count = R0 bits 23:16 127 = byte count = R0 bits 31:24
6-5	RxByteAddr	0-3 = This field selects the beginning byte number in the source / destination register for the data transfer.
4-0	Rx	0-30 = This field selects the beginning source / destination register number for the data transfer.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 13-21. Format 6d: (LBCO/SBCO - Immediate Offset)

31	29	28	27	25	24	23	16
OP		LoadStore	BurstLen[6:4]		IO	Imm	
15	13	12	8	7	6	5	4 0
BurstLen[3:1]		Cb		BurstLen[0]	RxByteAddr		Rx

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

Table 13-26. Format 6d: (LBCO/SBCO - Immediate Offset)

Bit	Field	Description
31-29	OP	0b100 = Specifies Format 6c / 6d
28	LoadStore	0 = SBCO 1 = LBCO
27-25	BurstLen[6:4]	The following 3 fields specify the burst length (in bytes) for the transfer.
24	IO	1 = The offset is an immediate 8-bit value
23-16	Imm	0-255 = Immediate 8-bit offset value
15-13	BurstLen[3:1]	
12-8	Cb	0 – 31 This field selects the constant table entry number which contains the base address for the transfer
7	BurstLen[0]	0-123 = byte count = BurstLen + 1 (1 – 124 Bytes) 124 = byte count = R0 bits 7:0 125 = byte count = R0 bits 15:8 126 = byte count = R0 bits 23:16 127 = byte count = R0 bits 31:24
6-5	RxByteAddr	0=3 = This field selects the beginning byte number in the source / destination register for the data transfer
4-0	Rx	0-30 = This field selects the beginning source / destination register number for the data transfer.

13.7 PRU Interrupt Controller

13.7.1 Introduction

The PRUSS interrupt controller (INTC) is an hardware interface between interrupts coming from different parts of the system (these are referred to as system events), and the PRUs interrupt inputs.

The PRUSS INTC has the following features:

- Capturing up to 32 System Events external to the PRUSS
- 32 additional System events generated by the PRUs
- Supports up to 10 interrupt channels
- Generation of 10 Host Interrupts
  - 2 Host Interrupts for the PRUs
  - 8 Host Interrupts exported from the PRUSS for signaling the host (ARM/DSP) interrupt controllers
- Each system event can be enabled and disabled
- Each host event can be enabled and disabled.
- Hardware prioritization of events

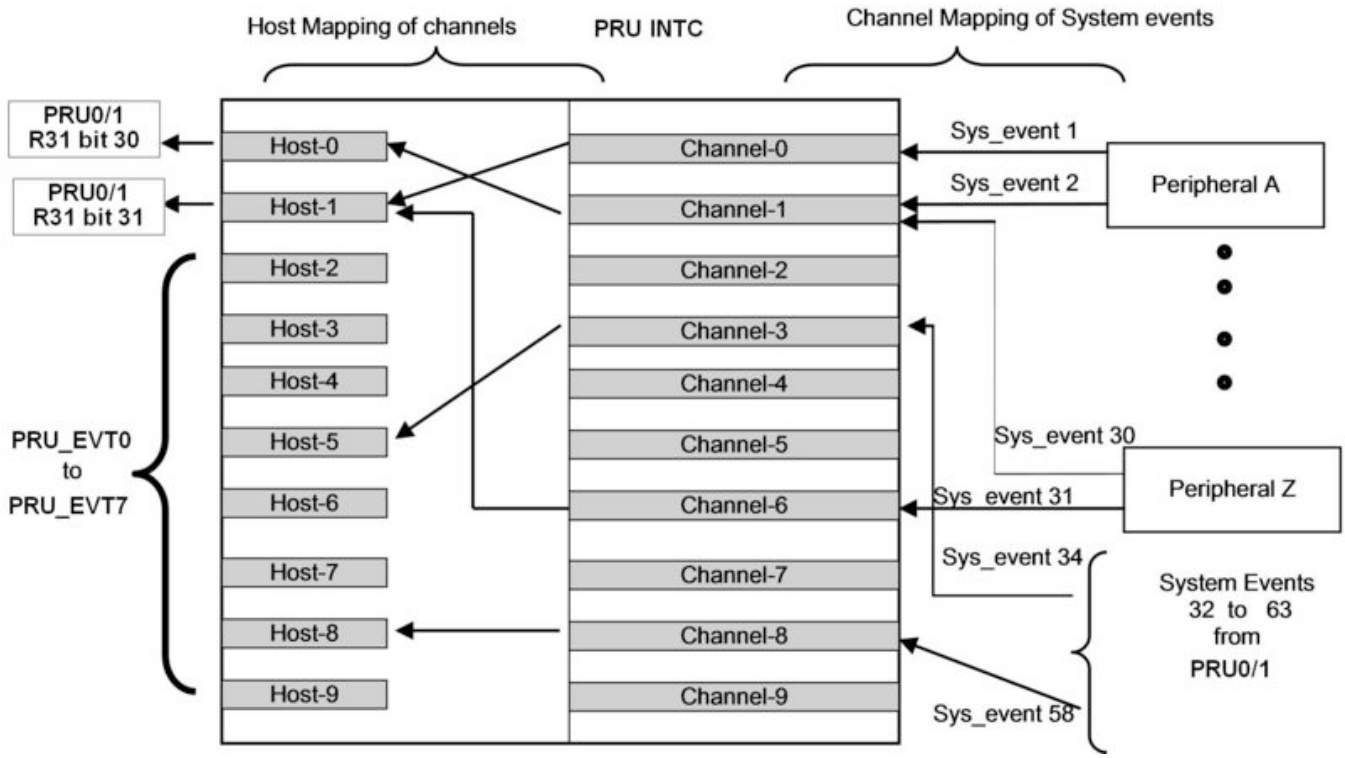
13.7.2 Interrupt Mapping

The PRUSS INTC supports up to 64 system interrupts from different peripherals and PRUs to be mapped to 10 channels inside the INTC (see Figure 1). Interrupts from these 10 channels are further mapped to 10 Host Interrupts.

Copyright © 2013–2016, Texas Instruments Incorporated

- Any of the 64 system interrupts can be mapped to any of the 10 channels.
- Multiple interrupts can be mapped to a single channel.
- An interrupt should not be mapped to more than one channel.
- Any of the 10 channels can be mapped to any of the 10 host interrupts. It is recommended to map channel "x" to host interrupt "x", where x is from 0 to 9
- A channel should not be mapped to more than one host interrupt.
- For channels mapping to the same host interrupt, lower number channels have higher priority.
- For interrupts on same channel, priority is determined by the hardware interrupt number. The lower the interrupt number, the higher the priority.
- Host Interrupt 0 is connected to bit 30 in register 31 of PRU0 and PRU1.
- Host Interrupt 1 is connected to bit 31 in register 31 for PRU0 and PRU1. Host Interrupts 2 through 9 exported from PRUSS for signaling ARM and DSP interrupt controllers generating system Events PRUSS_EVTOUT0 to PRUSS_EVTOUT7 respectively.

Figure 13-22.



13.7.3 PRUSS System Events

System events 0 through 31 are external to the PRUSS subsystem and generated from different peripherals. The source of the first 32 events from the device is listed in the Table 13-27.

System events 32 to 63 are generated by a PRU writing to its own R31 register. The system interrupt is used to either post a completion event to one of the host CPUs (ARM/DSP) or to signal the other PRU core of the PRUSS. For more information on the steps to generate the system events 32 to 63 refer to Section 13.4.1.

The device includes a mux that with a single select signal selects the PRUSS EVT inputs as shown in the table below. The control signal, PRUSSEVTSEL, can be modified by software in system register CFGCHIP3[3]. PRUSSEVTSEL defaults to 0 after reset. Note that not all system events are defined for all devices. Refer to the device datasheet or System Reference Guide for more information.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 13-27. PRUSS System Events [0:31] Assignments

Event	PRUSSEVTSEL = 0 Description	PRUSSEVTSEL = 1 Description
0	Emulation Suspend Signal (Software Use Only)	Emulation Suspend Signal (Software Use Only)
1	ECAP0 Interrupt	Timer64P2_T12CMPEVT0
2	ECAP1 Interrupt	Timer64P2_T12CMPEVT1
3	Timer64P0 Event Out 12	Timer64P2_T12CMPEVT2
4	ECAP2 Interrupt	Timer64P2_T12CMPEVT3
5	McASP0 TX DMA Request	Timer64P2_T12CMPEVT4
6	McASP0 RX DMA Request	Timer64P2_T12CMPEVT5
7	McBSP0 TX DMA Request	Timer64P2_T12CMPEVT6
8	McBSP0 RX DMA Request	Timer64P2_T12CMPEVT7
9	McBSP1 TX DMA Request	Timer64P3_T12CMPEVT0
10	McBSP1 RX DMA Request	Timer64P3_T12CMPEVT1
11	SPI0 Interrupt 0	Timer64P3_T12CMPEVT2
12	SPI1 Interrupt 0	Timer64P3_T12CMPEVT3
13	UART0 Interrupt	Timer64P3_T12CMPEVT4
14	UART1 Interrupt	Timer64P3_T12CMPEVT5
15	I2C0 Interrupt	Timer64P3_T12CMPEVT6
16	I2C1 Interrupt	Timer64P3_T12CMPEVT7
17	UART2 Interrupt	Timer64P0_T12CMPEVT0 or Timer64P0_T12CMPEVT1 or Timer64P0_T12CMPEVT2 or Timer64P0_T12CMPEVT3 or Timer64P0_T12CMPEVT4 or Timer64P0_T12CMPEVT5 or Timer64P0_T12CMPEVT6 or Timer64P0_T12CMPEVT7
18	MMCSD0 Interrupt 0	Timer64P2 Event Out 12
19	MMCSD0 Interrupt 1	Timer64P3 Event Out 12
20	USB0 (USB2.0 HS OTG) Subsystem Interrupt Request (aggregated from subsystem's INTD)	Timer64P1 Event Out 12
21	USB1 (USB1.1 FS OHCI) Subsystem IRQ Interrupt	UART1 Interrupt
22	Timer64P0 Event Out 34	UART2 Interrupt
23	ECAP0 input (output from mux)	SPI0 Interrupt 0
24	EPWM0 Interrupt	EPWM0 Interrupt
25	EPWM1 Interrupt	EPWM1 Interrupt
26	SATA Interrupt	SPI1 Interrupt 0
27	EDMA3_0_CC0_INT2 (region 2)	GPIO Bank 0 Interrupt
28	EDMA3_0_CC0_INT3 (region 3)	GPIO Bank 1 Interrupt
29	UHPI CPU_INT	McBSP0 TX DMA Request
30	EPWM0TZ Interrupt or EPWM1TZ Interrupt	McBSP0 RX DMA Request
31	McASP0 TX Interrupt or McASP0 RX Interrupt	McASP0 TX Interrupt or McASP0 RX Interrupt

Copyright © 2013–2016, Texas Instruments Incorporated

13.7.4 ARM and DSP Interrupt Controller Mapping

Events PRUSS_EVTOUT0 to PRUSS_EVTOUT7 are mapped to the ARM and DSP interrupt controllers. The following tables show the interrupt mapping.

Table 13-28. ARM Interrupt Controller Mapping

Event Number	Source
3	EVTOUT0
4	EVTOUT1
5	EVTOUT2
6	EVTOUT3
7	EVTOUT4
8	EVTOUT5
9	EVTOUT6
10	EVTOUT7

Table 13-29. DSP Interrupt Controller Mapping

Event Number	Source
6	EVTOUT0
17	EVTOUT1
22	EVTOUT2
35	EVTOUT3
66	EVTOUT4
39	EVTOUT5
44	EVTOUT6
50	EVTOUT7

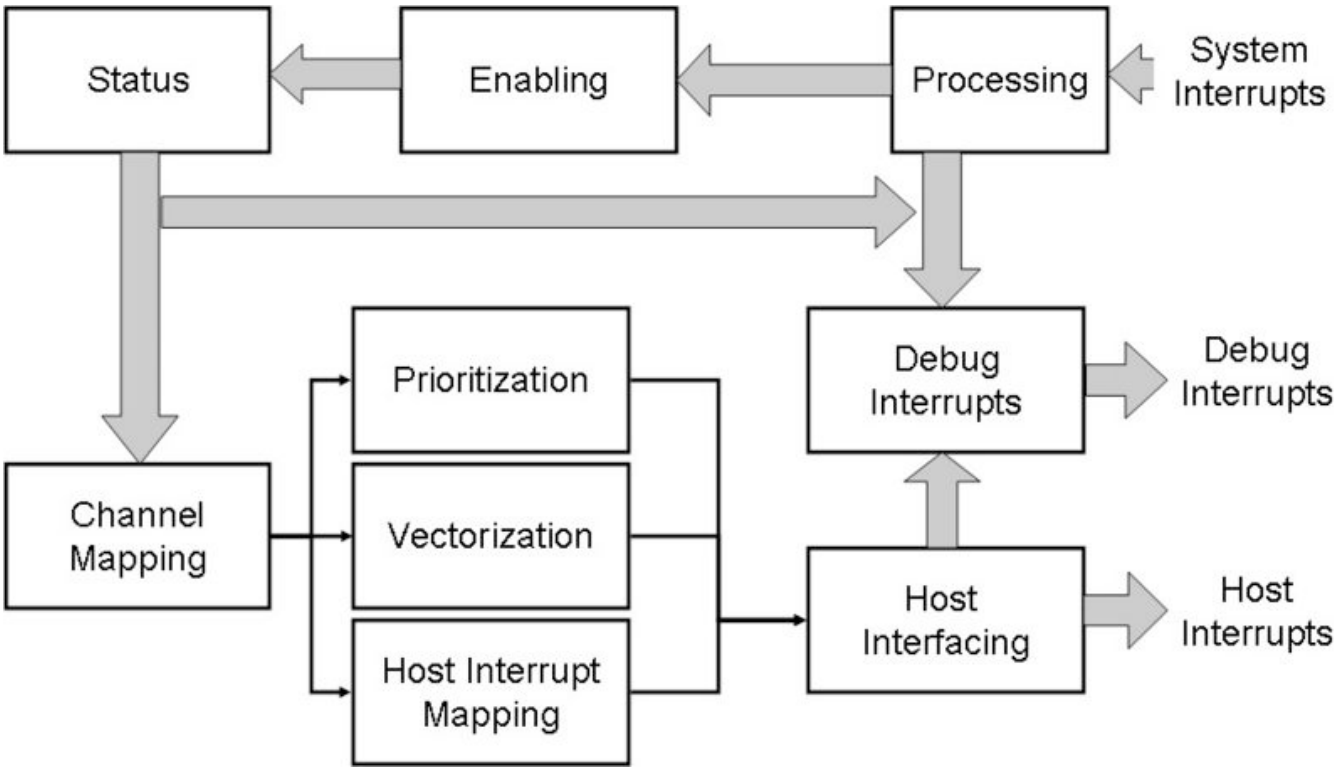
Copyright © 2013–2016, Texas Instruments Incorporated

13.7.5 INTC Methodology

The INTC module controls the system event mapping to the host interrupt interface. System events are generated by the device peripherals or PRUs. The INTC receives the system interrupts and maps them to internal channels. The channels are used to group interrupts together and to prioritize them. These channels are then mapped onto the host interrupts. Interrupts from system side are active high in polarity. Also, they are pulse type of interrupts.

The INTC encompasses many functions to process the system interrupts and prepare them for the host interface. These functions are: processing, enabling, status, channel mapping, host interrupt mapping, prioritization, and host interfacing. Figure 13-23 illustrates the flow of system interrupts through the functions to the host. The following subsections describe each part of the flow.

Figure 13-23.



13.7.5.1 Interrupt Processing

This block does following tasks:

- Synchronization of slower and asynchronous interrupts
- Conversion of polarity to active high
- Conversion of interrupt type to pulse interrupts

After the "processing block", all interrupts will be active high pulses.

13.7.5.2 Interrupt Enabling

The next stage of INTC is to enable system interrupts based on programmed settings. The following sequence is to be followed to enable interrupts:

1. Enable all host interrupts: By setting the ENABLE bit in the global enable register (GER) to 1, all host interrupts will be enabled. Individual host interrupts are still enabled or disabled from their individual enables and are not overridden by the global enable
2. Enable required host interrupts: By writing to the INDEX field in the host interrupt enable indexed set register (HIEISR), enable the required host interrupts. The host interrupt to enable is the index value

Copyright © 2013–2016, Texas Instruments Incorporated

written. This enables the host interrupt output or triggers the output again if that host interrupt is already enabled.

3. Enable required system interrupts: System interrupts that are required to get propagated to host are to be enabled individually by writing to INDEX field in the system interrupt enable indexed set register (EISR). The interrupt to enable is the index value written. This sets the Enable Register bit of the given index.

System interrupts can also be enabled by setting the bits of the system interrupt enable set registers (ESR1-ESR3). Only enabled system interrupts will generate an interrupt to the host.

13.7.5.3 Interrupt Status Checking

The next stage is to capture which system interrupts are pending. There are two kinds of pending status: raw status and enabled status. Raw status is the pending status of the system interrupt without regards to the enable bit for the system interrupt. The raw status of system interrupts is captured in system interrupt status raw/set registers (SRSR1-SRSR2). Enabled status is the pending status of the system interrupts with the enable bits set. When the enable bit is not set, the enabled status will always be inactive. The enabled status of system interrupts is captured in system interrupt status enabled/clear registers (SECR1-SECR2).

Status of system interrupt 'N' is indicated by the Nth bit of SECR1-SECR2. Since there are 64 system interrupts, two 32-bit registers are used to capture the enabled status of interrupts. The pending status reflects whether the system interrupt occurred since the last time the status register bit was cleared. Each bit in the status register can be individually cleared.

13.7.5.4 Interrupt Channel Mapping

The INTC has 10 internal channels to which enabled system interrupts can be mapped. Channel 0 has highest priority and channel 9 has the lowest priority. Channels are used to group the system interrupts into a smaller number of priorities that can be given to a host interface with a very small number of interrupt inputs.

When multiple system interrupts are mapped to the same channel their interrupts are ORed together so that when either is active the output is active. The channel map registers (CMR1-CMR16) define the channel for each system interrupt. There is one register per 4 system interrupts; therefore, there are 16 channel map registers for a system of 64 interrupts. Channels for each system interrupt can be set using these registers.

13.7.5.5 Host Interrupt Mapping

The INTC generates 10 host interrupts which can be . The hosts can be the two PRUs, the ARM CPU, and DSP CPU. The 10 channels from the INTC can be mapped to any of the 10 host interrupts. The host map registers (HMR1-HMR3) define the host interrupt for each channel. There is one register per 4 channels; therefore, there are 3 host map registers for 10 channels. Multiple channels can be mapped to the same host interrupt. When multiple channels are mapped to the same host interrupt prioritization is done to select which interrupt is in the highest-priority channel and which should be sent first to the host.

13.7.5.6 Interrupt Prioritization

The next stage of the INTC is prioritization. Since multiple interrupts can feed into a single channel and multiple channels can feed into a single host interrupt, it is necessary to read the status of all system interrupts to determine the highest priority interrupt that is pending. The INTC provides hardware to perform this prioritization with a given scheme so that software does not have to do this. There are two levels of prioritization:

- The first level of prioritization is between the active channels for a host interrupt. Channel 0 has the highest priority and channel 9 has the lowest. So the first level of prioritization picks the lowest numbered active channel.
- The second level of prioritization is between the active system interrupts for the prioritized channel. The system interrupt in position 0 has the highest priority and system interrupt 63 has the lowest priority. So the second level of prioritization picks the lowest position active system interrupt.

Copyright © 2013–2016, Texas Instruments Incorporated

The highest priority interrupt pending across all host interrupts is stored in the global prioritized index register (GPIR). The highest priority pending interrupt with respect to each host interrupt can be obtained using the host interrupt prioritized index registers (HIPIRn).

13.7.5.7 Interrupt Nesting

The INTC can also perform a nesting function in its prioritization. Nesting is a method of disabling certain interrupts (usually lower-priority interrupts) when an interrupt is taken so that only those desired interrupts can trigger to the host while it is servicing the current interrupt. The typical usage is to nest on the current interrupt and disable all interrupts of the same or lower priority (or channel). Then the host will only be interrupted from a higher priority interrupt.

The nesting is done in 1 of 3 methods:

- Nesting for all host interrupts, based on channel priority: When an interrupt is taken, the nesting level is set to its channel priority. From then, that channel priority and all lower priority channels will be disabled from generating host interrupts and only higher priority channels are allowed. When the interrupt is completely serviced, the nesting level is returned to its original value. When there is no interrupt being serviced, there are no channels disabled due to nesting. The global nesting level register (GNLR) allows the checking and setting of the global nesting level across all host interrupts. The nesting level is the channel (and all of lower priority channels) that are nested out because of a current interrupt.
- Nesting for individual host interrupts, based on channel priority: Always nest based on channel priority for each host interrupt individually. When an interrupt is taken on a host interrupt, then, the nesting level is set to its channel priority for just that host interrupt, and other host interrupts do not have their nesting affected. Then for that host interrupt, equal or lower priority channels will not interrupt the host but may on other host interrupts if programmed. When the interrupt is completely serviced the nesting level for the host interrupt is returned to its original value. The host interrupt nesting level registers (HINLR1 and HINLR2) display and control the nesting level for each host interrupt. The nesting level controls which channel and lower priority channels are nested. There is one register per host interrupt.
- Software manually performs the nesting of interrupts. When an interrupt is taken, the software will disable all the host interrupts, manually update the enables for any or all the system interrupts, and then re-enables all the host interrupts. This now allows only the system interrupts that are still enabled to trigger to the host. When the interrupt is completely serviced the software must reverse the changes to re-enable the nested out system interrupts. This method requires the most software interaction but gives the most flexibility if simple channel based nesting mechanisms are not adequate.

13.7.5.8 Interrupt Status Clearing

After servicing the interrupt (after execution of the ISR), interrupt status is to be cleared. If a system interrupt status is not cleared, then another host interrupt may not be triggered or another host interrupt may be triggered incorrectly.It is also essential to clear all system interrupts before the PRU is halted as the PRU does not power down unless all the interrupt status are cleared. For clearing the status of an interrupt, whose interrupt number is N, write a 1 to the Nth bit position in the system interrupt status enabled/clear registers (SECR1-SECR2). System interrupt N can also be cleared by writing the value N into the system interrupt status indexed clear register (SICR).

13.7.5.9 Interrupt Disabling

At any time, if any interrupt is not to be propagated to the host, then that interrupt should be disabled. For disabling an interrupt whose interrupt number is N, write a 1 to the Nth bit in the system interrupt enable clear registers (ECR1-ECR2). System interrupt N can also be disabled by writing the value N in the system interrupt enable indexed clear register (EICR).

13.7.5.10 Configuring the Interrupt Controller

Follow these steps to configure the interrupt controller.

1. Set polarity and type of system event through the System Interrupt Polarity Registers (SIPR1 and SPIR2) and the System Interrupt Type Registers (SITR1 and SITR2). Polarity of all system interrupts is always high. Type of all system interrupts is always pulse.

Copyright © 2013–2016, Texas Instruments Incorporated

2. Map system event to INTC channel through CHANMAP registers.
3. Map channel to host interrupt through HOSTMAP registers. Recommend channel “x” be mapped to host interrupt “x”.
4. Clear system interrupt by writing 1s to SECR registers.
5. Enable host interrupt by writing index value to HOSTINTENIDX register.
6. Enable interrupt nesting if desired.
7. Globally enable all interrupts through GLBLEN register.

13.8 Registers

Copyright © 2013–2016, Texas Instruments Incorporated

13.8.1 PRUSS Memory Map

13.8.1.1 Local Memory Map

The PRUSS comprises various distinct addressable regions. Externally the subsystem presents a single 64Kbyte range of addresses. The internal interconnect bus (also called switched central resource, or SCR) of the PRUSS decodes accesses for each of the individual regions. The PRUSS memory map is documented in Table 13-30 (Instruction Space) and in Table 13-31 (Data Space). Note that these two memory maps are implemented inside the PRUSS and are local to the components of the PRUSS.

Table 13-30. Local Instruction Space Memory Map

Start Address	End Address	PRU0	PRU1
0x00000000	0x00000FFF	PRU0 Instruction RAM	PRU1 Instruction RAM

Table 13-31. Local Data Space Memory Map

Start Address	End Address	PRU0	PRU1
0x00000000	0x000001FF	Data RAM 0(1)	Data RAM 1(1)
0x00000200	0x00001FFF	Reserved	Reserved
0x00002000	0x000021FF	Data RAM 1(1)	Data RAM 0(1)
0x00002200	0x00003FFF	Reserved	Reserved
0x00004000	0x00006FFF	INTC Registers	INTC Registers
0x00007000	0x000077FF	PRU0 Registers	PRU0 Registers
0x00007800	0x00007FFF	PRU1 Registers	PRU1 Registers
0x00008000	0x0000FFFF	Reserved	Reserved
0x00010000	0xFFFFFFFF	Reserved	Reserved

(1) Note that PRU0 accesses Data RAM0 at address 0x00000000, also PRU1 accesses Data RAM1 at address 0x00000000. Data RAM0 is intended to be the primary data memory for PRU0 and Data RAM1 is intended to be the primary data memory for PRU1. However for passing information between PRUs, each PRU can access the data ram of the other PRU at address 0x00002000.

13.8.1.2 Global Memory Map

The global view of the PRUSS internal memories and control ports is documented in Table 13-32. The offset addresses of each region are implemented inside the PRUSS but the global device memory mapping places the PRUSS slave port in the address range 0x01C30000-0x01C3FFFF. The PRU0 and PRU1 can use either the local or global addresses to access their internal memories, but using the local addresses will provide access time several cycles faster than using the global addresses. This is because when accessing via the global address the access needs to be routed through the switch fabric outside PRUSS and back in through the PRUSS slave port.

Table 13-32. Subsystem Global Memory Map

Start Address	End Address	Region
0x01C30000	0x01C301FF	Data RAM 0
0x01C30200	0x01C31FFF	Reserved
0x01C32000	0x01C321FF	Data RAM 1
0x01C32200	0x01C33FFF	Reserved
0x01C34000	0x01C36FFF	INTC Registers
0x01C37000	0x01C377FF	PRU0 Registers
0x01C37800	0x01C37FFF	PRU1 Registers
0x01C38000	0x01C38FFF	PRU0 Instruction RAM
0x01C39000	0x01C3BFFF	Reserved
0x01C3C000	0x01C3CFFF	PRU1 Instruction RAM

Copyright © 2013–2016, Texas Instruments Incorporated

Table 13-32. Subsystem Global Memory Map (continued)

Start Address	End Address	Region
0x01C3D000	0x01C3FFFF	Reserved

Each of the PRUs can access the rest of the device memory (including memory mapped peripheral and configuration registers) using the global memory space addresses. Please refer to the device's System Reference Guide or datasheet for device specific memory mapping.

13.8.1.3 PRU Memory Map Overview

The PRU control and status registers region contains control and status registers for the PRU. The control / status registers region memory map is shown in Table 13-33.

Table 13-33. PRU Control/Status Register Memory Map

Address Offset	Register Name	Register Description
0h	CONTROL	PRU Control Register
4h	STATUS	PRU Status Register
8h	WAKEUP	PRU Wakeup Enable Register
Ch	CYCLECNT	PRU Cycle Count
10h	STALLCNT	PRU Stall Count
20h	CONTABBLKIDX0	PRU Constant Table Block Index Register 0
28h	CONTABPROPTR0	PRU Constant Table Programmable Pointer Register 0
2Ch	CONTABPROPTR1	PRU Constant Table Programmable Pointer Register 1
400h to 47Ch	INTGPR0 to INTGPR31	PRU Internal General Purpose Registers (for Debug)
480h to 4FCh	INTCTER0 to INTCTER31	PRU Internal Constants Table Entry Registers (for Debug)

13.8.1.3.1 CONTROL Register (Offset = 0h)

Figure 13-24. CONTROL Register

31															16
PCRESETVAL															
R/W-0															
15	14					9	8	7			4	3	2	1	0
RUNSTATE	RESERVED						SINGLESTEP	RESERVED				COUNTENABLE	SLEEPING	ENABLE	SOFTRESET
R/W-0	R-0						R/W-0	R-0				R/W-0	R/W-0	R/W-0	R-0 → 1

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

Table 13-34. CONTROL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	PCRESETVAL	R/W	0	Program Counter Reset Value: This field controls the address where the PRU will start executing code from after it is taken out of reset*
15	RUNSTATE	R	0	Run State: This bit indicates whether the PRU is currently executing an instruction or is halted. 0 = PRU is halted and host has access to the instruction RAM and debug registers regions. 1 = PRU is currently running and the host is locked out of the instruction RAM and debug registers regions This bit is used by an external debug agent to know when the PRU has actually halted when waiting for a HALT instruction to execute, a single step to finish, or any other time when the pru_enable has been cleared.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 13-34. CONTROL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
14-9	RESERVED	R	0	
8	SINGLESTEP	R/W	0	Single Step Enable: This bit controls whether or not the PRU will only execute a single instruction when enabled. 0 = PRU will free run when enabled 1 = PRU will execute a single instruction and then the pru_enable bit will be cleared. Note that this bit does not actually enable the PRU, it only sets the policy for how much code will be run after the PRU is enabled. The pru_enable bit must be explicitly asserted. It is legal to initialize both the single_step and pru_enable bits simultaneously. (Two independent writes are not required to cause the stated functionality)
7-4	RESERVED	R	0	
3	COUNTENABLE	R/W	0	PRU Cycle Counter Enable: Enables PRU cycle counters 0 = Counters not enabled 1 = Counters enabled
2	SLEEPING	R/W	0	PRU Sleep Indicator: This bit indicates whether or not the PRU is currently asleep. 0 = PRU is not asleep 1 = PRU is asleep If this bit is written to a 0, the PRU will be forced to power up from sleep mode.
1	ENABLE	R/W	0	Processor Enable: This bit controls whether or not the PRU is allowed to fetch new instructions. 0 = PRU is disabled 1 = PRU is enabled If this bit is de-asserted while the PRU is currently running and has completed the initial cycle of a multi-cycle instruction (LBxO,SBxO,SCAN, etc.), the current instruction will be allowed to complete before the PRU pauses execution. Otherwise, the PRU will halt immediately. Because of the unpredictability/timing sensitivity of the instruction execution loop, this bit is not a reliable indication of whether or not the PRU is currently running. The pru_state bit should be consulted for an absolute indication of the run state of the core. When the PRU is halted, it's internal state remains coherent therefore this bit can be reasserted without issuing a software reset and the PRU will resume processing exactly where it left off in the instruction stream.
0	SOFTRESET	R	0 → 1	Soft Reset: When this bit is cleared, the PRU will be reset. This bit is set back to 1 on the next cycle after it has been cleared.

Copyright © 2013–2016, Texas Instruments Incorporated

13.8.1.3.2 STATUS Register (Offset = 4h)

Figure 13-25. STATUS Register

31	16	15	0
RESERVED		PCOUNTER	
R-0		R-0	

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

Table 13-35. STATUS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0	
15-0	PCOUNTER	R	0	Program Counter: This field is a registered (1 cycle delayed) reflection of the PRU program counter[(1)]

(1) Note that the PC is an instruction address where each instruction is a 32 bit word. This is not a byte address and to compute the byte address just multiply the PC by 4 (PC of 2 = byte address of 0x8, or PC of 8 = byte address of 0x20).

13.8.1.3.3 WAKEUP Register (Offset = 8h)

31	0
BITWISEENABLES	
R/W-0	

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

Figure 13-26. WAKEUP Register

31	0
BITWISEENABLES	
R/W-0	

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

Table 13-36. WAKEUP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	BITWISEENABLES	R/W	0	Wakeup Enables: This field is ANDed with the incoming R31 status inputs (whose bit positions were specified in the stmap parameter) to produce a vector which is unary ORed to produce the status_wakeup source for the core. Setting any bit in this vector will allow the corresponding status input to wake up the core when it is asserted high. The PRU should set this enable vector prior to executing a SLP (sleep) instruction to ensure that the desired sources can wake up the core.

13.8.1.3.4 CYCLECNT Register (Offset = Ch)

This register counts the number of cycles for which the PRU has been enabled.

Figure 13-27. CYCLECNT Register

31	0
CYCLECOUNT	
R/WC-0	

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

Copyright © 2013–2016, Texas Instruments Incorporated

Table 13-37. CYCLECNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CYCLECOUNT	R/WC	0	This value is incremented by 1 for every cycle during which the PRU is enabled and the counter is enabled (both bits "ENABLE" and "COUNTENABLE" set in the PRU control register). Counting halts while the PRU is disabled or counter is disabled, and resumes when re-eneabled. Counter clears the "COUNTENABLE" bit in the PRU control register when the count reaches 0xFFFFFFFF. (Count does does not wrap). The register can be read at any time. The register can be cleared when the counter or PRU is disabled. Clearing this register also clears the PRU Stall Count Register.

13.8.1.3.5 STALLCNT Register (Offset = 10h)

This register counts the number of cycles for which the PRU has been enabled, but unable to fetch a new instruction. It is linked to the Cycle Count Register (0x0C) such that this register reflects the stall cycles measured over the same cycles as counted by the cycle count register. Thus the value of this register is always less than or equal to cycle count.

Figure 13-28. STALLCNT Register

31	0
STALLCOUNT	
R-0	

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

Table 13-38. STALLCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	STALLCOUNT	R	0	This value is incremented by 1 for every cycle during which the PRU is enabled and the counter is enabled (both bits "ENABLE" and "COUNTENABLE" set in the PRU control register), and the PRU was unable to fetch a new instruction for any reason. Counting halts while the PRU is disabled or the counter is disabled, and resumes when re-enabled. The register can be read at any time. The register is cleared when PRU Cycle Count Register is cleared.

13.8.1.3.6 CONTABBLKIDX0 Register (Offset = 20h)

This register is used to set the block indices which are used to modify entries 24 and 25 in the PRU Constant Table. This register can be written by the PRU whenever it needs to change to a new base pointer for a block in the State / Scratchpad RAM. This function is useful since the PRU is often processing multiple processing threads which require it to change contexts. The PRU can use this register to avoid requiring excessive amounts of code for repetitive context switching. The format of this register is as follows:

Figure 13-29. CONTABBLKIDX0 Register

31	20	19	16
RESERVED		C25	
R-0		R/W-0	
15	4	3	0
RESERVED		C24	
R-0		R/W-0	

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

Copyright © 2013–2016, Texas Instruments Incorporated

Table 13-39. CONTABBLKIDX0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0	
19-16	C25	R/W	0	PRU Constant Entry 25 Block Index: This field sets the value that will appear in bits 11:8 of entry 25 in the PRU Constant Table
15-4	RESERVED	R	0	
3-0	C24	R/W	0	PRU Constant Entry 24 Block Index: This field sets the value that will appear in bits 11:8 of entry 24 in the PRU Constant Table

13.8.1.3.7 CONTABPROPTR0 Register (Offset = 28h)

This register allows the PRU to set up the 256-byte page index for entries 28 and 29 in the PRU Constant Table which serve as general purpose pointers which can be configured to point to any locations inside the session router address map. This register is useful when the PRU needs to frequently access certain structures inside the session router address space whose locations are not hard coded such as tables in scratchpad memory. This register is formatted as follows:

Figure 13-30. CONTABPROPTR0 Register

31	16	15	0
C29		C28	
R/W-0		R/W-0	

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

Table 13-40. CONTABPROPTR0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	C29	R/W	0	PRU Constant Entry 29 Pointer: This field sets the value that will appear in bits 23:8 of entry 29 in the PRU Constant Table
15-0	C28	R/W	0	PRU Constant Entry 28 Pointer: This field sets the value that will appear in bits 23:8 of entry 28 in the PRU Constant Table

13.8.1.3.8 CONTABPROPTR1 Register (Offset = 2Ch)

This register functions the same as the PRU Constant Table Programmable Pointer Register 0 but allows the PRU to control entries 30 and 31 in the PRU Constant Table. This register is formatted as follows:

Figure 13-31. CONTABPROPTR1 Register

31	16	15	0
C31		C30	
R/W-0		R/W-0	

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

Table 13-41. CONTABPROPTR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	C31	R/W	0	PRU Constant Entry 31 Pointer: This field sets the value that will appear in bits 23:8 of entry 31 in the PRU Constant Table
15-0	C30	R/W	0	PRU Constant Entry 30 Pointer: This field sets the value that will appear in bits 23:8 of entry 30 in the PRU Constant Table

Copyright © 2013–2016, Texas Instruments Incorporated

13.8.1.3.9 INTGPR0 to INTGPR31 Register (Offset = 400h + 4*n)

This register allows an external agent to debug the PRU while it is disabled. Reading or writing to these registers will have the same effect as a read or write to these registers from an internal instruction in the PRU. For R30, this includes generation of the pulse outputs whenever the register is written. This register is formatted as follows:

31	0
INTGPRn	
R/W-X	

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

Figure 13-32. INTGPR0 to INTGPR31 Register

31	0
INTGPRn	
R/W-X	

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

Table 13-42. INTGPR0 to INTGPR31 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	INTGPRn	R/W	X	PRU Internal GP Register n: Reading / writing this field directly inspects/modifies the corresponding internal register in the PRU internal regfile.

13.8.1.3.10 INTCTER0 to INTCTER31 Register (Offset = 480h + 4*n)

This register allows an external agent to debug the PRU while it is disabled. Since some of the constants table entries may actually depend on system inputs / and or the internal state of the PRU, these registers are provided to allow an external agent to easily determine the state of the constants table. This register is formatted as follows:

Figure 13-33. INTCTER0 to INTCTER31 Register

31	0
INTCTERn	
R/W-X	

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

Table 13-43. INTCTER0 to INTCTER31 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	INTCTERn	R	X	PRU Internal Constants Table Entry n: Reading this field directly inspects the corresponding entry in the PRU internal constants table.

Copyright © 2013–2016, Texas Instruments Incorporated

13.8.1.4 PRU Instruction RAM Region

The instruction RAM region contains storage for instructions. The instruction RAM region memory map is as follows:

Table 13-44. Instruction RAM Memory Region

Address Offset	Register
0x0000	PRU Instruction RAM

13.8.1.4.1 PRU Instruction RAM (0h)

A total of 1K 32-bit words of instruction storage are provided for the PRU. This memory is to be initialized by an external (to the PRUSS) host processor. This region is only accessible to external masters when the PRU is not-running.

Note that the instruction RAM is accessed using byte addresses on the bus. The PC is an instruction address where each instruction is a 32 bit word and is not a byte address. To compute the byte address just multiply the PC by 4 (PC of 2 = byte address of 0x8, or PC of 8 = byte address of 0x20), or to compute the PC just divide the word aligned byte address by 4 (byte address of 0x8 = PC of 2, byte address of 0x20 = PC of 8).

13.8.2 INTC Registers

Table 13-45 lists the memory-mapped registers for the INTC. See Section 13.8.1 for the memory address of these registers.

Table 13-45. PRUSS Interrupt Controller (INTC) Registers

Address Offset	Register Name	Description
0h	REVID	Revision ID Register
4h	CONTROL	Control Register
10h	GLBLEN	Global Enable Register
1Ch	GLBLNSTLVL	Global Nesting Level Register
20h	STATIDXSET	System Interrupt Status Indexed Set Register
24h	STATIDXCLR	System Interrupt Status Indexed Clear Register
28h	ENIDXSET	System Interrupt Enable Indexed Set Register
2Ch	ENIDXCLR	System Interrupt Enable Indexed Clear Register
34h	HSTINTENIDXSET	Host Interrupt Enable Indexed Set Register
38h	HSTINTENIDXCLR	Host Interrupt Enable Indexed Clear Register
80h	GLBLPRIIDX	Global Prioritized Index Register
200h	STATSETINT0	System Interrupt Status Raw/Set Register 0
204h	STATSETINT1	System Interrupt Status Raw/Set Register 1
280h	STATCLRINT0	System Interrupt Status Enabled/Clear Register 0
284h	STATCLRINT1	System Interrupt Status Enabled/Clear Register 1
300h	ENABLESET0	System Interrupt Enable Set Register 0
304h	ENABLESET1	System Interrupt Enable Set Register 1
380h	ENABLECLR0	System Interrupt Enable Clear Register 0
384h	ENABLECLR1	System Interrupt Enable Clear Register 1
400h to 440h	CHANMAP0 to CHANMAP15	Channel Map Registers 0-15
800h to 808h	HOSTMAP0 to HOSTMAP2	Host Map Register 0-2
900h to 928h	HOSTINTPRIIDX0 to HOSTINTPRIIDX9	Host Interrupt Prioritized Index Registers 0-9
D00h	POLARITY0	System Interrupt Polarity Register 0
D04h	POLARITY1	System Interrupt Polarity Register 1
D80h	TYPE0	System Interrupt Type Register 0

Copyright © 2013–2016, Texas Instruments Incorporated

Table 13-45. PRUSS Interrupt Controller (INTC) Registers (continued)

Address Offset	Register Name	Description
D84h	TYPE1	System Interrupt Type Register 1
1100h to 1128h	HOSTINTNSTLVL0 to HOSTINTNSTLVL9	Host Interrupt Nesting Level Registers 0-9
1500h	HOSTINTEN	Host Interrupt Enable Register

13.8.2.1 REVID Register (Offset = 0h)

The Revision Register contains the ID and revision information.

Table 13-46. REVID Register

31	0
REV	
R/O-1	

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

Table 13-47. REVID Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	REV	R/O	1	Revision ID

13.8.2.2 CONTROL Register (Offset = 4h)

The Control Register holds global control parameters and can forces a soft reset on the module.

Table 13-48. CONTROL Register

31	4	3	2	1	0
RESERVED		NEST MODE		RESE RVED	
R-0		R/W-0		R-0	

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

Table 13-49. CONTROL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0	
3-2	NESTMODE	R/W	0	The nesting mode. 0 = no nesting 1 = automatic individual nesting (per host interrupt) 2 = automatic global nesting (over all host interrupts) 3 = manual nesting
1-0	RESERVED	R	0	

13.8.2.3 GLBLEN Register (Offset = 10h)

The Global Enable Register enables all the host interrupts. Individual host interrupts are still enabled or disabled from their individual enables and are not overridden by the global enable.

Table 13-50. GLBLEN Register

31	1	0
RESERVED		ENABLE
R-0		R/W-0

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

Copyright © 2013–2016, Texas Instruments Incorporated

Table 13-51. GLBLEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0	
0	ENABLE	R/W	0	The current global enable value when read. Writes set the global enable

13.8.2.4 GLBLNSTLVL Register (Offset = 1Ch)

The Global Nesting Level Register allows the checking and setting of the global nesting level across all host interrupts when automatic global nesting mode is set. The nesting level is the channel (and all of lower priority) that are nested out because of a current interrupt. *This register is only available when nesting is configured.*

Table 13-52. GLBLNSTLVL Register

31	30 9	8 0
OVERRIDE	RESERVED	NESTLEVEL
W/O-X	R-0	R/W-A

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

Table 13-53. GLBLNSTLVL Register Field Descriptions

Bit	Field	Type	Reset	Description
31	OVERRIDE	W/O	X	Always read as 0. Writes of 1 override the automatic nesting and set the nesting_level to the written data.
30-9	RESERVED	R	0	
8-0	NESTLEVEL	R/W	Ah	The current global nesting level (highest channel that is nested). Writes set the nesting level. In auto nesting mode this value is updated internally unless the auto_override bit is set.

13.8.2.5 STATIDXSET Register (Offset = 20h)

The System Interrupt Status Indexed Set Register allows setting the status of an interrupt. The interrupt to set is the index value written. This sets the Raw Status Register bit of the given index.

Table 13-54. STATIDXSET Register

31 10	9 0
RESERVED	INDEX
R-0	W/S-0

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

Table 13-55. STATIDXSET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0	
9-0	INDEX	W/S	0	Writes set the status of the interrupt given in the index value. Reads return 0.

13.8.2.6 STATIDXCLR Register (Offset = 24h)

The System Interrupt Status Indexed Clear Register allows clearing the status of an interrupt. The interrupt to clear is the index value written. This clears the Raw Status Register bit of the given index.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 13-56. STATIDXCLR Register

31	10	9	0
RESERVED		INDEX	
R-0		W/C-0	

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

Table 13-57. STATIDXCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0	
9-0	INDEX	W/C	0	Writes clear the status of the interrupt given in the index value. Reads return 0.

13.8.2.7 ENIDXSET Register (Offset = 28h)

The System Interrupt Enable Indexed Set Register allows enabling an interrupt. The interrupt to enable is the index value written. This sets the Enable Register bit of the given index.

Table 13-58. ENIDXSET Register

31	10	9	0
RESERVED		INDEX	
R-0		W/S-0	

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

Table 13-59. ENIDXSET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0	
9-0	INDEX	W/S	0	Writes set the enable of the interrupt given in the index value. Reads return 0.

13.8.2.8 ENIDXCLR Register (Offset = 2Ch)

The System Interrupt Enable Indexed Clear Register allows disabling an interrupt. The interrupt to disable is the index value written. This clears the Enable Register bit of the given index.

Table 13-60. ENIDXCLR Register

31	10	9	0
RESERVED		INDEX	
R-0		W/C-0	

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

Table 13-61. ENIDXCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0	
9-0	INDEX	W/C	0	Writes clear the enable of the interrupt given in the index value. Reads return 0.

13.8.2.9 HSTINTENIDXSET Register (Offset = 34h)

The Host Interrupt Enable Indexed Set Register allows enabling a host interrupt output. The host interrupt to enable is the index value written. This enables the host interrupt output or triggers the output again if already enabled.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 13-62. HSTINTENIDXSET Register

31	10	9	0
RESERVED		INDEX	
R-0		W/S-0	

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

Table 13-63. HSTINTENIDXSET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0	
9-0	INDEX	W/S	0	Writes set the enable of the host interrupt given in the index value. Reads return 0.

13.8.2.10 HSTINTENIDXCLR Register (Offset = 38h)

The Host Interrupt Enable Indexed Clear Register allows disabling a host interrupt output. The host interrupt to disable is the index value written. This disables the host interrupt output.

Table 13-64. HSTINTENIDXCLR Register

31	10	9	0
RESERVED		INDEX	
R-0		W/C-0	

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

Table 13-65. HSTINTENIDXCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0	
9-0	INDEX	W/C	0	Writes clear the enable of the host interrupt given in the index value. Reads return 0.

13.8.2.11 GLBLPRIIDX Register (Offset = 80h)

The Global Prioritized Index Register shows the interrupt number of the highest priority interrupt pending across all the host interrupts.

Table 13-66. GLBLPRIIDX Register

31	30	10	9	0
NONE	RESERVED		PRI_INDEX	
R/O-0	R-0		R/O-0	

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

Table 13-67. GLBLPRIIDX Register Field Descriptions

Bit	Field	Type	Reset	Description
31	NONE	R/O	1	No Interrupt is pending. Can be used by host to test for a negative value to see if no interrupts are pending.
30-10	RESERVED	R	0	
9-0	PRI_INDEX	R/O	0	The currently highest priority interrupt index pending across all the host interrupts.

13.8.2.12 STATESETINT0 Register (Offset = 200h)

The System Interrupt Status Raw/Set Registers show the pending enabled status of the system interrupts. Software can write to the Status Set Registers to manually set a system interrupt. There is one bit per system interrupt.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 13-68. STATESETINT0 Register

31	0
RAW_STATUS	
W/S-0	

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

Table 13-69. STATESETINT0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RAW_STATUS	W/S	0	System interrupt raw status and setting of the system interrupts 0 to 31. Reads return the raw status. Write a 1 in a bit position to set the status of the system interrupt. Writing a 0 has no effect.

13.8.2.13 STATESETINT1 Register (Offset = 204h)

The System Interrupt Status Raw/Set Registers show the pending enabled status of the system interrupts. Software can write to the Status Set Registers to manually set a system interrupt. There is one bit per system interrupt.

Table 13-70. STATESETINT1 Register

31	0
RAW_STATUS	
W/S-0	

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

Table 13-71. STATESETINT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RAW_STATUS	W/S	0	System interrupt raw status and setting of the system interrupts 32 to 63. Reads return the raw status. Write a 1 in a bit position to set the status of the system interrupt. Writing a 0 has no effect.

13.8.2.14 STATCLRINT0 Register (Offset = 280h)

The System Interrupt Status Enabled/Clear Registers show the pending enabled status of the system interrupts. Software can write to the Status Clear Registers to clear a system interrupt after it has been serviced. If a system interrupt status is not cleared then another host interrupt may not be triggered or another host interrupt may be triggered incorrectly. There is one bit per system interrupt.

Table 13-72. STATCLRINT0 Register

31	0
ENABLE_STATUS	
W/C-0	

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

Table 13-73. STATCLRINT0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ENABLED_STATUS	W/C	0	System interrupt enabled status and clearing of the system interrupts 0 to 31. Reads return the enabled status (before enabling with the Enable Registers). Write a 1 in a bit position to clear the status of the system interrupt. Writing a 0 has no effect.

Copyright © 2013–2016, Texas Instruments Incorporated

13.8.2.15 STATCLRINT1 Register (Offset = 284h)

The System Interrupt Status Enabled/Clear Registers show the pending enabled status of the system interrupts. Software can write to the Status Clear Registers to clear a system interrupt after it has been serviced. If a system interrupt status is not cleared then another host interrupt may not be triggered or another host interrupt may be triggered incorrectly. There is one bit per system interrupt.

Table 13-74. STATCLRINT1 Register

31	0
ENABLE_STATUS	
W/C-0	

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

Table 13-75. STATCLRINT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ENABLED_STATUS	W/C	0	System interrupt enabled status and clearing of the system interrupts 32to 63. Reads return the enabled status (before enabling with the Enable Registers). Write a 1 in a bit position to clear the status of the system interrupt. Writing a 0 has no effect.

13.8.2.16 ENABLESET0 Register (Offset = 300h)

The System Interrupt Enable Set Register enables system interrupts to trigger outputs. System interrupts that are not enabled do not interrupt the host. There is a bit per system interrupt.

Table 13-76. ENABLESET0 Register

31	0
ENABLE	
W/S-0	

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

Table 13-77. ENABLESET0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ENABLE	W/S	0	System interrupt enables system interrupts 0 to 31. Read returns the enable value (0 = disabled, 1 = enabled) Write a 1 in a bit position to set that enable. Writing a 0 has no effect.

13.8.2.17 ENABLESET1 Register (Offset = 304h)

The System Interrupt Enable Set Register enables system interrupts to trigger outputs. System interrupts that are not enabled do not interrupt the host. There is a bit per system interrupt.

Table 13-78. ENABLESET1 Register

31	0
ENABLE	
W/S-0	

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

Table 13-79. ENABLESET1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ENABLE	W/S	0	System interrupt enables system interrupts 32 to 63. Read returns the enable value (0 = disabled, 1 = enabled) Write a 1 in a bit position to set that enable. Writing a 0 has no effect.

Copyright © 2013–2016, Texas Instruments Incorporated

13.8.2.18 ENABLECLR0 Register (Offset = 380h)

The System Interrupt Enable Clear Register disables system interrupts to map to channels. System interrupts that are not enabled do not interrupt the host. There is a bit per system interrupt.

Table 13-80. ENABLECLR0 Register

31	0
ENABLE	
W/C-0	

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

Table 13-81. ENABLECLR0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ENABLE	W/C	0	System interrupt enables system interrupts 0 to 31. Read returns the enable value (0 = disabled, 1 = enabled) Write a 1 in a bit position to clear that enable. Writing a 0 has no effect.

13.8.2.19 ENABLECLR1 Register (Offset = 384h)

The System Interrupt Enable Clear Register disables system interrupts to map to channels. System interrupts that are not enabled do not interrupt the host. There is a bit per system interrupt.

Table 13-82. ENABLECLR1 Register

31	0
ENABLE	
W/C-0	

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

Table 13-83. ENABLECLR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ENABLE	W/C	0	System interrupt enables system interrupts 32 to 63. Read returns the enable value (0 = disabled, 1 = enabled) Write a 1 in a bit position to clear that enable. Writing a 0 has no effect.

13.8.2.20 CHANMAP0 to CHANMAP15 Register (Offset = 400h to 440h)

The Channel Map Registers specify the channel for each system interrupt. There is one register per 4 system interrupts.

Table 13-84. CHANMAP0 to CHANMAP15 Register

31 24	23 16	15 8	7 0
SYSN3_MAP	SYSN2_MAP	SYSN1_MAP	SYSN_MAP
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 13-85. CHANMAP0 to CHANMAP15 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	SYSN3_MAP	R/W	0	Sets the channel for the system interrupt N + 3.
23-16	SYSN2_MAP	R/W	0	Sets the channel for the system interrupt N + 2.
15-8	SYSN1_MAP	R/W	0	Sets the channel for the system interrupt N + 1.
7-0	SYSN_MAP	R/W	0	Sets the channel for the system interrupt N.

Copyright © 2013–2016, Texas Instruments Incorporated

13.8.2.21 HOSTMAP0 to HOSTMAP2 Register (Offset = 800h to 808h)

The Host Interrupt Map Registers define the host interrupt for each channel. There is one register per 4 channels. *Channels with forced host interrupt mappings will have their fields read-only.*

Table 13-86. HOSTMAP0 to HOSTMAP2 Register

31 24	23 16	15 8	7 0
CHANN3_MAP	CHANN2_MAP	CHANN1_MAP	CHANN_MAP
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 13-87. HOSTMAP0 to HOSTMAP2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	CHANN3_MAP	R/W	0	Sets the host interrupt for channel N + 3
23-16	CHANN2_MAP	R/W	0	Sets the host interrupt for channel N + 2
15-8	CHANN1_MAP	R/W	0	Sets the host interrupt for channel N + 1
7-0	CHANN_MAP	R/W	0	Sets the host interrupt for channel N

13.8.2.22 HOSTINTPRIIDX0 to HOSTINTPRIIDX9 Register (Offset = 900h to 928h)

The Host Interrupt Prioritized Index Registers show the highest priority current pending interrupt for the host interrupt. There is one register per host interrupt.

Table 13-88. HOSTINTPRIIDX0 to HOSTINTPRIIDX9 Register

31	30 10	9 0
NONE	RESERVED	PRI_INDEX
R/O-1	R-0	R/O-0

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

Table 13-89. HOSTINTPRIIDX0 to HOSTINTPRIIDX9 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	NONE	R/O	1	No pending interrupt.
30-10	RESERVED	R	0	
9-0	PRI_INDEX	R/O	0	Interrupt number of the highest priority pending interrupt for this host interrupt.

13.8.2.23 POLARITY0 Register (Offset = D00h)

The System Interrupt Polarity Registers define the polarity of the system interrupts. The polarity of all system interrupts is active high; always write 1 to the bits of this register.

Table 13-90. POLARITY0 Register

31 0
POLARITY
R/W-Default_polarity[N]

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

Copyright © 2013–2016, Texas Instruments Incorporated

Table 13-91. POLARITY0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	POLARITY	R/W	Default_polarity[N]	Interrupt polarity of the system interrupts 0 to 31. 0 = active low 1 = active high

13.8.2.24 POLARITY1 Register (Offset = D04h)

The System Interrupt Polarity Registers define the polarity of the system interrupts. The polarity of all system interrupts is active high; always write 1 to the bits of this register.

Table 13-92. POLARITY1 Register

31	0
POLARITY	
R/W-Default_polarity[N]	

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

Table 13-93. POLARITY1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	POLARITY	R/W	Default_polarity[N]	Interrupt polarity of the system interrupts 32 to 63. 0 = active low 1 = active high

13.8.2.25 TYPE0 Register (Offset = D80h)

The Interrupt Type Registers define the type of the system interrupts. The type of all system interrupts is pulse; always write 0 to the bits of this register.

Table 13-94. TYPE0 Register

31	0
TYPE	
R/W-Default_type[N]	

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

Table 13-95. TYPE0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TYPE	R/W	Default_type[N]	Interrupt type of the system interrupts 0 to 31. 0 = level or pulse interrupt 1 = edge interrupt (required edge detect)

13.8.2.26 TYPE1 Register (Offset = D84h)

The Interrupt Type Registers define the type of the system interrupts. The type of all system interrupts is pulse; always write 0 to the bits of this register.

Table 13-96. TYPE1 Register

31	0
TYPE	
R/W-Default_type[N]	

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

Copyright © 2013–2016, Texas Instruments Incorporated

Table 13-97. TYPE1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TYPE	R/W	Default_type[N]	Interrupt type of the system interrupts 32 to 63. 0 = level or pulse interrupt 1 = edge interrupt (required edge detect)

13.8.2.27 HOSTINTNSTLVL0 to HOSTINTNSTLVL9 Register (Offset = 1100h to 1128h)

The Host Interrupt Nesting Level Registers display and control the nesting level for each host interrupt. The nesting level controls which channel and lower priority channels are nested. There is one register per host interrupt.

Table 13-98. HOSTINTNSTLVL0 to HOSTINTNSTLVL9 Register

31	30 9	8 0
AUTO_OVERRIDE	RESERVED	NESTING_LEVEL
W/O-0	R-0	R/W-0

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

Table 13-99. HOSTINTNSTLVL0 to HOSTINTNSTLVL9 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	AUTO_OVERRIDE	W/O	0	Reads return 0. Writes of a 1 override the auto updating of the nesting_level and use the write data.
30-9	RESERVED	R	0	
8-0	NESTING_LEVEL	R/W	0	Reads return the current nesting level for the host interrupt. Writes set the nesting level for the host interrupt. In auto mode the value is updated internally unless the auto_override is set and then the write data is used.

13.8.2.28 HOSTINTEN Register (Offset = 1500h)

The Host Interrupt Enable Registers enable or disable individual host interrupts. These work separately from the global enables. There is one bit per host interrupt. These bits are updated when writing to the Host Interrupt Enable Index Set and Host Interrupt Enable Index Clear registers.

Table 13-100. HOSTINTEN Register

31 10	9 0
RESERVED	ENABLES
R-0	R/W-0

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

Table 13-101. HOSTINTEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0	
9-0	ENABLES	R/W	0	The enable of the host interrupts (one per bit). 0 = disabled 1 = enabled

Copyright © 2013–2016, Texas Instruments Incorporated

DDR2/mDDR Memory Controller

This chapter describes the DDR2/mobile DDR (mDDR) memory controller.

Topic **Page**

14.1 Introduction **354**
14.2 Architecture **356**
14.3 Supported Use Cases **384**
14.4 Registers **389**

Copyright © 2013–2016, Texas Instruments Incorporated

14.1 Introduction

14.1.1 Purpose of the Peripheral

The DDR2/mDDR memory controller is used to interface with JESD79D-2 standard compliant DDR2 SDRAM devices and JESD209 standard mobile DDR (mDDR) SDRAM devices. Memories types such as DDR1 SDRAM, SDR SDRAM, SBSRAM, and asynchronous memories are not supported. The DDR2/mDDR memory is the major memory location for program and data storage.

14.1.2 Features

The DDR2/mDDR memory controller supports the following features:

- JESD79D-2 standard compliant DDR2 SDRAM
- JESD209 standard compliant mobile DDR (mDDR)
- Data bus width of 16 bits
- CAS latencies:
  - DDR2: 2, 3, 4, and 5
  - mDDR: 2 and 3
- Internal banks:
  - DDR2: 1, 2, 4, and 8
  - mDDR: 1, 2, and 4
- Burst length: 8
- Burst type: sequential
- 1 CS signal
- Page sizes: 256, 512, 1024, and 2048
- SDRAM auto-initialization
- Self-refresh mode
- Partial array self-refresh (for mDDR)
- Power-down mode
- Prioritized refresh
- Programmable refresh rate and backlog counter
- Programmable timing parameters
- Little-endian mode

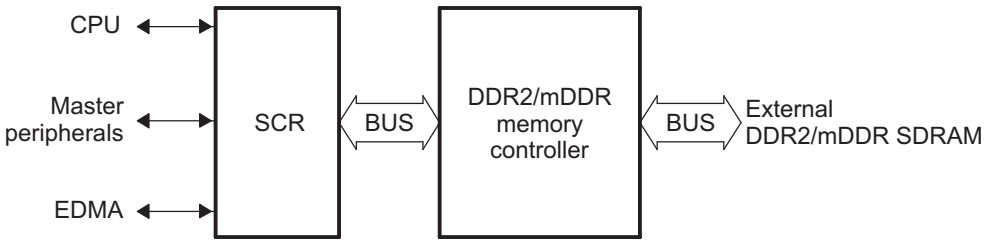
Copyright © 2013–2016, Texas Instruments Incorporated

14.1.3 Functional Block Diagram

The DDR2/mDDR memory controller is the main interface to external DDR2/mDDR memory. Figure 14-1 displays the general data paths to on-chip peripherals and external DDR2/mDDR SDRAM.

Master peripherals, EDMA, and the CPU can access the DDR2/mDDR memory controller through the switched central resource (SCR).

Figure 14-1. Data Paths to DDR2/mDDR Memory Controller



14.1.4 Supported Use Case Statement

The DDR2/mDDR memory controller supports JESD79D-2 DDR2 SDRAM memories and the JESD209 mobile DDR (mDDR) SDRAM memories utilizing 16 bits of the DDR2/mDDR memory controller data bus. See Section 14.3 for more details.

14.1.5 Industry Standard(s) Compliance Statement

The DDR2/mDDR memory controller is compliant with the JESD79D-2 DDR2 SDRAM standard and the JESD209 mobile DDR (mDDR) standard with the following exception:

- On-Die Termination (ODT). The DDR2/mDDR memory controller does not include any on-die terminating resistors. Furthermore, the on-die terminating resistors of the DDR2/mDDR SDRAM device must be disabled by tying the ODT input pin of the DDR2/mDDR SDRAM to ground.

Copyright © 2013–2016, Texas Instruments Incorporated

14.2 Architecture

This section describes the architecture of the DDR2/mDDR memory controller as well as how it is structured and how it works within the context of the system-on-a-chip. The DDR2/mDDR memory controller can gluelessly interface to most standard DDR2/mDDR SDRAM devices and supports such features as self-refresh mode and prioritized refresh. In addition, it provides flexibility through programmable parameters such as the refresh rate, CAS latency, and many SDRAM timing parameters. The following sections include details on how to interface and properly configure the DDR2/mDDR memory controller to perform read and write operations to externally-connected DDR2/mDDR SDRAM devices. Also, Section 14.3 provides a detailed example of interfacing the DDR2/mDDR memory controller to a common DDR2/mDDR SDRAM device.

14.2.1 Clock Control

The DDR2/mDDR memory controller receives two input clocks from internal clock sources, VCLK and 2X_CLK (Figure 14-2). VCLK is a divided-down version of the PLL0 clock. 2X_CLK is the PLL1 clock. 2X_CLK should be configured to clock at the frequency of the desired data rate, or stated similarly, it should operate at twice the frequency of the desired DDR2/mDDR memory clock. DDR_CLK and $\overline{\text{DDR_CLK}}$ are the two output clocks of the DDR2/mDDR memory controller providing the interface clock to the DDR2/mDDR SDRAM memory. These two clocks operate at a frequency of 2X_CLK/2.

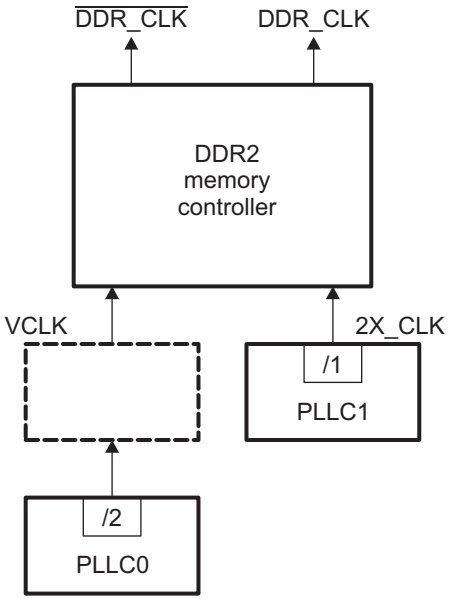
14.2.1.1 Clock Source

VCLK and 2X_CLK are sourced from two independent PLLs (Figure 14-2). VCLK is sourced from PLL controller 0 (PLLC0) and 2X_CLK is sourced from PLL controller 1 (PLLC1).

VCLK is clocked at a fixed divider ratio of PLL0. This divider is fixed at 2, meaning VCLK is clocked at a frequency of PLL0/2.

The clock from PLLC1 is not divided before reaching 2X_CLK. PLLC1 should be configured to supply 2X_CLK at the desired frequency. For example, if a 138-MHz DDR2/mDDR interface clock (DDR_CLK) is desired, then PLLC1 must be configured to generate a 276-MHz clock on 2X_CLK.

Figure 14-2. DDR2/mDDR Memory Controller Clock Block Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

14.2.1.2 Clock Configuration

The frequency of 2X_CLK is configured by selecting the appropriate PLL multiplier. The PLL multiplier is selected by programming registers within PLLC1. The PLLC1 divider ration is fixed at 1. For information on programming the PLL controllers, see the *Phase-Locked Loop Controller (PLLC)* chapter. For information on supported clock frequencies, see the *Device Clocking* chapter and your device-specific data manual.

NOTE: PLLC1 should be configured and a stable clock present on 2X_CLK before releasing the DDR2/mDDR memory controller from reset.

14.2.1.3 DDR2/mDDR Memory Controller Internal Clock Domains

There are two clock domains within the DDR2/mDDR memory controller. The two clock domains are driven by VCLK and a divided-down by 2 version of 2X_CLK called MCLK. The command FIFO, write FIFO, and read FIFO described in Section 14.2.6 are all on the VCLK domain. From this, VCLK drives the interface to the peripheral bus.

The MCLK domain consists of the DDR2/mDDR memory controller state machine and memory-mapped registers. This clock domain is clocked at the rate of the external DDR2/mDDR memory, 2X_CLK/2.

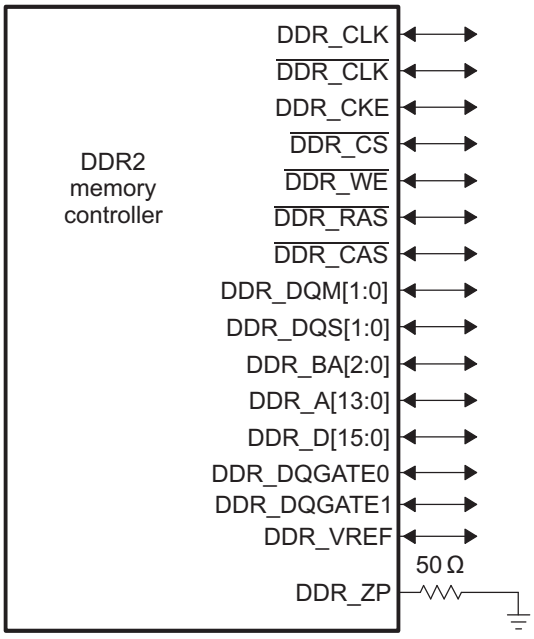
To conserve power within the DDR2/mDDR memory controller, VCLK, MCLK, and 2X_CLK may be stopped. See Section 14.2.16 for proper clock stop procedures.

14.2.2 Signal Descriptions

The DDR2/mDDR memory controller signals are shown in Figure 14-3 and described in DDR2/mDDR Memory Controller Signal Descriptions. The following features are included:

- The maximum data bus is 16-bits wide.
- The address bus is 14-bits wide with an additional three bank address pins.
- Two differential output clocks driven by internal clock sources.
- Command signals: Row and column address strobe, write enable strobe, data strobe, and data mask.
- One chip select signal and one clock enable signal.

Figure 14-3. DDR2/mDDR Memory Controller Signals



Copyright © 2013–2016, Texas Instruments Incorporated

DDR2/mDDR Memory Controller Signal Descriptions

Pin	Type[1]	Description
DDR_CLK, $\overline{\text{DDR_CLK}}$	O/Z	**Clock:** Differential clock outputs.
DDR_CKE	O/Z	**Clock enable:** Active high.
$\overline{\text{DDR_CS}}$	O/Z	**Chip select:** Active low.
$\overline{\text{DDR_WE}}$	O/Z	**Write enable strobe:** Active low, command output.
$\overline{\text{DDR_RAS}}$	O/Z	**Row address strobe:** Active low, command output.
$\overline{\text{DDR_CAS}}$	O/Z	**Column address strobe:** Active low, command output.
DDR_DQM[1:0]	O/Z	**Data mask**: Active high, output mask signal for write data.
DDR_DQS[1:0]	I/O/Z	**Data strobe:** Active high, bi-directional signals. Output with write data, input with read data.
DDR_BA[2:0]	O/Z	**Bank select:** Output, defining which bank a given command is applied.
DDR_A[13:0]	O/Z	**Address:** Address bus.
DDR_D[15:0]	I/O/Z	**Data:** Bi-directional data bus. Input for read data, output for write data.
DDR_DQGATE0	O/Z	**Strobe Enable:** Active high.
DDR_DQGATE1	I/O/Z	**Strobe Enable Delay:** Loopback signal for timing adjustment (DQS gating). Route from DDR_DQGATE0 to DDR device and back to DDR_DQGATE1 with same constraints as used for DDR clock and data.
DDR_ZP	I/O/Z	**Output drive strength reference:** Reference output for drive strength calibration of N and P channel outputs. Tie to ground via 50 ohm .5% tolerance 1/16th watt resistor (49.9 ohm .5% tolerance is acceptable).
DDR_VREF	pwr	**Voltage reference input:** Voltage reference input for the SSTL_18 I/O buffers. Note even in the case of mDDR an external resistor divider connected to this pin is necessary.

(1) Legend: I = input, O = Output, Z = high impedance, pwr = power

14.2.3 Protocol Description(s)

The DDR2/mDDR memory controller supports the DDR2/mDDR SDRAM commands listed in Table 14-1. Table 14-2 shows the signal truth table for the DDR2/mDDR SDRAM commands.

Table 14-1. DDR2/mDDR SDRAM Commands

Command	Function
ACTV	Activates the selected bank and row.
DCAB	Precharge all command. Deactivates (precharges) all banks.
DEAC	Precharge single command. Deactivates (precharges) a single bank.
DESEL	Device Deselect.
EMRS	Extended Mode Register set. Allows altering the contents of the mode register.
MRS	Mode register set. Allows altering the contents of the mode register.
NOP	No operation.
Power Down	Power-down mode.
READ	Inputs the starting column address and begins the read operation.
READ with autoprecharge	Inputs the starting column address and begins the read operation. The read operation is followed by a precharge.
REFR	Autorefresh cycle.
SLFREFR	Self-refresh mode.
WRT	Inputs the starting column address and begins the write operation.
WRT with autoprecharge	Inputs the starting column address and begins the write operation. The write operation is followed by a precharge.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 14-2. Truth Table for DDR2/mDDR SDRAM Commands

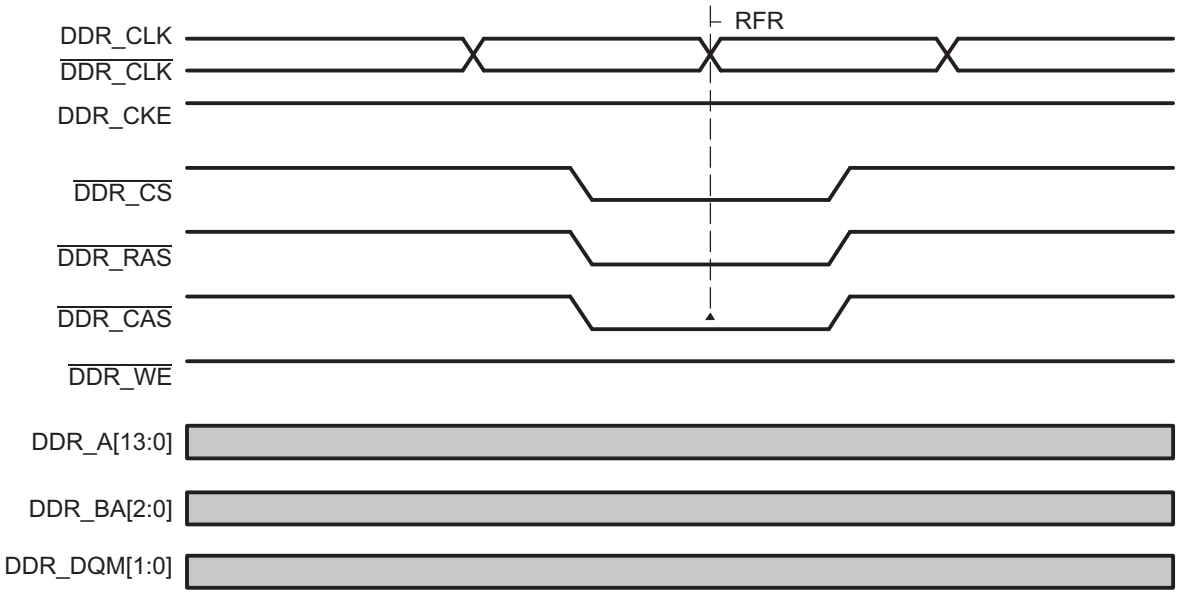
DDR2/mDDR SDRAM:	CKE		$\overline{CS}$	$\overline{RAS}$	$\overline{CAS}$	$\overline{WE}$	BA[2:0]	A[13:11, 9:0]	A10
DDR2/mDDR memory controller:	DDR_CKE Previous Cycles	DDR_CKE Current Cycle	$\overline{DDR_CS}$	$\overline{DDR_RAS}$	$\overline{DDR_CAS}$	$\overline{DDR_WE}$	DDR_BA[2:0]	DDR_A[13:11, 9:0]	DDR_A[10]
ACTV	H	H	L	L	H	H	Bank	Row Address	
DCAB	H	H	L	L	H	L	X	X	H
DEAC	H	H	L	L	H	L	Bank	X	L
MRS	H	H	L	L	L	L	BA	OP Code	
EMRS	H	H	L	L	L	L	BA	OP Code	
READ	H	H	L	H	L	H	BA	Column Address	L
READ with precharge	H	H	L	H	L	H	BA	Column Address	H
WRT	H	H	L	H	L	L	BA	Column Address	L
WRT with precharge	H	H	L	H	L	L	BA	Column Address	H
REFR	H	H	L	L	L	H	X	X	X
SLFREFR entry	H	L	L	L	L	H	X	X	X
SLFREFR exit	L	H	H	X	X	X	X	X	X
			L	H	H	H	X	X	X
NOP	H	X	L	H	H	H	X	X	X
DESEL	H	X	H	X	X	X	X	X	X
Power Down entry	H	L	H	X	X	X	X	X	X
			L	H	H	H	X	X	X
Power Down exit	L	H	H	X	X	X	X	X	X
			L	H	H	H	X	X	X

Copyright © 2013–2016, Texas Instruments Incorporated

14.2.3.1 Refresh Mode

The DDR2/mDDR memory controller issues refresh commands to the DDR2/mDDR SDRAM memory (Figure 14-4). REFR is automatically preceded by a DCAB command, ensuring the deactivation of all CE spaces and banks selected. Following the DCAB command, the DDR2/mDDR memory controller begins performing refreshes at a rate defined by the refresh rate (RR) bit in the SDRAM refresh control register (SDRCR). Page information is always invalid before and after a REFR command; thus, a refresh cycle always forces a page miss. This type of refresh cycle is often called autorefresh. Autorefresh commands may not be disabled within the DDR2/mDDR memory controller. See Section 14.2.7 for more details on REFR command scheduling.

Figure 14-4. Refresh Command

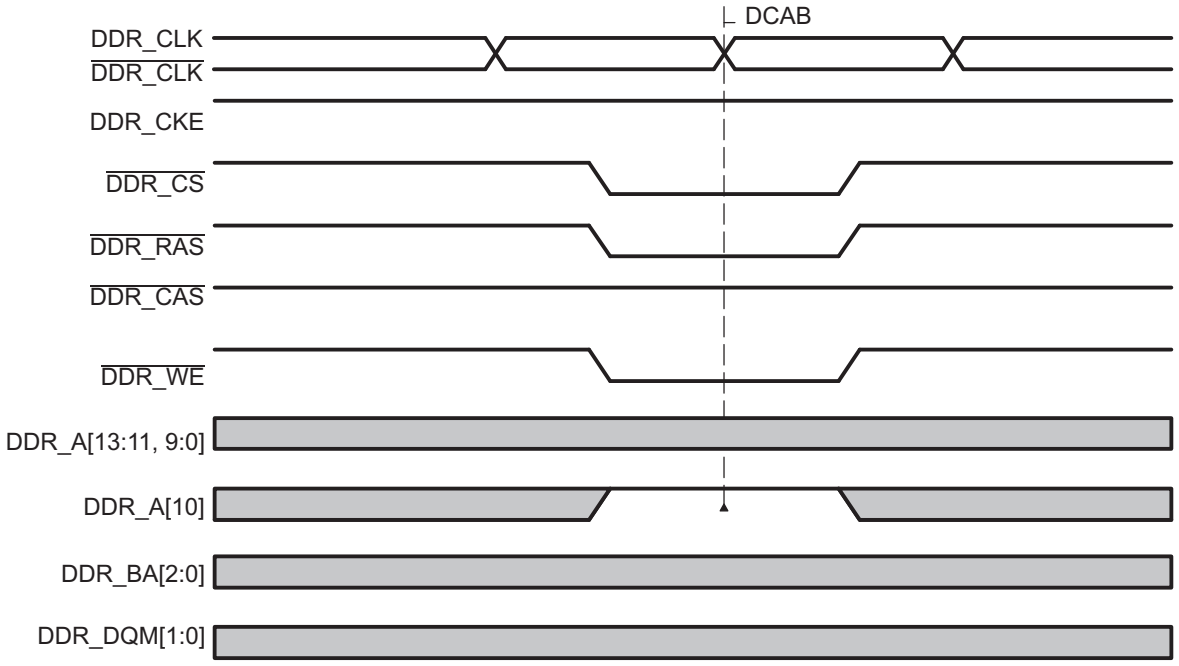


Copyright © 2013–2016, Texas Instruments Incorporated

14.2.3.2 Deactivation (DCAB and DEAC)

The precharge all banks command (DCAB) is performed after a reset to the DDR2/mDDR memory controller or following the initialization sequence. DDR2/mDDR SDRAMs also require this cycle prior to a refresh (REFR) and mode set register commands (MRS and EMRS). During a DCAB command, DDR_A[10] is driven high to ensure the deactivation of all banks. Figure 14-5 shows the timing diagram for a DCAB command.

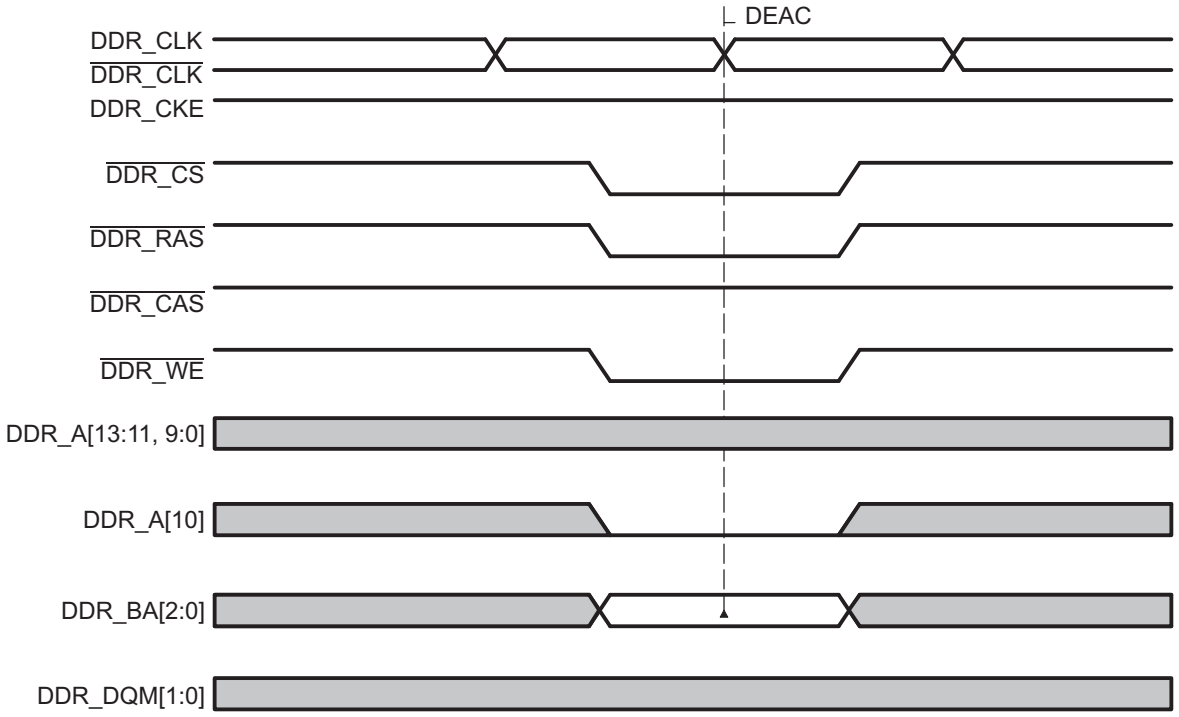
Figure 14-5. DCAB Command



Copyright © 2013–2016, Texas Instruments Incorporated

The DEAC command closes a single bank of memory specified by the bank select signals. Figure 14-6 shows the timings diagram for a DEAC command.

Figure 14-6. DEAC Command

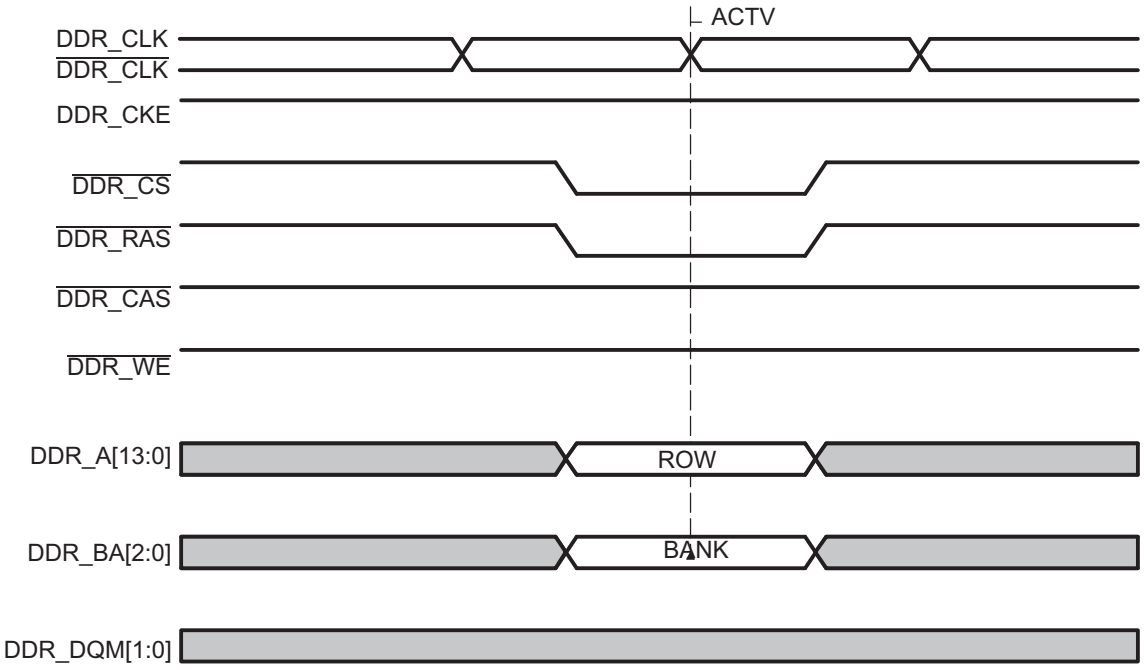


Copyright © 2013–2016, Texas Instruments Incorporated

14.2.3.3 Activation (ACTV)

The DDR2/mDDR memory controller automatically issues the activate (ACTV) command before a read or write to a closed row of memory. The ACTV command opens a row of memory, allowing future accesses (reads or writes) with minimum latency. The value of DDR_BA[2:0] selects the bank and the value of DDR_A[13:0] selects the row. When the DDR2/mDDR memory controller issues an ACTV command, a delay of t_{RCD} is incurred before a read or write command is issued. Figure 14-7 shows an example of an ACTV command. Reads or writes to the currently active row and bank of memory can achieve much higher throughput than reads or writes to random areas because every time a new row is accessed, the ACTV command must be issued and a delay of t_{RCD} incurred.

Figure 14-7. ACTV Command



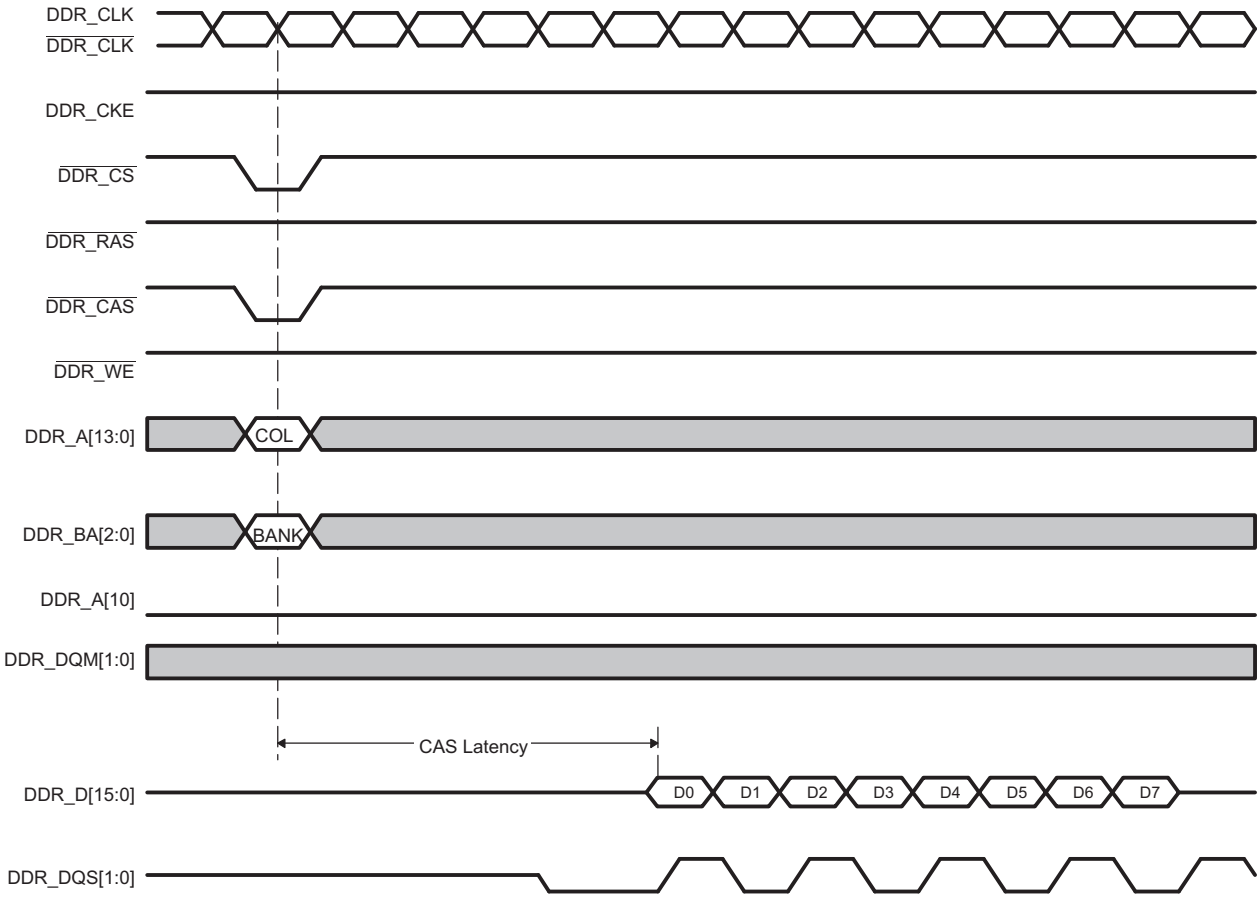
Copyright © 2013–2016, Texas Instruments Incorporated

14.2.3.4 READ Command

Figure 14-8 shows the DDR2/mDDR memory controller performing a read burst from DDR2/mDDR SDRAM. The READ command initiates a burst read operation to an active row. During the READ command, DDR_CAS drives low, DDR_WE and DDR_RAS remain high, the column address is driven on DDR_A[13:0], and the bank address is driven on DDR_BA[2:0].

The DDR2/mDDR memory controller uses a burst length of 8, and has a programmable CAS latency of 2, 3, 4, or 5. The CAS latency is three cycles in Figure 14-8. Read latency is equal to CAS latency plus additive latency. The DDR2/mDDR memory controller always configures the memory to have an additive latency of 0, so read latency equals CAS latency. Since the default burst size is 8, the DDR2/mDDR memory controller returns 8 pieces of data for every read command. If additional accesses are not pending to the DDR2/mDDR memory controller, the read burst completes and the unneeded data is disregarded. If additional accesses are pending, depending on the scheduling result, the DDR2/mDDR memory controller can terminate the read burst and start a new read burst. Furthermore, the DDR2/mDDR memory controller does not issue a DAB/DEAC command until page information becomes invalid.

Figure 14-8. DDR2/mDDR READ Command



Copyright © 2013–2016, Texas Instruments Incorporated

14.2.3.5 Write (WRT) Command

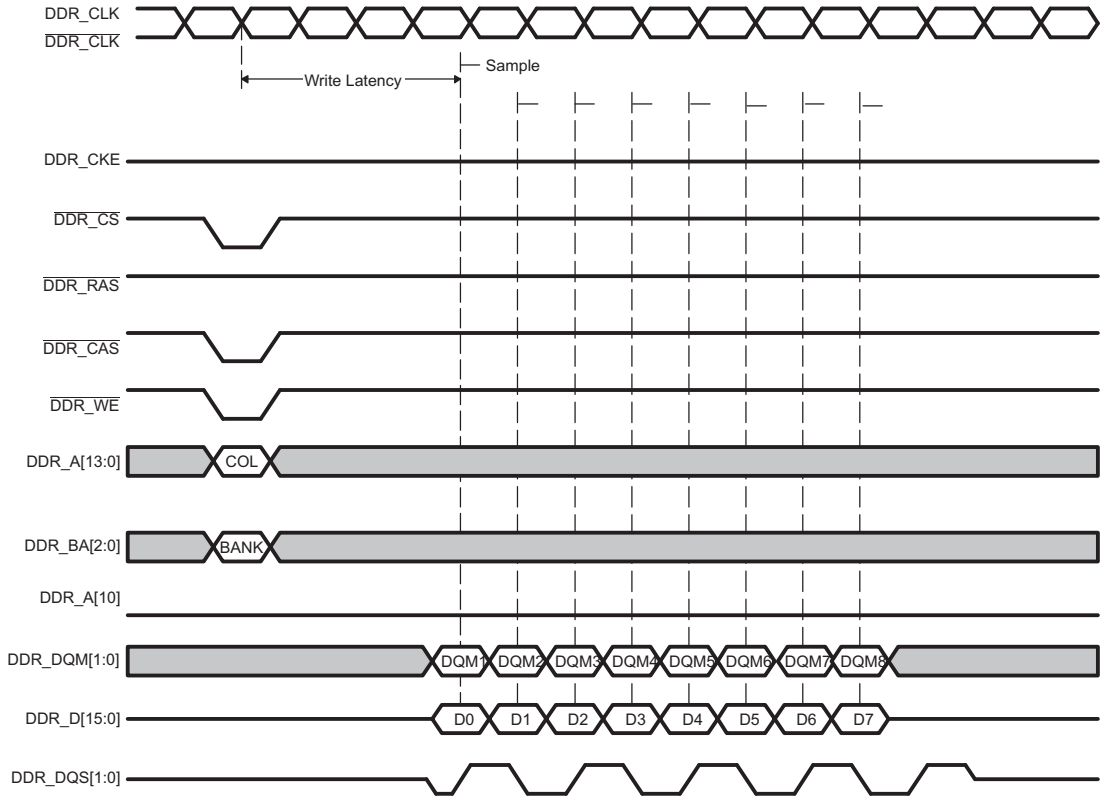
Prior to a WRT command, the desired bank and row are activated by the ACTV command. Following the WRT command, a write latency is incurred. For DDR2, write latency is equal to CAS latency minus 1 cycles. For mDDR, write latency is equal to 1 cycle, always. All writes have a burst length of 8. The use of the DDR_DQM outputs allows byte and halfword writes to be executed. Figure 14-9 shows the timing for a DDR2 write on the DDR2/mDDR memory controller.

If the transfer request is for less than 8 words, depending on the scheduling result and the pending commands, the DDR2/mDDR memory controller can:

- Mask out the additional data using DDR_DQM outputs
- Terminate the write burst and start a new write burst

The DDR2/mDDR memory controller does not perform the DEAC command until page information becomes invalid.

Figure 14-9. DDR2/mDDR WRT Command



NOTE: This diagrams shows write latency for DDR2. For mDDR, write latency is always equal to 1 cycle.

Copyright © 2013–2016, Texas Instruments Incorporated

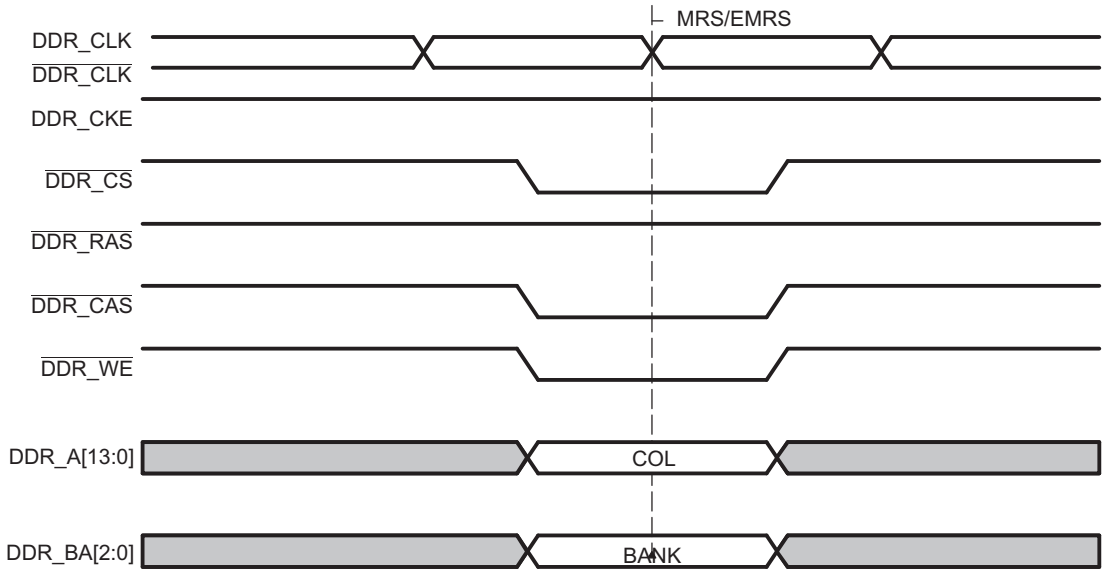
14.2.3.6 Mode Register Set (MRS and EMRS)

DDR2/mDDR SDRAM contains mode and extended mode registers that configure the DDR2/mDDR memory for operation. These registers control burst type, burst length, CAS latency, DLL enable/disable (on DDR2/mDDR device), single-ended strobe, differential strobe etc.

The DDR2/mDDR memory controller programs the mode and extended mode registers of the DDR2/mDDR memory by issuing MRS and EMRS commands. When the MRS or EMRS command is executed, the value on DDR_BA[2:0] selects the mode register to be written and the data on DDR_A[13:0] is loaded into the register. Figure 14-10 shows the timing for an MRS and EMRS command.

The DDR2/mDDR memory controller only issues MRS and EMRS commands during the DDR2/mDDR memory controller initialization sequence. See Section 14.2.13 for more information.

Figure 14-10. DDR2/mDDR MRS and EMRS Command



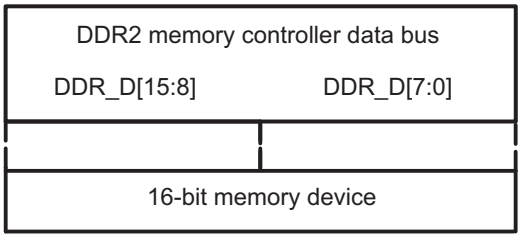
14.2.4 Memory Width and Byte Alignment

The DDR2/mDDR memory controller supports memory widths of 16 bits. Table 14-3 summarizes the addressable memory ranges on the DDR2/mDDR memory controller. Only little-endian format is supported. Figure 14-11 shows the byte lanes used on the DDR2/mDDR memory controller. The external memory is always right aligned on the data bus.

Table 14-3. Addressable Memory Ranges

Memory Width	Maximum addressable bytes per CS space	Description
×16	256 Mbytes	Halfword address

Figure 14-11. Byte Alignment



Copyright © 2013–2016, Texas Instruments Incorporated

14.2.5 Address Mapping

The memory controller views the DDR2/mDDR SDRAM device as one continuous block of memory. The memory controller receives memory access requests with a 32-bit logical address, and it uses the logical address to generate a row, column, and bank address for accessing the DDR2/mDDR SDRAM device.

The memory controller supports two address mapping schemes: normal address mapping and special address mapping. Special address mapping is typically used only with mDDR devices using partial array self-refresh.

When the internal bank position (IBANKPOS) bit in the SDRAM configuration register (SDCR) is cleared, the memory controller operates with normal address mapping. In this case, the number of column and bank address bits is determined by the IBANK and PAGESIZE fields in SDCR. The number of row address bits is determined by the number of valid address pins for the device and does not need to be set in a register.

When IBANKPOS is set to 1, the memory controller operates with special address mapping. In this case, the number of column, row, and bank address bits is determined by the PAGESIZE, ROWSIZE, and IBANK fields. The ROWSIZE field is in the SDRAM configuration register 2 (SDCR2). See Table 14-4 for a descriptions of these bit fields.

Table 14-4. Configuration Register Fields for Address Mapping

Bit Field	Bit Value	Bit Description
IBANK		Defines the number of internal banks in the external DDR2/mDDR memory.
	0	1 bank
	1h	2 banks
	2h	4 banks
	3h	8 banks
PAGESIZE		Defines the page size of each page in the external DDR2/mDDR memory.
	0	256 words (requires 8 column address bits)
	1h	512 words (requires 9 column address bits)
	2h	1024 words (requires 10 column address bits)
	3h	2048 words (requires 11 column address bits)
ROWSIZE		Defines the row size of each row in the external DDR2/mDDR memory
	0	512 (requires 9 row address bits)
	1h	1024 (requires 10 row address bits)
	2h	2048 (requires 11 row address bits)
	3h	4096 (requires 12 row address bits)
	4h	8192 (requires 13 row address bits)
	5h	16384 (requires 14 row address bits)

Copyright © 2013–2016, Texas Instruments Incorporated

14.2.5.1 Normal Address Mapping (IBANKPOS = 0)

As stated in Table 14-4, the IBANK and PAGESIZE fields of SDCR control the mapping of the logical, source address of the DDR2/mDDR memory controller to the DDR2/mDDR SDRAM row, column, and bank address bits. The DDR2/mDDR memory controller logical address always contains up to 14 row address bits, whereas the number of column and bank bits are determined by the IBANK and PAGESIZE fields. Table 14-5 show how the logical address bits map to the DDR2/mDDR SDRAM row, column, and bank bits for combinations of IBANK and PAGESIZE values. The same DDR2/mDDR memory controller pins provide the row and column address to the DDR2/mDDR SDRAM, thus the DDR2/mDDR memory controller appropriately shifts the address during row and column address selection.

Logical Address-to-DDR2/mDDR SDRAM Address Map shows how this address-mapping scheme organizes the DDR2/mDDR SDRAM rows, columns, and banks into the device memory-map. Note that during a linear access, the DDR2/mDDR memory controller increments the column address as the logical address increments. When the DDR2/mDDR memory controller reaches a page/row boundary, it moves onto the same page/row in the next bank. This movement continues until the same page has been accessed in all banks. To the DDR2/mDDR SDRAM, this process looks as shown in Figure 14-12.

By traversing across banks while remaining on the same row/page, the DDR2/mDDR memory controller maximizes the number of activated banks for a linear access. This results in the maximum number of open pages when performing a linear access being equal to the number of banks. Note that the DDR2/mDDR memory controller never opens more than one page per bank.

Ending the current access is not a condition that forces the active DDR2/mDDR SDRAM row to be closed. The DDR2/mDDR memory controller leaves the active row open until it becomes necessary to close it. This decreases the deactivate-reactivate overhead.

Table 14-5. Logical Address-to-DDR2/mDDR SDRAM Address Map for 16-bit SDRAM

SDCR Bit		Logical Address																		
IBANK	PAGESIZE	31	30	29	28	27	26	25	24	23	22	21:15	14	13	12	11	10	9	8:1	0
0	0	-									nrb=14								ncb=8	
1	0	-								nrb=14								nbb=1	ncb=8	
2h	0	-							nrb=14								nbb=2		ncb=8	
3h	0	-						nrb=14								nbb=3			ncb=8	
0	1	-								nrb=14								ncb=9		
1	1	-							nrb=14								nbb=1	ncb=9		
2h	1	-						nrb=14								nbb=2		ncb=9		
3h	1	-					nrb=14								nbb=3			ncb=9		
0	2h	-							nrb=14								ncb=10			
1	2h	-						nrb=14								nbb=1	ncb=10			
2h	2h	-					nrb=14								nbb=2		ncb=10			
3h	2h	-				nrb=14								nbb=3			ncb=10			
0	3h	-						nrb=14								ncb=11				
1	3h	-					nrb=14								nbb=1	ncb=11				
2h	3h	-				nrb=14								nbb=2		ncb=11				
3h	3h	-			nrb=14								nbb=3			ncb=11				

Copyright © 2013–2016, Texas Instruments Incorporated

Logical Address-to-DDR2/mDDR SDRAM Address Map

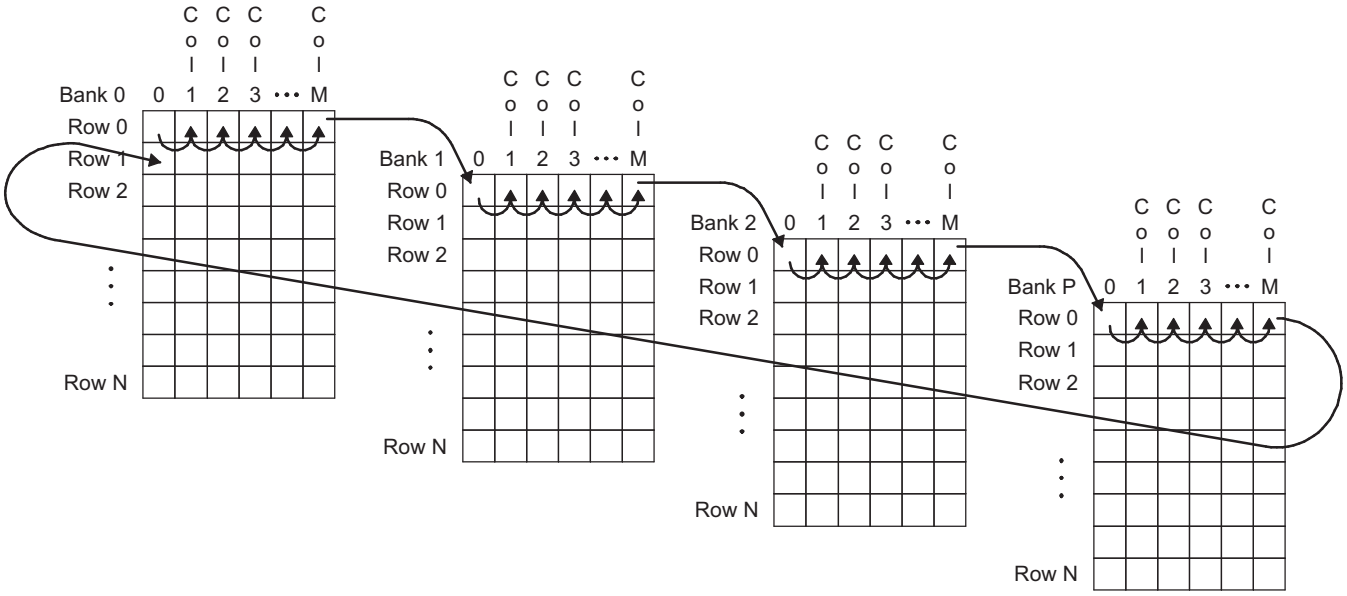
Col. 0	Col. 1	Col. 2	Col. 3	Col. 4	• • •	Col. M−1	Col. M	
					• • •			Row 0, bank 0
					• • •			Row 0, bank 1
					• • •			Row 0, bank 2
•	•	•	•	•	• • •	•	•	•
•	•	•	•	•	• • •	•	•	•
•	•	•	•	•	• • •	•	•	•
					• • •			Row 0, bank P
					• • •			Row 1, bank 0
					• • •			Row 1, bank 1
					• • •			Row 1, bank 2
•	•	•	•	•	• • •	•	•	•
•	•	•	•	•	• • •	•	•	•
•	•	•	•	•	• • •	•	•	•
					• • •			Row 1, bank P
					• • •			•
					• • •			•
					• • •			•
					• • •			Row N, bank 0
					• • •			Row N, bank 1
					• • •			Row N, bank 2
•	•	•	•	•	• • •	•	•	•
•	•	•	•	•	• • •	•	•	•
•	•	•	•	•	• • •	•	•	•
					• • •			Row N, bank P

NOTE: M is number of columns (as determined by PAGESIZE) minus 1, P is number of banks (as determined by IBANK) minus 1, and N is number of rows (as determined by both PAGESIZE and IBANK) minus 1.

Copyright © 2013–2016, Texas Instruments Incorporated

Logical Address-to-DDR2/mDDR SDRAM Address Map (continued)

Figure 14-12. DDR2/mDDR SDRAM Column, Row, and Bank Access



NOTE: M is number of columns (as determined by PAGESIZE) minus 1, P is number of banks (as determined by IBANK) minus 1, and N is number of rows (as determined by both PAGESIZE and IBANK) minus 1.

14.2.5.2 Special Address Mapping (IBANKPOS = 1)

When the internal bank position (IBANKPOS) bit is set to 1, the PAGESIZE, ROWSIZE, and IBANK fields control the mapping of the logical source address of the memory controller to the column, row, and bank address bits of the SDRAM device. Table 14-6 shows which source address bits map to the SDRAM column, row, and bank address bits for all combinations of PAGESIZE, ROWSIZE, and IBANK.

When IBANKPOS is set to 1, the effect of the address-mapping scheme is that as the source address increments across an SDRAM page boundary, the memory controller proceeds to the next page in the same bank. This movement along the same bank continues until all the pages have been accessed in the same bank. The memory controller then proceeds to the next bank in the device. This sequence is shown in Figure 14-13 and Figure 14-14.

Since, in this address mapping scheme, the memory controller can keep only one bank open, this scheme is lower in performance than the case when IBANKPOS is cleared to 0. Therefore, this case is only recommended to be used with Partial Array Self-refresh for mDDR SDRAM where performance may be traded-off for power savings.

Table 14-6. Address Mapping Diagram for 16-Bit SDRAM (IBANKPOS = 1)

31	Source Address	1
Bank Address	**Row Address**	**Column Address**
Number of bank bits is defined by IBANK nbb = 1, 2, or 3	Number of row bits is defined by ROWSIZE: nrb = 9, 10, 11, 12, 13, or 14	Number of column bits is defined by PAGESIZE: ncb = 8, 9, 10, or 11

Copyright © 2013–2016, Texas Instruments Incorporated

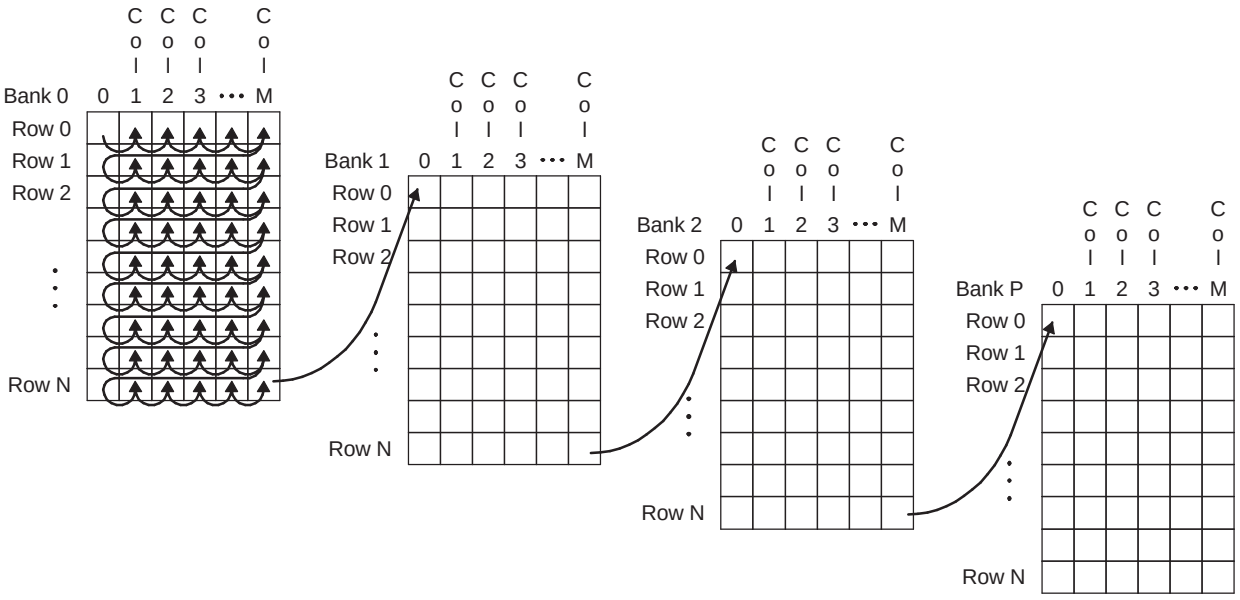
Figure 14-13. Address Mapping Diagram (IBANKPOS = 1)

Col. 0	Col. 1	Col. 2	Col. 3	Col. 4	• • •	Col. M−1	Col. M	
					• • •			Row 1, bank 0
					• • •			Row 2, bank 0
					• • •			Row 3, bank 0
•	•	•	•	•	• • •	•	•	•
•	•	•	•	•	• • •	•	•	•
•	•	•	•	•	• • •	•	•	•
					• • •			Row N, bank 0
					• • •			Row 1, bank 1
					• • •			Row 2, bank 1
					• • •			Row 3, bank 1
•	•	•	•	•	• • •	•	•	•
•	•	•	•	•	• • •	•	•	•
•	•	•	•	•	• • •	•	•	•
					• • •			Row N, bank 1
					• • •			•
					• • •			•
					• • •			•
					• • •			Row 1, bank P
					• • •			Row 2, bank P
					• • •			Row 3, bank P
•	•	•	•	•	• • •	•	•	•
•	•	•	•	•	• • •	•	•	•
•	•	•	•	•	• • •	•	•	•
					• • •			Row N, bank P

NOTE: M is number of columns (as determined by PAGESIZE) minus 1, P is number of banks (as determined by IBANK) minus 1, and N is number of rows (as determined by ROWSIZE) minus 1.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 14-14. SDRAM Column, Row, Bank Access (IBANKPOS = 1)



NOTE: M is number of columns (as determined by PAGESIZE) minus 1, P is number of banks (as determined by IBANK) minus 1, and N is number of rows (as determined by ROWSIZE) minus 1.

14.2.6 DDR2/mDDR Memory Controller Interface

To move data efficiently from on-chip resources to external DDR2/mDDR SDRAM memory, the DDR2/mDDR memory controller makes use of a command FIFO, a write FIFO, a read FIFO, and command and data schedulers. Table 14-7 describes the purpose of each FIFO.

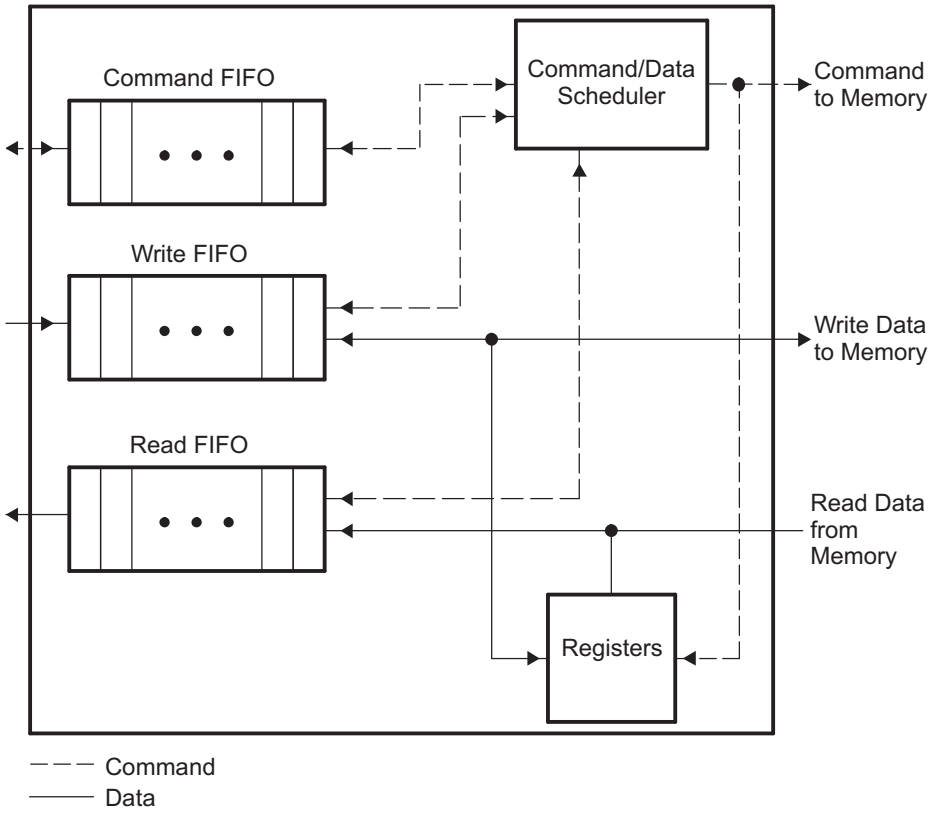
Figure 14-15 shows the block diagram of the DDR2/mDDR memory controller FIFOs. Commands, write data, and read data arrive at the DDR2/mDDR memory controller parallel to each other. The same peripheral bus is used to write and read data from external memory as well as internal memory-mapped registers.

Table 14-7. DDR2/mDDR Memory Controller FIFO Description

FIFO	Description	Depth (64-bit doublewords)
Command	Stores all commands coming from on-chip requestors	7
Write	Stores write data coming from on-chip requestors to memory	11
Read	Stores read data coming from memory to on-chip requestors	17

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 14-15. DDR2/mDDR Memory Controller FIFO Block Diagram



14.2.6.1 Command Ordering and Scheduling, Advanced Concept

The DDR2/mDDR memory controller performs command re-ordering and scheduling in an attempt to achieve efficient transfers with maximum throughput. The goal is to maximize the utilization of the data, address, and command buses while hiding the overhead of opening and closing DDR2/mDDR SDRAM rows. Command re-ordering takes place within the command FIFO.

Typically, a given master issues commands on a single priority. EDMA transfer controller read and write ports are different masters. The DDR2/mDDR memory controller first reorders commands from each master based on the following rules:

- Selects the oldest command (first command in the queue)
- Selects a read before a write if:
  - The read is to a different block address (2048 bytes) than the write
  - The read has greater or equal priority

The second bullet above may be viewed as an exception to the first bullet. This means that for an individual master, all of its commands will complete from oldest to newest, with the exception that a read may be advanced ahead of an older, lower or equal priority write. Following this scheduling, each master may have one command ready for execution.

Next, the DDR2/mDDR memory controller examines each of the commands selected by the individual masters and performs the following reordering:

- Among all pending reads, selects reads to rows already open. Among all pending writes, selects writes to rows already open.
- Selects the highest priority command from pending reads and writes to open rows. If multiple commands have the highest priority, then the DDR2/mDDR memory controller selects the oldest command.

Copyright © 2013–2016, Texas Instruments Incorporated

The DDR2/mDDR memory controller may now have a final read and write command. If the Read FIFO is not full, then the read command will be performed before the write command, otherwise the write command will be performed first.

Besides commands received from on-chip resources, the DDR2/mDDR memory controller also issues refresh commands. The DDR2/mDDR memory controller attempts to delay refresh commands as long as possible to maximize performance while meeting the SDRAM refresh requirements. As the DDR2/mDDR memory controller issues read, write, and refresh commands to DDR2/mDDR SDRAM memory, it adheres to the following rules:

1. Refresh request resulting from the Refresh Must level of urgency being reached
2. Read request without a higher priority write (selected from above reordering algorithm)
3. Refresh request resulting from the Refresh Need level of urgency being reached
4. Write request (selected from above reordering algorithm)
5. Refresh request resulting from Refresh May level of urgency being reached
6. Request to enter self-refresh mode

The following results from the above scheduling algorithm:

- All writes from a single master will complete in order
- All reads from a single master will complete in order
- From the same master, any read to the same location (or within 2048 bytes) as a previous write will complete in order

14.2.6.2 Command Starvation

The reordering and scheduling rules listed above may lead to command starvation, which is the prevention of certain commands from being processed by the DDR2/mDDR memory controller. Command starvation results from the following conditions:

- A continuous stream of high-priority read commands can block a low-priority write command
- A continuous stream of DDR2/mDDR SDRAM commands to a row in an open bank can block commands to the closed row in the same bank.

To avoid these conditions, the DDR2/mDDR memory controller can momentarily raises the priority of the oldest command in the command FIFO after a set number of transfers have been made. The PR_OLD_COUNT bit in the peripheral bus burst priority register (PBBPR) sets the number of the transfers that must be made before the DDR2/mDDR memory controller will raise the priority of the oldest command.

14.2.6.3 Possible Race Condition

A race condition may exist when certain masters write data to the DDR2/mDDR memory controller. For example, if master A passes a software message via a buffer in DDR2/mDDR memory and does not wait for indication that the write completes, when master B attempts to read the software message it may read stale data and therefore receive an incorrect message. In order to confirm that a write from master A has landed before a read from master B is performed, master A must wait for the write completion status from the DDR2/mDDR memory controller before indicating to master B that the data is ready to be read. If master A does not wait for indication that a write is complete, it must perform the following workaround:

1. Perform the required write.
2. Perform a dummy write to the DDR2/mDDR memory controller SDRAM status register.
3. Perform a dummy read to the DDR2/mDDR memory controller SDRAM status register.
4. Indicate to master B that the data is ready to be read after completion of the read in step 3. The completion of the read in step 3 ensures that the previous write was done.

The EDMA peripheral does not need to implement the above workaround. The above workaround is required for all other peripherals. See your device-specific data manual for more information.

Copyright © 2013–2016, Texas Instruments Incorporated

14.2.7 Refresh Scheduling

The DDR2/mDDR memory controller issues autorefresh (REFR) commands to DDR2/mDDR SDRAM devices at a rate defined in the refresh rate (RR) bit field in the SDRAM refresh control register (SDRCR). A refresh interval counter is loaded with the value of the RR bit field and decrements by 1 each cycle until it reaches zero. Once the interval counter reaches zero, it reloads with the value of the RR bit. Each time the interval counter expires, a refresh backlog counter increments by 1. Conversely, each time the DDR2/mDDR memory controller performs a REFR command, the backlog counter decrements by 1. This means the refresh backlog counter records the number of REFR commands the DDR2/mDDR memory controller currently has outstanding.

The DDR2/mDDR memory controller issues REFR commands based on the level of urgency. The level of urgency is defined in Table 14-8. Whenever the refresh must level of urgency is reached, the DDR2/mDDR memory controller issues a REFR command before servicing any new memory access requests. Following a REFR command, the DDR2/mDDR memory controller waits T_RFC cycles, defined in the SDRAM timing register 1 (SDTIMR1), before rechecking the refresh urgency level.

In addition to the refresh counter previously mentioned, a separate backlog counter ensures the interval between two REFR commands does not exceed 8× the refresh rate. This backlog counter increments by 1 each time the interval counter expires and resets to zero when the DDR2/mDDR memory controller issues a REFR command. When this backlog counter is greater than 7, the DDR2/mDDR memory controller issues four REFR commands before servicing any new memory requests.

The refresh counters do not operate when the DDR2/mDDR memory is in self-refresh mode.

Table 14-8. Refresh Urgency Levels

Urgency Level	Description
Refresh May	Backlog count is greater than 0. Indicates there is a backlog of REFR commands, when the DDR2/mDDR memory controller is not busy it will issue the REFR command.
Refresh Release	Backlog count is greater than 3. Indicates the level at which enough REFR commands have been performed and the DDR2/mDDR memory controller may service new memory access requests.
Refresh Need	Backlog count is greater than 7. Indicates the DDR2/mDDR memory controller should raise the priority level of a REFR command above servicing a new memory access.
Refresh Must	Backlog count is greater than 11. Indicates the level at which the DDR2/mDDR memory controller should perform a REFR command before servicing new memory access requests.

14.2.8 Self-Refresh Mode

Clearing the self refresh/low power (SR_PD) bit to 0 and then setting the low power mode enable (LPMODEN) bit to 1 in the SDRAM refresh control register (SDRCR) , forces the DDR2/mDDR memory controller to place the external DDR2/mDDR SDRAM in a low-power mode (self refresh), in which the DDR2/mDDR SDRAM maintains valid data while consuming a minimal amount of power. When the LPMODEN bit is set to 1, the DDR2/mDDR memory controller continues normal operation until all outstanding memory access requests have been serviced and the refresh backlog has been cleared. At this point, all open pages of DDR2/mDDR SDRAM are closed and a self-refresh (SLFRFR) command (an autorefresh command with self refresh/low power) is issued.

The memory controller exits the self-refresh state when a memory access is received, when the LPMODEN bit in SDRCR is cleared to 0, or when the SR_PD bit in SDRCR changed to 1. While in the self-refresh state, if a request for a memory access is received, the DDR2/mDDR memory controller services the memory access request, returning to the self-refresh state upon completion. The DDR2/mDDR memory controller will not wake up from the self-refresh state (whether from a memory access request, from clearing the LPMODEN bit, or from clearing the SR_PD bit) until T_CKE + 1 cycles have expired since the self-refresh command was issued. The value of T_CKE is defined in the SDRAM timing register 2 (SDTIMR2).

In the case of DDR2, after exiting from the self-refresh state, the memory controller will not immediately start executing commands. Instead, it will wait T_SXNR + 1 clock cycles before issuing non-read/write commands and T_SXRD + 1 clock cycles before issuing read or write commands. The SDRAM timing register 2 (SDTIMR2) programs the values of T_SXNR and T_SXRD.

Copyright © 2013–2016, Texas Instruments Incorporated

In the case of mDDR, after exiting from the self-refresh state, the memory controller will not immediately start executing commands. Instead, it will wait T_SXNR+1 clock cycles and then execute auto-refresh command before issuing any other commands. The SDRAM timing register 2 (SDTIMR2) programs the value of T_SXNR.

Once in self-refresh mode, the DDR2/mDDR memory controller input clocks (VCLK and 2X_CLK) may be gated off or changed in frequency. Stable clocks must be present before exiting self-refresh mode. See Section 14.2.16 for more information describing the proper procedure to follow when shutting down DDR2/mDDR memory controller input clocks.

See Section 14.2.16.1 for a description of the self-refresh programming sequence.

14.2.9 Partial Array Self Refresh for Mobile DDR

For additional power savings during self-refresh, the partial array self-refresh (PASR) feature of the mDDR allows you to select the amount of memory that will be refreshed during self-refresh. Use the partial array self-refresh (PASR) bit field in the SDRAM configuration register 2 (SDCR2) to select the amount of memory to refresh during self-refresh. As shown in Table 14-9 you may select either 4, 2, 1, 1/2, or 1/4 bank(s). The PASR bits are loaded into the extended mode register of the mDDR device, during autoinitialization (see Section 14.2.13).

The mDDR performs bank interleaving when the internal bank position (IBANKPOS) bit in SDRAM configuration register (SDCR) is cleared to 0. Since the SDRAM banks are only partially refreshed during partial array self-refresh, it is recommended that you set IBANKPOS to 1 to avoid bank interleaving. When IBANKPOS is cleared to 0, it is the responsibility of software to move critical data into the banks that are to be refreshed during partial array self-refresh. Refer to Section 14.2.5.2 for more information on IBANKPOS and addressing mapping in general.

Table 14-9. Configuration Bit Field for Partial Array Self-refresh

Bit Field	Bit Value	Bit Description
PASR		Partial array self refresh.
	0	Refresh banks 0, 1, 2, and 3
	1h	Refresh banks 0 and 1
	2h	Refresh bank 0
	5h	Refresh 1/2 of bank 0
	6h	Refresh 1/4 of bank 0

14.2.10 Power-Down Mode

Setting the self-refresh/low power (SR_PD) bit and the low-power mode enable (LPMODEN) bit in the SDRAM refresh control register (SDRCR) to 1, forces the DDR2/mDDR memory controller to place the external DDR2 SDRAM in the power-down mode. When the LPMODEN bit is asserted, the DDR2/mDDR memory controller continues normal operation until all outstanding memory access requests have been serviced and the refresh backlog has been cleared. At this point, all open pages of DDR2 SDRAM are closed and a Power Down command (same as NOP command but driving DDR_CKE low on the same cycle) is issued.

The DDR2/mDDR memory controller exits the power-down state when a memory access is received, when a Refresh Must level is reached, when the LPMODEN bit in SDRCR is cleared to 0, or when the SR_PD bit in SDRCR changed to 0. While in the power-down state, if a request for a memory access is received, the DDR2/mDDR memory controller services the memory access request, returning to the power-down state upon completion. The DDR2/mDDR memory controller will not wake-up from the power-down state (whether from a memory access request, from reaching a Refresh Must level, from clearing the LPMODEN bit, or from clearing the SR_PD bit) until T_CKE + 1 cycles have expired since the power-down command was issued. The value of T_CKE is defined in the SDRAM timing register 2 (SDTIMR2).

Copyright © 2013–2016, Texas Instruments Incorporated

After exiting from the power-down state, the DDR2/mDDR memory controller will drive DDR_CKE high and then not immediately start executing commands. Instead, it will wait T_XP + 1 clock cycles before issuing commands. The SDRAM timing register 2 (SDTIMR2) programs the values of T_XP.

See Section 14.2.16.1 for a description of the power-down mode programming sequence.

NOTE: Power-down mode is best suited as a power savings mode when SDRAM is being used intermittently and the system requires power savings as well as a short recovery time. You may use self-refresh mode if you desire additional power savings from disabling clocks.

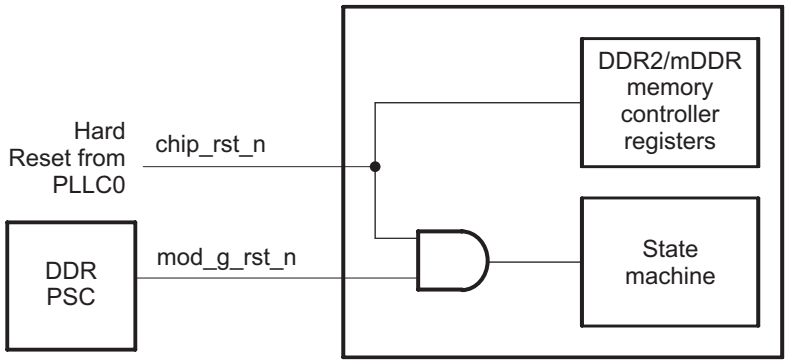
14.2.11 Reset Considerations

The DDR2/mDDR memory controller has two reset signals, chip_rst_n and mod_g_rst_n. The chip_rst_n is a module-level reset that resets both the state machine as well as the DDR2/mDDR memory controller memory-mapped registers. The mod_g_rst_n resets the state machine only; it does not reset the controller's registers, which allows soft reset (from PSC or WDT) to reset the module without resetting the configuration registers and reduces the programming overhead for setting up access to the DDR2/mDDR device. If the DDR2/mDDR memory controller is reset independently of other peripherals, the user's software should not perform memory, as well as register accesses, while chip_rst_n or mod_g_rst_n are asserted. If memory or register accesses are performed while the DDR2/mDDR memory controller is in the reset state, other masters may hang. Following the rising edge of chip_rst_n or mod_g_rst_n, the DDR2/mDDR memory controller immediately begins its initialization sequence. Command and data stored in the DDR2/mDDR memory controller FIFOs are lost. Table 14-10 describes the different methods for asserting each reset signal. The Power and Sleep Controller (PSC) acts as a master controller for power management for all of the peripherals on the device. For detailed information on power management procedures using the PSC, see the *Power and Sleep Controller (PSC)* chapter. Figure 14-16 shows the DDR2/mDDR memory controller reset diagram.

Table 14-10. Reset Sources

Reset Signal	Reset Source
chip_rst_n	Hardware/device reset
mod_g_rst_n	Power and sleep controller

Figure 14-16. DDR2/mDDR Memory Controller Reset Block Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

14.2.12 VTP IO Buffer Calibration

The DDR2/mDDR memory controller is able to control the impedance of the output IO. This feature allows the DDR2/mDDR memory controller to tune the output impedance of the IO to match that of the PCB board. Control of the output impedance of the IO is an important feature because impedance matching reduces reflections, creating a cleaner board design. Calibrating the output impedance of the IO will also reduce the power consumption of the DDR2/mDDR memory controller. The calibration is performed with respect to voltage, temperature, and process (VTP). The VTP information obtained from the calibration is used to control the output impedance of the IO.

The impedance of the output IO is selected by the value of a reference resistor connected to pin DDR_ZP. The DDR2/mDDR reference design requires the reference resistor to be a 50 ohm, 5.0% tolerance, 1/16th watt resistor (49.9 ohm, 0.5% tolerance is acceptable).

The VTP IO control register (VTPIO_CTL) is written to begin the calibration process. The VTP calibration process is described in the DDR2/mDDR initialization sequence in Section 14.2.13.1.

NOTE: VTP IO calibration must be performed following device power up and device reset. If the DDR2/mDDR memory controller is reset via the Power and Sleep Controller (PSC) and the VTP input clock is disabled, accesses to the DDR2/mDDR memory controller will not complete. To re-enable accesses to the DDR2/mDDR memory controller, enable the VTP input clock and then perform the VTP calibration sequence again.

14.2.13 Auto-Initialization Sequence

The DDR2/mDDR SDRAM contains mode and extended mode registers that configure the DDR2/mDDR memory for operation. These registers control burst type, burst length, CAS latency, DLL enable/disable (on the DDR2/mDDR device), single-ended strobe, differential strobe, etc. The DDR2/mDDR memory controller programs the mode and extended mode registers of the DDR2/mDDR memory by issuing MRS and EMRS commands during the initialization sequence. The SDRAMEN, MSDRAMEN, DDREN, and DDR2EN bits in the SDRAM configuration register (SDCR) determine if the DDR2/mDDR memory controller will perform a DDR2 or mobile DDR initialization sequence. Set these bits as follows for DDR2: SDRAMEN = 1, MSDRAMEN = 0, DDREN = 1, DDR2EN = 1. Set these bits as follow for mDDR: SDRAMEN = 1, MSDRAMEN = 1, DDREN = 1, DDR2EN = 0. The DDR2 initialization sequence performed by the DDR2/mDDR memory controller is compliant with the JESD79D-2 specification and the mDDR initialization sequence is compliant with the JESD209 specification. The DDR2/mDDR memory controller performs an initialization sequence under the following conditions:

- Following reset (rising edge of chip_rst_n or mod_g_rst_n)
- Following a write to the DDRDRIVE, CL, IBANK, or PAGESIZE bit fields in the SDRAM configuration register (SDCR)

During the initialization sequence, the memory controller issues MRS and EMRS commands that configure the DDR2/mobile DDR SDRAM mode register and extended mode register 1. The register values for DDR2 are described in Table 14-11 and Table 14-12, and the register values for mDDR are described in Table 14-13 and Table 14-14. The extended mode registers 2 and 3 are configured with a value of 0h. At the end of the initialization sequence, the memory controller performs an autorefresh cycle, leaving the memory controller in an idle state with all banks deactivated.

When a reset occurs, the DDR2/mDDR memory controller immediately begins the initialization sequence. Under this condition, commands and data stored in the DDR2/mDDR memory controller FIFOs will be lost. However, when the initialization sequence is initiated by a write to the two least-significant bytes in SDCR, data and commands stored in the DDR2/mDDR memory controller FIFOs will not be lost and the DDR2/mDDR memory controller will ensure read and write commands are completed before starting the initialization sequence.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 14-11. DDR2 SDRAM Configuration by MRS Command

Memory Controller Address Bus	Value	DDR2/mDDR SDRAM Register Bit	DDR2/mDDR SDRAM Field	Function Selection
DDR_A[12]	0	12	Power Down Exit	Fast exit
DDR_A[11:9]	t_WR	11:9	Write Recovery	Write recovery from autoprecharge. Value of 2, 3, 4, 5, or 6 is programmed based on value of the T_WR bit in the SDRAM timing register 1 (SDTIMR1).
DDR_A[8]	0	8	DLL Reset	Out of reset
DDR_A[7]	0	7	Mode: Test or Normal	Normal mode
DDR_A[6:4]	CL bit	6:4	CAS Latency	Value of 2, 3, 4, or 5 is programmed based on value of the CL bit in the SDRAM configuration register (SDCR).
DDR_A[3]	0	3	Burst Type	Sequential
DDR_A[2:0]	3h	2:0	Burst Length	Value of 8

Table 14-12. DDR2 SDRAM Configuration by EMRS(1) Command

Memory Controller Address Bus	Value	DDR2/mDDR SDRAM Register Bit	DDR2/mDDR SDRAM Field	Function Selection
DDR_A[12]	0	12	Output Buffer Enable	Output buffer enable
DDR_A[11]	0	11	RDQS Enable	RDQS disable
DDR_A[10]	1	10	$\overline{\text{DQS}}$ enable	Disables differential DQS signaling.
DDR_A[9:7]	0	9:7	OCD Calibration Program	Exit OCD calibration
DDR_A[6]	0	6	ODT Value (Rtt)	Cleared to 0 to select 75 ohms. This feature is not supported because the DDR_ODT signal is not pinned out.
DDR_A[5:3]	0	5:3	Additive Latency	0 cycles of additive latency
DDR_A[2]	1	2	ODT Value (Rtt)	Set to 1 to select 75 ohms. This feature is not supported because the DDR_ODT signal is not pinned out.
DDR_A[1]	DDRDRIVE[0]	1	Output Driver Impedance	Value of 0 or 1 is programmed based on value of DDRDRIVE0 bit in SDRAM configuration register (SDCR).
DDR_A[0]	0	0	DLL enable	DLL enable

Table 14-13. Mobile DDR SDRAM Configuration by MRS Command

Memory Controller Address Bus	Value	mDDR SDRAM Register Bit	mDDR SDRAM Field	Function Selection
DDR_A[11:7]	0	11:7	Operating mode	Normal operating mode
DDR_A[6:4]	CL bit	6:4	CAS Latency	Value of 2 or 3 is programmed based on value of CL bit in SDRAM configuration register (SDCR).
DDR_A[3]	0	3	Burst Type	Sequential
DDR_A[2:0]	3h	2:0	Burst Length	Value of 8

Copyright © 2013–2016, Texas Instruments Incorporated

Table 14-14. Mobile DDR SDRAM Configuration by EMRS(1) Command

Memory Controller Address Bus	Value	mDDR SDRAM Register Bit	mDDR SDRAM Field	Function Selection
DDR_A[11:7]	0	11:7	Operating Mode	Normal operating mode
DDR_A[6:5]	DDRDRIVE[1:0]	6:5	Output Driver Impedance	Value of 0, 1, 2, or 3 is programmed based on value of DDRDRIVE[1:0] bits in SDRAM configuration register (SDCR).
DDR_A[4:3]	0	4:3	Temperature Compensated Self Refresh	Value of 0
DDR_A[2:0]	PASR bits	2:0	Partial Array Self Refresh	Value of 0, 1, 2, 5, or 6 is programmed based on value of PASR bits in SDRAM configuration register 2 (SDCR2).

14.2.13.1 Initializing Following Device Power Up or Reset

Following device power up or reset, the DDR2/mDDR memory controller is held in reset with the internal clocks to the module gated off. Before releasing the DDR2/mDDR memory controller from reset, the clocks to the module must be turned on. Perform the following steps when turning the clocks on and initializing the module:

1. Program PLLC1 registers to start the PLL1_SYSCLK1 (that drives 2X_CLK). For information on programming PLLC1, see the *Phase-Locked Loop Controller (PLLC)* chapter.
2. Program Power and Sleep Controller (PSC) to enable the DDR2/mDDR memory controller clock.
3. Perform VTP IO calibration:
   (a) Clear POWERDN bit in the VTP IO control register (VTPIO_CTL).
   (b) Clear LOCK bit in VTPIO_CTL.
   (c) Pulse CLKRZ bit in VTPIO_CTL:
      (i) Set CLKRZ bit and wait at least 1 VTP clock cycle (clock cycle wait can be achieved by performing a read-modify-write of VTPIO_CTL in the next step).
      (ii) Clear CLKRZ bit and wait at least 1 VTP clock cycle (clock cycle wait can be achieved by performing a read-modify-write of VTPIO_CTL in the next step).
      (iii) Set CLKRZ bit.
   (d) Poll READY bit in VTPIO_CTL until it changes to 1.
   (e) Set LOCK bit in VTPIO_CTL. VTP is locked and dynamic calibration is disabled.
   (f) Set POWERDN bit in VTPIO_CTL to save power.
4. Set IOPWRDN bit in VTPIO_CTL to allow the input receivers to save power when the PWRDNEN bit in the DDR PHY control register 1 (DRPYC1R) is set.
5. Configure DRPYC1R. All of the following steps may be done with a single register write to DRPYC1R:
   (a) Set EXT_STRBEN bit to select external DQS strobe gating.
   (b) Set PWRDNEN bit to allow the input receivers to power down when they are idle.
   (c) Program RL bit value to meet the memory data sheet specification.
6. Configure the DDR slew register (DDR_SLEW):
   (a) For DDR2, clear DDR_PDENA and CMOSEN bits.
   (b) For mDDR, set the DDR_PDENA and CMOSEN bits.
7. Set the BOOTUNLOCK bit (unlocked) in the SDRAM configuration register (SDCR).
8. Program SDCR to the desired value with BOOTUNLOCK bit cleared to 0 and TIMUNLOCK bit set to 1 (unlocked).
9. For mDDR only, program the SDRAM configuration register 2 (SDCR2) to the desired value.
10. Program the SDRAM timing register 1 (SDTIMR1) and SDRAM timing register 2 (SDTIMR2) to the desired values to meet the memory data sheet specification.
11. Clear TIMUNLOCK bit (locked) in SDCR.

Copyright © 2013–2016, Texas Instruments Incorporated

12. Program the SDRAM refresh control register (SDRCR). All of the following steps may be done with a single register write to SDRCR:
    (a) Set LPMODEN bit to enable self-refresh. This is necessary for the next two steps.
    (b) Set MCLKSTOPEN bit to enable MCLK stopping. This is necessary for the next two steps.
    (c) Clear SR_PD bit to select self-refresh. This is necessary for the next two steps.
    (d) Program RR refresh rate value to meet the memory data sheet specification.
13. Program the Power and Sleep Controller (PSC) to reset (SyncReset) the DDR2/mDDR memory controller.
14. Program the Power and Sleep Controller (PSC) to re-enable the DDR2/mDDR memory controller.
15. Clear LPMODEN and MCLKSTOPEN bits in SDRCR to disable self-refresh.
16. Configure the peripheral bus burst priority register (PBBPR) to a value lower than the default value of FFh. A lower value reduces the likelihood of prolonged command starvation for accesses made from different master/peripherals to mDDR/DDR2 memory. The optimal value should be determined based on system considerations; however, a value of 20h or 30h is sufficient for typical applications.

NOTE: Some memory data sheet timing values such as those programmed into the SDRAM timing register 1 (SDTIMR1) and SDRAM timing register 2 (SDTIMR2) may need to be relaxed in order to compensate for signal delays introduced by board layout.

14.2.14 Interrupt Support

The DDR2/mDDR memory controller supports two addressing modes, linear incrementing and cache line wrap. Upon receipt of an access request for an unsupported addressing mode, the DDR2/mDDR memory controller generates an interrupt by setting the LT bit in the interrupt raw register (IRR). The DDR2/mDDR memory controller will then treat the request as a linear incrementing request.

This interrupt is called the line trap interrupt and is the only interrupt the DDR2/mDDR memory controller supports. It is an active-high interrupt and is enabled by the LTMSET bit in the interrupt mask set register (IMSR). This interrupt is mapped to the CPU and is multiplexed with RTCINT.

14.2.15 DMA Event Support

The DDR2/mDDR memory controller is a DMA slave peripheral and therefore does not generate DMA events. Data read and write requests may be made directly by masters and by the DMA.

Copyright © 2013–2016, Texas Instruments Incorporated

14.2.16 Power Management

Power dissipation from the DDR2/mDDR memory controller may be managed by the following methods:

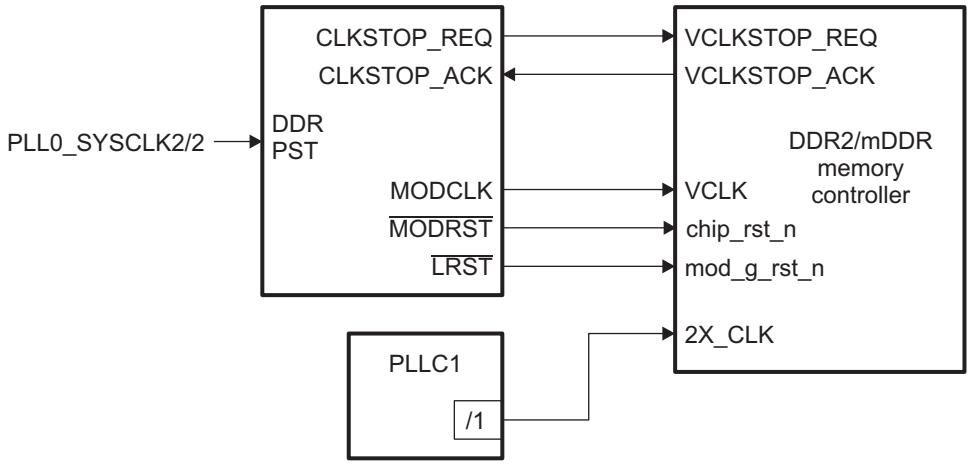
- Self-refresh mode (see Section 14.2.8)
- Power-down mode (see Section 14.2.10)
- Disabling the DDR PHY to reduce power

  The DDR2/mDDR memory controller supports low-power modes where the DLL internal to the PHY and the receivers at the I/O pins can be disabled. These functions are controlled through the DDR2/mDDR memory controller. Even if the PHY is active, the receivers can be configured to disable whenever writes are in progress and the receivers are not needed.
- Gating input clocks to the module off

  Gating input clocks off to the DDR2/mDDR memory controller achieves higher power savings when compared to the power savings of self-refresh mode and power-down mode. The input clocks are turned off outside of the DDR2/mDDR memory controller through the use of the Power and Sleep Controller (PSC) and the PLL controller 1 (PLLC1). Figure 14-17 shows the connections between the DDR2/mDDR memory controller, PSC, and PLLC1. For detailed information on power management procedures using the PSC, see the *Power and Sleep Controller (PSC)* chapter.

  Before gating clocks off, the DDR2/mDDR memory controller must place the DDR2/mDDR SDRAM memory in self-refresh mode. If the external memory requires a continuous clock, the DDR2/mDDR memory controller clock provided by PLLC1 must not be turned off because this may result in data corruption. See the following subsections for the proper procedures to follow when stopping the DDR2/mDDR memory controller clocks. Once the clocks are stopped, to re-enable the clocks follow the clock stop procedure in each respective subsection in reverse order.

Figure 14-17. DDR2/mDDR Memory Controller Power Sleep Controller Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

14.2.16.1 DDR2/mDDR Memory Controller Clock Stop Procedure

NOTE: If a data access occurs to the DDR2/mDDR memory after completing steps 1-4, the DLL will wake up and lock, then the MCLK will turn on and the access will be performed. Following steps 5 and 6, in which the clocks are disabled , all DDR2/mDDR memory accesses are not possible until the clocks are reenabled.

In power-down mode, the DDR2/mDDR memory controller input clocks (VCLK and 2X_CLK) may not be gated off. This is a limitation of the DDR2/mDDR controller. For this reason, power-down mode is best suited as a power savings mode when SDRAM is being used intermittently and the system requires power savings as well as a short recovery time. You may use self-refresh mode if you desire additional power savings from disabling clocks.

To achieve maximum power savings VCLK, MCLK, 2X_CLK, DDR_CLK, and $\overline{\text{DDR_CLK}}$ should be gated off. The procedure for clock gating is described in the following steps.

1. Allow software to complete the desired DDR transfers.
2. Change the SR_PD bit to 0 and set the LPMODEN bit to 1 in the DDR2 SDRAM refresh control register (SDRCR) to enable self-refresh mode. The DDR2/mDDR memory controller will complete any outstanding accesses and backlogged refresh cycles and then place the external DDR2/mDDR memory in self-refresh mode.
3. Set the MCLKSTOPEN bit in SDRCR to 1. This enables the DDR2/mDDR memory controller to shut off the MCLK.
4. Wait 150 CPU clock cycles to allow the MCLK to stop.
5. Program the PSC to disable the DDR2/mDDR memory controller VCLK. You must not disable VCLK in power-down mode; use only for self-refresh mode (see notes in this section).
6. For maximum power savings, the PLL/PLLC1 should be placed in bypass and powered-down mode to disable 2X_CLK. You must not disable 2X_CLK in power-down mode; use only for self-refresh mode (see notes in this section). For information on programming PLLC1, see the *Phase-Locked Loop Controller (PLLC)* chapter.

To turn clocks back on:

1. Place the PLL/PLLC1 in PLL mode to start 2X_CLK to the DDR2/mDDR memory controller.
2. Once 2X_CLK is stable, program the PSC to enable VCLK.
3. Set the RESET_PHY bit in the DDR PHY reset control register (DRPYRCR) to 1. This resets the DDR2/mDDR memory controller PHY. This bit will self-clear to 0 when reset is complete.
4. Clear the MCLKSTOPEN bit in SDRCR to 0.
5. Clear the LPMODEN bit in the DDR2 SDRAM refresh control register (SDRCR) to 0.

14.2.17 Emulation Considerations

The DDR2/mDDR memory controller will remain fully functional during emulation halts to allow emulation access to external memory.

NOTE: VTP IO calibration must be performed before emulation tools attempt to access the register or data space of the DDR2/mDDR memory controller. A bus lock-up condition will occur if the emulation tool attempts to access the register or data space of the DDR2/mDDR memory controller before completing VTP IO calibration. See Section 14.2.12 for information on VTP IO calibration.

Copyright © 2013–2016, Texas Instruments Incorporated

14.3 Supported Use Cases

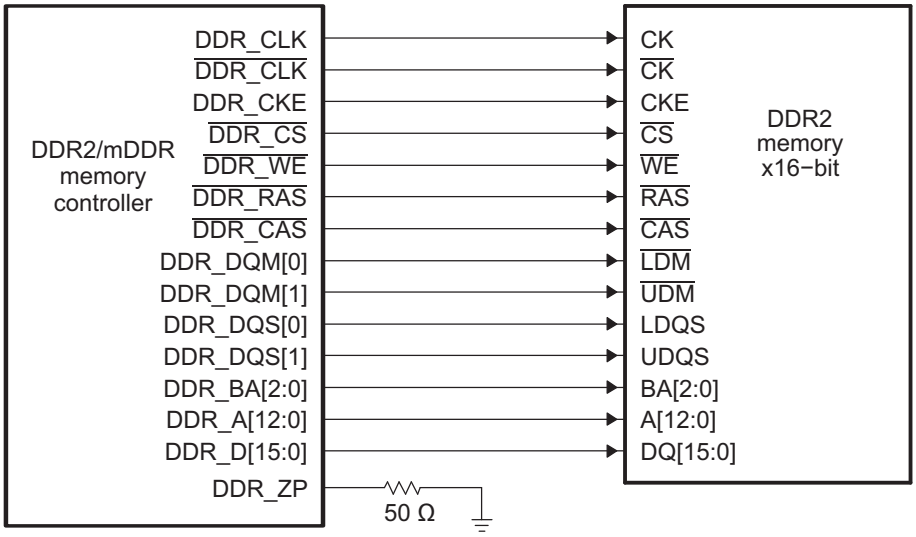
The DDR2/mDDR memory controller allows a high degree of programmability for shaping DDR2/mDDR accesses. The programmability inherent to the DDR2/mDDR memory controller provides the DDR2/mDDR memory controller with the flexibility to interface with a variety of DDR2/mDDR devices. By programming the SDRAM configuration register (SDCR), SDRAM refresh control register (SDRCR), SDRAM timing register 1 (SDTIMR1), and SDRAM timing register 2 (SDTIMR2), the DDR2/mDDR memory controller can be configured to meet the data sheet specification for DDR2 SDRAM as well as mDDR memory devices.

This section presents an example describing how to interface the DDR2 memory controller to a DDR2/mDDR-400 device. The DDR2/mDDR memory controller is assumed to be operating at 150 MHz. A similar procedure can be followed when interfacing to a mDDR memory device.

Connecting the DDR2/mDDR Memory Controller to DDR2/mDDR Memory

Figure 14-18 shows how to connect the DDR2/mDDR memory controller to a DDR2 device. Figure 14-18 displays a 16-bit interface; you can see that all signals are point-to-point connection.

Figure 14-18. Connecting DDR2/mDDR Memory Controller to a 16-Bit DDR2 Memory



Copyright © 2013–2016, Texas Instruments Incorporated

Configuring Memory-Mapped Registers to Meet DDR2 Specification

As previously stated, four memory-mapped registers must be programmed to configure the DDR2/mDDR memory controller to meet the data sheet specification of the attached DDR2/mDDR device. The registers are:

- SDRAM configuration register (SDCR)
- SDRAM refresh control register (SDRCR)
- SDRAM timing register 1 (SDTIMR1)
- SDRAM timing register 2 (SDTIMR2)

In addition to these registers, the DDR PHY control register (DRPYC1R) must also be programmed. The configuration of DRPYC1R is not dependent on the DDR2 device specification but rather on the board layout.

The following sections describe how to configure each of these registers. See Section 14.4 for more information on the DDR2/mDDR memory controller registers.

NOTE: When interfacing the DDR2/mDDR memory controller to a mDDR device, the SDRAM configuration register 2 (SDCR2) must be programmed in addition to the registers mentioned above.

Configuring SDRAM Configuration Register (SDCR)

The SDRAM configuration register (SDCR) contains register fields that configure the DDR2/mDDR memory controller to match the data bus width, CAS latency, number of banks, and page size of the attached memory. In this example, we assume the following DDR2 configuration:

- Data bus width = 16 bits
- CAS latency = 3
- Number of banks = 8
- Page size = 1024 words

Table 14-15 shows the resulting SDCR configuration. Note that the value of the TIMING_UNLOCK field is dependent on whether or not it is desirable to unlock SDTIMR1 and SDTIMR2. The TIMING_UNLOCK bit should only be set to 1 when the SDTIMR1 and SDTIMR2 needs to be updated.

Table 14-15. SDCR Configuration

Field	Value	Function Selection
TIMING_UNLOCK	x	Set to 1 to unlock the SDRAM timing register 1 and SDRAM timing register 2. Cleared to 0 to lock the SDRAM timing register 1 and SDRAM timing register 2.
NM	1h	To configure the DDR2/mDDR memory controller for a 16-bit data bus width.
CL	3h	To select a CAS latency of 3.
IBANK	3h	To select 8 internal DDR2 banks.
PAGESIZE	2h	To select 1024-word page size.

Copyright © 2013–2016, Texas Instruments Incorporated

Configuring SDRAM Refresh Control Register (SDRCR)

The SDRAM refresh control register (SDRCR) configures the DDR2/mDDR memory controller to meet the refresh requirements of the attached memory device. SDRCR also allows the DDR2/mDDR memory controller to enter and exit self refresh and enable and disable the MCLK stopping. In this example, we assume that the DDR2/mDDR memory controller is not is in self-refresh mode or power-down mode and that MCLK stopping is disabled.

The RR field in SDRCR is defined as the rate at which the attached memory device is refreshed in DDR2/mDDR cycles. The value of this field may be calculated using the following equation:

RR = DDR2/mDDR clock frequency × DDR2/mDDR memory refresh period

Table 14-16 displays the DDR2-400 refresh rate specification.

Table 14-16. DDR2 Memory Refresh Specification

Symbol	Description	Value
t_{REF}	Average Periodic Refresh Interval	7.8 µs

Therefore, the following results assuming 150 MHz DDR2/mDDR clock frequency.

RR = 150 MHz × 7.8 µs = 1170

Therefore, RR = 1170 = 492h.

Table 14-17 shows the resulting SDRCR configuration.

Table 14-17. SDRCR Configuration

Field	Value	Function Selection
LPMODEN	0	DDR2/mDDR memory controller is not in power-down mode.
MCLKSTOP_EN	0	MCLK stopping is disabled.
SR_PD	0	Leave a default value.
RR	492h	Set to 492h DDR2 clock cycles to meet the DDR2/mDDR memory refresh rate requirement.

Copyright © 2013–2016, Texas Instruments Incorporated

Configuring SDRAM Timing Registers (SDTIMR1 and SDTIMR2)

The SDRAM timing register 1 (SDTIMR1) and SDRAM timing register 2 (SDTIMR2) configure the DDR2/mDDR memory controller to meet the data sheet timing parameters of the attached memory device. Each field in SDTIMR1 and SDTIMR2 corresponds to a timing parameter in the DDR2/mDDR data sheet specification. Table 14-18 and Table 14-19 display the register field name and corresponding DDR2 data sheet parameter name along with the data sheet value. These tables also provide a formula to calculate the register field value and displays the resulting calculation. Each of the equations include a minus 1 because the register fields are defined in terms of DDR2/mDDR clock cycles minus 1. See Section 14.4.4 and Section 14.4.5 for more information.

Table 14-18. SDTIMR1 Configuration

Register Field Name	DDR2 Data Manual Parameter Name	Description	Data Manual Value (nS)	Formula (Register field must be ≥)	Register Value
T_RFC	t_{RFC}	Refresh cycle time	127.5	(t_{RFC} × $f_{DDR2/mDDR_CLK}$) - 1	19
T_RP	t_{RP}	Precharge command to refresh or activate command	15	(t_{RP} × $f_{DDR2/mDDR_CLK}$) - 1	2
T_RCD	t_{RCD}	Activate command to read/write command	15	(t_{RCD} × $f_{DDR2/mDDR_CLK}$) - 1	2
T_WR	t_{WR}	Write recovery time	15	(t_{WR} × $f_{DDR2/mDDR_CLK}$) - 1	2
T_RAS	t_{RAS}	Active to precharge command	40	(t_{RAC} × $f_{DDR2/mDDR_CLK}$) - 1	5
T_RC	t_{RC}	Activate to Activate command in the same bank	55	(t_{RC} × $f_{DDR2/mDDR_CLK}$) - 1	8
T_RRD[(1)]	t_{RRD}	Activate to Activate command in a different bank	10	((4 × t_{RRD}) + (2 × t_{CK}))/(4 × t_{CK}) - 1	1
T_WTR	t_{WTR}	Write to read command delay	10	(t_{WTR} × $f_{DDR2/mDDR_CLK}$) - 1	1

(1) The formula for the T_RRD field applies only for 8 bank DDR2/mDDR memories; when interfacing to DDR2/mDDR memories with less than 8 banks, the T_RRD field should be calculated using the following formula: (t_{RRD} × $f_{DDR2/mDDR_CLK}$) - 1.

Table 14-19. SDTIMR2 Configuration

Register Field Name	DDR2 Data Manual Parameter Name	Description	Data Manual Value	Formula (Register field must be ≥)	Register Value
T_RASMAX	t_{RAS}(MAX)	Active to precharge command	70 µs	t_{RAS}(MAX)/$_{DDR\ refresh\ rate}$ - 1	8
T_XP	t_{XP}	Exit power down to a non-read command	2(t_{CK} cycles)	If t_{XP} > t_{CKE}, then T_XP = t_{XP} - 1, else T_XP = t_{CKE} - 1	2
T_XSNR	t_{XSNR}	Exit self refresh to a non-read command	137.5 nS	(t_{XSNR} × $f_{DDR2/mDDR_CLK}$) - 1	18
T_XSRD	t_{XSRD}	Exit self refresh to a read command	200 (t_{CK} cycles)	t_{XSRD} - 1	199
T_RTP	t_{RTP}	Read to precharge command delay	15 nS	(t_{RTP} × $f_{DDR2/mDDR_CLK}$) - 1	1
T_CKE	t_{CKE}	CKE minimum pulse width	3 (t_{CK} cycles)	t_{CKE} - 1	2

Copyright © 2013–2016, Texas Instruments Incorporated

Configuring DDR PHY Control Register (DRPYC1R)

The DDR PHY control register (DRPYC1R) contains a read latency (RL) field that helps the DDR2/mDDR memory controller determine when to sample read data. The RL field should be programmed to a value equal to the CAS latency plus the round trip board delay minus 1. The minimum RL value is CAS latency plus 1 and the maximum RL value is CAS latency plus 2 (again, the RL field would be programmed to these values minus 1). Table 14-20 shows the resulting DRPYC1R configuration.

When calculating round trip board delay the signals of primary concern are the differential clock signals (DDR_CLK and $\overline{\text{DDR_CLK}}$) and data strobe signals (DDR_DQS). For these signals, calculate the round trip board delay from the DDR memory controller to the memory and then choose the maximum delay to determine the RL value. In this example, we will assume the round trip board delay is one DDR_CLK cycle; therefore, RL can be calculated as:

RL = CAS latency + round trip board delay – 1 = 4 + 1 – 1 = 4

Table 14-20. DRPYC1R Configuration

Field	Value	Function Selection
EXT_STRBEN	1h	Programs to select external strobe gating
RL	4h	Read latency is equal to CAS latency plus round trip board delay for data minus 1
PWRDNEN	0	Programmed to power up the DDR2/mDDR memory controller receivers

Copyright © 2013–2016, Texas Instruments Incorporated

14.4 Registers

Table 14-21 lists the memory-mapped registers for the DDR2/mDDR memory controller. Note that the VTP IO control register (VTPIO_CTL) resides in the System Configuration Module.

Table 14-21. DDR2/mDDR Memory Controller Registers

Address Offset	Acronym	Register Description	Section
0h	REVID	Revision ID Register	Revision ID Register (REVID)
4h	SDRSTAT	SDRAM Status Register	Section 14.4.1
8h	SDCR	SDRAM Configuration Register	Section 14.4.2
Ch	SDRCR	SDRAM Refresh Control Register	Section 14.4.3
10h	SDTIMR1	SDRAM Timing Register 1	Section 14.4.4
14h	SDTIMR2	SDRAM Timing Register 2	Section 14.4.5
1Ch	SDCR2	SDRAM Configuration Register 2	Section 14.4.6
20h	PBBPR	Peripheral Bus Burst Priority Register	Section 14.4.7
40h	PC1	Performance Counter 1 Register	Section 14.4.8
44h	PC2	Performance Counter 2 Register	Section 14.4.9
48h	PCC	Performance Counter Configuration Register	Section 14.4.10
4Ch	PCMRS	Performance Counter Master Region Select Register	Section 14.4.11
50h	PCT	Performance Counter Time Register	Performance Counter Time Register (PCT)
60h	DRPYRCR	DDR PHY Reset Control Register	Section 14.4.12
C0h	IRR	Interrupt Raw Register	Section 14.4.13
C4h	IMR	Interrupt Masked Register	Section 14.4.14
C8h	IMSR	Interrupt Mask Set Register	Section 14.4.15
CCh	IMCR	Interrupt Mask Clear Register	Section 14.4.16
E4h	DRPYC1R	DDR PHY Control Register 1	Section 14.4.17
01E2 C000h[1]	VTPIO_CTL	VTP IO Control Register	Section 10.5.19
01E2 C004h[1]	DDR_SLEW	DDR Slew Register	Section 10.5.20

(1) This register resides in the register space of the System Configuration (SYSCFG) Module. It is listed in the register space of the DDR2/mDDR controller because it is applicable to the DDR2/mDDR controller.

Revision ID Register (REVID)

The revision ID register (REVID) contains the current revision ID for the DDR2/mDDR memory controller. The REVID is shown in Figure 14-19 and described in Table 14-22.

Figure 14-19. Revision ID Register (REVID)

31	0
REV	
R-4031 1B1Fh	

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

Table 14-22. Revision ID Register (REVID) Field Descriptions

Bit	Field	Value	Description
31-0	REV	4031 1B1Fh	Revision ID value of the DDR2/mDDR memory controller.

Copyright © 2013–2016, Texas Instruments Incorporated

14.4.1 SDRAM Status Register (SDRSTAT)

The SDRAM status register (SDRSTAT) is shown in Figure 14-20 and described in Table 14-23.

Figure 14-20. SDRAM Status Register (SDRSTAT)

31	30	29	16
Rsvd	DUALCLK	Reserved	
R-0	R-1	R-0	

15	3	2	1	0
Reserved		PHYRDY	Reserved	
R-0		R-0	R-0	

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

Table 14-23. SDRAM Status Register (SDRSTAT) Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reserved
30	DUALCLK		Dual clock. Specifies whether the VCLK and MCLK inputs are asynchronous. This bit should always be read as 1.
		0	VCLK and MCLK are not asynchronous.
		1	VCLK and MCLK are asynchronous.
29-3	Reserved	0	Reserved
2	PHYRDY		DDR2/mDDR memory controller DLL ready. Specifies whether the DDR2/mDDR memory controller DLL is powered up and locked.
		0	DLL is not ready, either powered down, in reset, or not locked.
		1	DLL is powered up, locked, and ready for operation.
1-0	Reserved	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

14.4.2 SDRAM Configuration Register (SDCR)

The SDRAM configuration register (SDCR) contains fields that program the DDR2/mDDR memory controller to meet the specification of the attached DDR2/mDDR memory. These fields configure the DDR2/mDDR memory controller to match the data bus width, CAS latency, number of internal banks, and page size of the attached DDR2/mDDR memory. Writing to the DDRDRIVE[1:0], CL, IBANK, and PAGESIZE bit fields causes the DDR2/mDDR memory controller to start the DDR2/mDDR SDRAM initialization sequence. The SDCR is shown in Figure 14-21 and described in Table 14-24.

Figure 14-21. SDRAM Configuration Register (SDCR)

31	30	29	28	27	26	25	24
Reserved				DDR2TERM1	IBANK_POS	MSDRAMEN	DDRDRIVE1
R-0				R/W-1	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
BOOTUNLOCK	DDR2DDQS	DDR2TERM0	DDR2EN	DDRDLL_DIS	DDRDRIVE0	DDREN	SDRAMEN
R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-1	R/W-1
15	14	13	12	11	10	9	8
TIMUNLOCK	NM	Reserved		CL			Reserved
R/W-0	R/W-1	R-0		R/W-5h			R-0
7	6	5	4	3	2	1	0
Reserved	IBANK			Reserved	PAGESIZE		
R-0	R/W-2h			R-0	R/W-0		

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

Table 14-24. SDRAM Configuration Register (SDCR) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reserved
27	DDR2TERM1	0-3h	DDR2 termination resistor value. This bit is used in conjunction with the DDR2TERM0 bit to make a 2-bit field. This bit is writeable only when the BOOTUNLOCK bit is unlocked. See the DDR2TERM0 bit. Note that the reset value of DDR2TERM[1:0] = 10, these bits must be cleared and forced to 00 to disable the termination because the ODT feature is not supported.
26	IBANK_POS		Internal Bank position.
		0	Normal addressing
		1	Special addressing. Typically used with mobile DDR partial array self-refresh.
25	MSDRAMEN		Mobile SDRAM enable. Use this bit in conjunction with DDR2EN, DDREN, and SDRAMEN to enable/disable mobile SDRAM. To change this bit value, use the following sequence: 1. Write a 1 to the BOOTUNLOCK bit. 2. Write a 0 to the BOOTUNLOCK bit along with the desired value of the MSDRAMEN bit.
		0	Disable mobile SDRAM
		1	Enable mobile SDRAM
24	DDRDRIVE1	0-3h	SDRAM drive strength. This bit is used in conjunction with the DDRDRIVE0 bit to make a 2-bit field. This bit is writeable only when the BOOTUNLOCK bit is unlocked. See the DDRDRIVE0 bit.
23	BOOTUNLOCK		Boot Unlock. Controls the write permission settings for the DDR2TERM[1:0], MSDRAMEN, DDRDRIVE[1:0], DDR2DDQS, DDR2EN, DDRDLL_DIS, DDREN and SDRAMEN bit fields. To change these bits, use the following sequence: 1. Write a 1 to the BOOTUNLOCK bit. 2. Write a 0 to the BOOTUNLOCK bit along with the desired value of the DDR2TERM[1:0], MSDRAMEN, DDRDRIVE[1:0], DDR2DDQS, DDR2EN, DDRDLL_DIS, DDREN and SDRAMEN bits.
		0	DDR2TERM[1:0], MSDRAMEN, DDRDRIVE[1:0], DDR2DDQS, DDR2EN, DDRDLL_DIS, DDREN and SDRAMEN bit fields may not be changed.
		1	DDR2TERM[1:0], MSDRAMEN, DDRDRIVE[1:0], DDR2DDQS, DDR2EN, DDRDLL_DIS, DDREN and SDRAMEN bit fields may be changed.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 14-24. SDRAM Configuration Register (SDCR) Field Descriptions (continued)

Bit	Field	Value	Description
22	DDR2DDQS		DDR2 SDRAM differential DQS enable. This bit is writeable only when the BOOTUNLOCK bit is unlocked. To change this bit value, use the following sequence: 1. Write a 1 to the BOOTUNLOCK bit. 2. Write a 0 to the BOOTUNLOCK bit along with the desired value of the DDR2DDQS bit.
		0	Single-ended DQS
		1	Reserved
21	DDR2TERM0	0-3h	DDR2 termination resistor value. This bit is used in conjunction with the DDR2TERM1 bit to make a 2-bit field. This bit is writeable only when the BOOTUNLOCK bit is unlocked. To change this bit value, use the following sequence: 1. Write a 1 to the BOOTUNLOCK bit. 2. Write a 0 to the BOOTUNLOCK bit along with the desired value of the DDR2TERM[1:0] bits. Note that the reset value of DDR2TERM[1:0] = 10, these bits must be cleared and forced to 00 to disable the termination because the ODT feature is not supported.
		0	Disable termination
		1h-3h	Reserved
20	DDR2EN		DDR2 enable. This bit is used in conjunction with the DDREN and SDRAMEN bits to enable/disable DDR2. This bit is writeable only when the BOOTUNLOCK bit is unlocked. To change this bit value, use the following sequence: 1. Write a 1 to the BOOTUNLOCK bit. 2. Write a 0 to the BOOTUNLOCK bit along with the desired value of the DDR2EN bit.
		0	Disable DDR2
		1	Enable DDR2
19	DDRDLL_DIS		DLL disable for DDR SDRAM. This bit is writeable only when the BOOTUNLOCK bit is unlocked. To change this bit value, use the following sequence: 1. Write a 1 to the BOOTUNLOCK bit. 2. Write a 0 to the BOOTUNLOCK bit along with the desired value of the DDRDLL_DIS bit.
		0	Enable DLL
		1	Disable DLL inside DDR SDRAM
18	DDRDRIVE0	0-3h	SDRAM drive strength. This bit is used in conjunction with the DDRDRIVE1 bit to make a 2-bit field. The DDRDRIVE[1:0] bits configure the output driver impedance control value of the SDRAM memory. This bit is writeable only when the BOOTUNLOCK bit is unlocked. To change this bit value, use the following sequence: 1. Write a 1 to the BOOTUNLOCK bit. 2. Write a 0 to the BOOTUNLOCK bit along with the desired value of the DDRDRIVE[1:0] bits.
		0	For DDR2, normal drive strength. For mobile DDR, full drive strength.
		1h	For DDR2, weak drive strength. For mobile DDR, 1/2 drive strength.
		2h	For DDR2, reserved. For mobile DDR, 1/4 drive strength.
		3h	For DDR2, reserved. For mobile DDR, 1/8 drive strength.
17	DDREN		DDR enable. This bit is used in conjunction with the DDR2EN and SDRAMEN bits to enable/disable DDR. This bit is writeable only when the BOOTUNLOCK bit is unlocked. To change this bit value, use the following sequence: 1. Write a 1 to the BOOTUNLOCK bit. 2. Write a 0 to the BOOTUNLOCK bit along with the desired value of the DDREN bit.
		0	Disable DDR
		1	Enable DDR
16	SDRAMEN		SDRAM enable. This bit is used in conjunction with the DDR2EN and DDREN bits to enable/disable SDRAM. This bit is writeable only when the BOOTUNLOCK bit is unlocked. To change this bit value, use the following sequence: 1. Write a 1 to the BOOTUNLOCK bit. 2. Write a 0 to the BOOTUNLOCK bit along with the desired value of the SDRAMEN bit.
		0	Disable SDRAM
		1	Enable SDRAM

Copyright © 2013–2016, Texas Instruments Incorporated

Table 14-24. SDRAM Configuration Register (SDCR) Field Descriptions (continued)

Bit	Field	Value	Description
15	TIMUNLOCK		Timing unlock. Controls the write permission settings for the CL bit field, and the SDRAM timing register 1 (SDTIMR1) and the SDRAM timing register 2 (SDTIMR2) bit fields. To change these bits, use the following sequence: 1. Write a 1 to the TIMUNLOCK bit. 2. Write a 0 to the TIMUNLOCK bit along with the desired value of the CL bit and SDTIMR1 and SDTIMR2 bit fields.
		0	CL bit, and SDTIMR1 and SDTIMR2 bit fields may not be changed.
		1	CL bit, and SDTIMR1 and SDTIMR2 bit fields may be changed.
14	NM		SDRAM data bus width.
		0	Reserved
		1	16-bit bus width.
13-12	Reserved	0	Reserved
11-9	CL	0-7h	SDRAM CAS latency. This bit is writeable only when the TIMUNLOCK bit is unlocked. To change this bit value, use the following sequence: 1. Write a 1 to the TIMUNLOCK bit. 2. Write a 0 to the TIMUNLOCK bit along with the desired value of the CL bit.
		0-1h	Reserved
		2h	CAS Latency = 2
		3h	CAS Latency = 3
		4h	CAS Latency = 4
		5h	CAS Latency = 5
		6h-7h	Reserved
8-7	Reserved	0	Reserved
6-4	IBANK	0-7h	Internal SDRAM bank setup. Defines the number of internal banks on the external SDRAM device.
		0	1 bank
		1h	2 banks
		2h	4 banks
		3h	8 banks
		4h-7h	Reserved
3	Reserved	0	Reserved
2-0	PAGESIZE	0-7h	Page Size. Defines the page size of the SDRAM device.
		0	256-word page requiring 8 column address bits.
		1h	512-word page requiring 9 column address bits.
		2h	1024-word page requiring 10 column address bits.
		3h	2048-word page requiring 11 column address bits.
		4h-7h	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

14.4.3 SDRAM Refresh Control Register (SDRCR)

The SDRAM refresh control register (SDRCR) is used to configure the DDR2/mDDR memory controller to:

- Enter and Exit the self-refresh and power-down states.
- Enable and disable MCLK, stopping when in the self-refresh state.
- Meet the refresh requirement of the attached DDR2/mDDR device by programming the rate at which the DDR2/mDDR memory controller issues autorefresh commands.

The SDRCR is shown in Table 14-25 and described in Figure 14-22.

Figure 14-22. SDRAM Refresh Control Register (SDRCR)

31	30	29 24	23	22 16
LPMODEN	MCLKSTOPEN	Reserved	SR_PD	Reserved
R/W-0	R/W-0	R-0	R/W-0	R-0

15 0
RR
R/W-884h

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

Table 14-25. SDRAM Refresh Control Register (SDRCR) Field Descriptions

Bit	Field	Value	Description
31	LPMODEN		Low-power mode enable.
		0	Disable low-power mode.
		1	Enable low-power mode. The state of bit SR_PD selects either self-refresh or power-down mode.
30	MCLKSTOPEN		MCLK stop enable.
		0	Disables MCLK stopping, MCLK may not be stopped.
		1	Enables MCLK stopping, MCLK may be stopped. The LPMODEN bit must be set to 1 before setting the MCLKSTOPEN bit to 1.
29-24	Reserved	0	Reserved
23	SR_PD		Self-refresh or Power-down select. This bit is only in effect when the LPMODEN bit is set to 1; this bit is ignored when the LPMODEN bit is cleared to 0.
		0	Self-refresh mode.
		1	Power-down mode.
22-16	Reserved	0	Reserved
15-0	RR	0-FFFFh	Refresh rate. Defines the rate at which the attached SDRAM devices will be refreshed. The value of this field may be calculated with the following equation: *RR = SDRAM frequency/SDRAM refresh rate* where *SDRAM refresh rate* is derived from the SDRAM data sheet.

Copyright © 2013–2016, Texas Instruments Incorporated

14.4.4 SDRAM Timing Register 1 (SDTIMR1)

The SDRAM timing register 1 (SDTIMR1) configures the DDR2/mDDR memory controller to meet many of the AC timing specification of the DDR2/mDDR memory. The SDTIMR1 is programmable only when the TIMUNLOCK bit is set to 1 in the SDRAM configuration register (SDCR). Note that DDR_CLK is equal to the period of the DDR_CLK signal. See the DDR2/mDDR memory data sheet for information on the appropriate values to program each field. The SDTIMR1 is shown in Figure 14-23 and described in Table 14-26.

Figure 14-23. SDRAM Timing Register 1 (SDTIMR1)

31	25	24	22	21	19	18	16
T_RFC		T_RP		T_RCD		T_WR	
R/W-Fh		R/W-2h		R/W-2h		R/W-2h	

15	11	10	6	5	3	2	1	0
T_RAS		T_RC		T_RRD		Rsvd	T_WTR	
R/W-6h		R/W-9h		R/W-1		R-0	R/W-1	

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

Table 14-26. SDRAM Timing Register 1 (SDTIMR1) Field Descriptions

Bit	Field	Value	Description
31-25	T_RFC	0-7Fh	Specifies the minimum number of DDR_CLK cycles from a refresh or load mode command to a refresh or activate command, minus 1. Corresponds to the t_{rfc} AC timing parameter in the DDR2/mDDR data sheet. Calculate by: T_RFC = (t_{rfc}/DDR_CLK) - 1
24-22	T_RP	0-7h	Specifies the minimum number of DDR_CLK cycles from a precharge command to a refresh or activate command, minus 1. Corresponds to the t_{rp} AC timing parameter in the DDR2/mDDR data sheet. Calculate by: T_RP = (t_{rp}/DDR_CLK) - 1
21-19	T_RCD	0-7h	Specifies the minimum number of DDR_CLK cycles from an activate command to a read or write command, minus 1. Corresponds to the t_{rcd} AC timing parameter in the DDR2/mDDR data sheet. Calculate by: T_RCD = (t_{rcd}/DDR_CLK) - 1
18-16	T_WR	0-7h	Specifies the minimum number of DDR_CLK cycles from the last write transfer to a precharge command, minus 1. Corresponds to the t_{wr} AC timing parameter in the DDR2/mDDR data sheet. Calculate by: T_WR = (t_{wr}/DDR_CLK) - 1 When the value of this field is changed from its previous value, the initialization sequence will begin.
15-11	T_RAS	0-1Fh	Specifies the minimum number of DDR_CLK cycles from an activate command to a precharge command, minus 1. Corresponds to the t_{ras} AC timing parameter in the DDR2/mDDR data sheet. Calculate by: T_RAS = (t_{ras}/DDR_CLK) - 1 T_RAS must be greater than or equal to T_RCD.
10-6	T_RC	0-1Fh	Specifies the minimum number of DDR_CLK cycles from an activate command to an activate command, minus 1. Corresponds to the t_{rc} AC timing parameter in the DDR2/mDDR data sheet. Calculate by: T_RC = (t_{rc}/DDR_CLK) - 1
5-3	T_RRD	0-7h	Specifies the minimum number of DDR_CLK cycles from an activate command to an activate command in a different bank, minus 1. Corresponds to the t_{rrd} AC timing parameter in the DDR2/mDDR data sheet. Calculate by: T_RRD = (t_{rrd}/DDR_CLK) - 1 For an 8 bank DDR2/mDDR device, this field must be equal to (($4 \times t_{RRD}$) + ($2 \times t_{CK}$)) / ($4 \times t_{CK}$) - 1.
2	Reserved	0	Reserved
1-0	T_WTR	0-3h	Specifies the minimum number of DDR_CLK cycles from the last write to a read command, minus 1. Corresponds to the t_{wtr} AC timing parameter in the DDR2/mDDR data sheet. Calculate by: T_WTR = (t_{wtr}/DDR_CLK) - 1

Copyright © 2013–2016, Texas Instruments Incorporated

14.4.5 SDRAM Timing Register 2 (SDTIMR2)

Like the SDRAM timing register 1 (SDTIMR1), the SDRAM timing register 2 (SDTIMR2) also configures the DDR2/mDDR memory controller to meet the AC timing specification of the DDR2/mDDR memory. The SDTIMR2 is programmable only when the TIMUNLOCK bit is set to 1 in the SDRAM configuration register (SDCR). Note that DDR_CLK is equal to the period of the DDR_CLK signal. See the DDR2/mDDR data sheet for information on the appropriate values to program each field. SDTIMR2 is shown in Figure 14-24 and described in Table 14-27.

Figure 14-24. SDRAM Timing Register 2 (SDTIMR2)

31	30 27	26 25	24 23	22 16
Rsvd	T_RASMAX	T_XP	T_ODT	T_XSNR
R-0	R/W-8h	R/W-2h	R/W-2h	R/W-32h

15 8	7 5	4 0
T_XSRD	T_RTP	T_CKE
R/W-A7h	R/W-1	R/W-2h

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

Table 14-27. SDRAM Timing Register 2 (SDTIMR2) Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Any writes to these bit(s) must always have a value of 0.
30-27	T_RASMAX	0-Fh	Specifies the maximum number of refresh rate intervals from Activate to Precharge command. Corresponds to the t_{ras} AC timing parameter and the refresh rate in the DDR2/mDDR data sheet. Calculate by: T_RASMAX = (t_{ras}max/refresh_rate) - 1 Round down to the nearest cycle.
26-25	T_XP	0-3h	Specifies the minimum number of DDR_CLK cycles from Power Down exit to any other command except a read command, minus 1. Corresponds to the t_{xp} or t_{cke} AC timing parameter in the DDR2/mDDR data sheet. This field must satisfy the greater of t_{XP} or t_{CKE}. If $t_{xp} > t_{cke}$, then calculate by T_XP = t_{xp} - 1 If $t_{xp} < t_{cke}$, then calculate by T_XP = t_{cke} - 1
24-23	T_ODT	0-3h	Specifies the minimum number of DDR_CLK cycles from ODT enable to write data driven for DDR2 SDRAM. T_ODT must be equal to (CAS latency - tAOND -1). T_ODT must be less than CAS latency minus 1. This feature is not supported because the DDR_ODT signal is not pinned out.
22-16	T_XSNR	0-7Fh	Specifies the minimum number of DDR_CLK cycles from a self_refresh exit to any other command except a read command, minus 1. Corresponds to the t_{xsnr} AC timing parameter in the DDR2/mDDR data sheet. Calculate by: T_XSNR = (t_{xsnr}/DDR_CLK) - 1
15-8	T_XSRD	0-FFh	Specifies the minimum number of DDR_CLK cycles from a self_refresh exit to a read command, minus 1. Corresponds to the t_{xsrd} AC timing parameter in the DDR2/mDDR data sheet. Calculate by: T_XSRD = t_{xsrd} - 1
7-5	T_RTP	0-7h	Specifies the minimum number of DDR_CLK cycles from a last read command to a precharge command, minus 1. Corresponds to the t_{rtp} AC timing parameter in the DDR2/mDDR data sheet. Calculate by: T_RTP = (t_{rtp}/DDR_CLK) - 1
4-0	T_CKE	0-1Fh	Specifies the minimum number of DDR_CLK cycles between transitions on the DDR_CKE pin, minus 1. Corresponds to the t_{cke} AC timing parameter in the DDR2/mDDR data sheet. Calculate by: T_CKE = t_{cke} - 1

Copyright © 2013–2016, Texas Instruments Incorporated

14.4.6 SDRAM Configuration Register 2 (SDCR2)

The SDRAM configuration register 2 (SDCR2) contains fields to configure partial array self-refresh and rowsize of the mDDR. This register is applicable only when the IBANK_POS bit in the SDRAM configuration register (SDCR) is set to 1 for special addressing. Writing to the PASR and ROWSIZE bit fields will cause the DDR2/mDDR memory controller to start the DDR2/mDDR SDRAM initialization sequence. SDCR2 is shown in Figure 14-25 and described in Table 14-28.

Figure 14-25. SDRAM Configuration Register 2 (SDCR2)

31	19	18	16
Reserved		PASR	
R-0		R/W-0	

15	3	2	0
Reserved		ROWSIZE	
R-0		R/W-0	

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

Table 14-28. SDRAM Configuration Register 2 (SDCR2) Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reserved
18-16	PASR	0-7h	Partial array self-refresh.
		0	4 banks will be refreshed.
		1h	2 banks will be refreshed.
		2h	1 bank will be refreshed.
		3h-4h	Reserved
		5h	1/2 bank will be refreshed.
		6h	1/4 bank will be refreshed.
		7h	Reserved
15-3	Reserved	0	Reserved
2-0	ROWSIZE	0-7h	Row size. Defines the number of row address bit for DDR device.
		0	9 row address bits
		1h	10 row address bits
		2h	11 row address bits
		3h	12 row address bits
		4h	13 row address bits
		5h	14 row address bits
		6h	15 row address bits
		7h	16 row address bits

Copyright © 2013–2016, Texas Instruments Incorporated

14.4.7 Peripheral Bus Burst Priority Register (PBBPR)

The peripheral bus burst priority register (PBBPR) helps prevent command starvation within the DDR2/mDDR memory controller. To avoid command starvation, the DDR2/mDDR memory controller momentarily raises the priority of the oldest command in the command FIFO after a set number of transfers have been made. The PR_OLD_COUNT bit sets the number of transfers that must be made before the DDR2/mDDR memory controller raises the priority of the oldest command. See Section 14.2.6.2 for more details on command starvation.

Proper configuration of the PBBPR is critical to correct system operation. The DDR2/mDDR memory controller always prioritizes accesses to open rows as highest, if there is any bank conflict regardless of master priority. This is done to allow most efficient utilization of the DDR2/mDDR. However, it could lead to excessive blocking of high priority masters. If the PR_OLD_COUNT bits are cleared to 00h, then the DDR2/mDDR memory controller always honors the master priority, regardless of open row/bank status. For most systems, the PBBPR should be set to a moderately low value to provide an acceptable balance of DDR2/mDDR efficiency and latency for high priority masters (for example, 10h or 20h).

The PBBPR is shown in Figure 14-26 and described in Table 14-29.

Figure 14-26. Peripheral Bus Burst Priority Register (PBBPR)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		PR_OLD_COUNT	
R-0		R/W-FFh	

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

Table 14-29. Peripheral Bus Burst Priority Register (PBBPR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Any writes to these bit(s) must always have a value of 0.
7-0	PR_OLD_COUNT	0-FFh	Priority raise old counter. Specifies the number of memory transfers after which the DDR2/mDDR memory controller will elevate the priority of the oldest command in the command FIFO. Clearing to 00h will ensure master priority is strictly honored (at the cost of decreased DDR2/mDDR memory controller efficiency, as open row will always be closed immediately if any bank conflict occurs). Recommended setting for typical system operation is between 10h and 20h.
		0	1 memory transfer
		1h	2 memory transfers
		2h	3 memory transfers
		3h-FFh	4 to 256 memory transfers

Copyright © 2013–2016, Texas Instruments Incorporated

14.4.8 Performance Counter 1 Register (PC1)

For debug or gathering performance statistics, the PC1 and PC2 counters and associated configuration registers are provided. These are intended for debug and analysis only. By configuring the performance counter configuration register (PCC) to define the type of statistics to gather and configuring the performance counter master region select register (PCMRS) to filter accesses only to specific chip select regions, performing system applications and then reading these counters, different statistics can be gathered. To reset the counters, you must reset (mod_g_rst_n) the DDR2/mDDR memory controller through the PSC. For details on the PSC, see the *Power and Sleep Controller (PSC)* chapter.

The performance counter 1 register (PC1) is shown in Figure 14-27 and described in Table 14-30.

Figure 14-27. Performance Counter 1 Register (PC1)

31	0
Counter1	
R-0	

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

Table 14-30. Performance Counter 1 Register (PC1) Field Descriptions

Bit	Field	Value	Description
31-0	Counter1	0-FFFF FFFFh	32-bit counter that can be configured as specified in the performance counter configuration register (PCC) and the performance counter master region select register (PCMRS).

14.4.9 Performance Counter 2 Register (PC2)

The performance counter 2 register (PC2) is shown in Figure 14-28 and described in Table 14-31.

Figure 14-28. Performance Counter 2 Register (PC2)

31	0
Counter2	
R-0	

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

Table 14-31. Performance Counter 2 Register (PC2) Field Descriptions

Bit	Field	Value	Description
31-0	Counter2	0-FFFF FFFFh	32-bit counter that can be configured as specified in the performance counter configuration register (PCC) and the performance counter master region select register (PCMRS).

Copyright © 2013–2016, Texas Instruments Incorporated

14.4.10 Performance Counter Configuration Register (PCC)

The performance counter configuration register (PCC) is shown in Figure 14-29 and described in Table 14-32.

Table 14-33 shows the possible filter configurations for the two performance counters. These filter configurations can be used in conjunction with a Master ID and/or an external chip select to obtain performance statistics for a particular master and/or an external chip select.

Figure 14-29. Performance Counter Configuration Register (PCC)

31	30	29 20	19 16
CNTR2_MSTID_EN	CNTR2_REGION_EN	Reserved	CNTR2_CFG
R/W-0	R/W-0	R-0	R/W-1

15	14	13 4	3 0
CNTR1_MSTID_EN	CNTR1_REGION_EN	Reserved	CNTR1_CFG
R/W-0	R/W-0	R-0	R/W-0

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

Table 14-32. Performance Counter Configuration Register (PCC) Field Descriptions

Bit	Field	Value	Description
31	CNTR2_MSTID_EN		Master ID filter enable for performance counter 2 register (PC2). Refer to Table 14-33 for details.
		0	Master ID filter is disabled. PC2 counts accesses from all masters to DDR2/mDDR SDRAM.
		1	Master ID filter is enabled. PC2 counts accesses from the master, corresponding to the Master ID value in the MST_ID2 bit field of the performance counter master region select register (PCMRS).
30	CNTR2_REGION_EN		Chip select filter enable for performance counter 2 register (PC2). Refer to Table 14-33 for details.
		0	Chip select filter is disabled. PC2 counts total number of accesses (DDR2/mDDR SDRAM + DDR2/mDDR memory controller memory-mapped register accesses). The REGION_SEL2 bit field value in the performance counter master region select register (PCMRS) is a don't care.
		1	Chip select filter is enabled. If the REGION_SEL2 bit field value in the performance counter master region select register (PCMRS) is: **REGION_SEL2 = 0:** PC2 counts accesses to DDR2/mDDR SDRAM memory. **REGION_SEL2 = 7h:** PC2 counts accesses to DDR2/mDDR memory controller memory-mapped registers.
29-20	Reserved	0	Any writes to these bit(s) must always have a value of 0.
19-16	CNTR2_CFG	0-Fh	Filter configuration for performance counter 2 register (PC2). Refer to Table 14-33 for details.
15	CNTR1_MSTID_EN		Master ID filter enable for performance counter 1 register (PC1). Refer to Table 14-33 for details.
		0	Master ID filter is disabled. PC1 counts accesses from all masters to DDR2/mDDR SDRAM.
		1	Master ID filter is enabled. PC1 counts accesses from the master, corresponding to the Master ID value in the MST_ID1 bit field of the performance counter master region select register (PCMRS).
14	CNTR1_REGION_EN		Chip select filter enable for performance counter 1 register (PC1). Refer to Table 14-33 for details.
		0	Chip select filter is disabled. PC1 counts total number of accesses (DDR2/mDDR SDRAM + DDR2/mDDR memory controller memory-mapped register accesses). The REGION_SEL1 bit field value in the performance counter master region select register (PCMRS) is a don't care.
		1	Chip select filter is enabled. If the REGION_SEL1 bit field value in the performance counter master region select register (PCMRS) is: **REGION_SEL1 = 0:** PC1 counts accesses to DDR2/mDDR SDRAM memory. **REGION_SEL1 = 7h:** PC1 counts accesses to DDR2/mDDR memory controller memory-mapped registers.
13-4	Reserved	0	Any writes to these bit(s) must always have a value of 0.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 14-32. Performance Counter Configuration Register (PCC) Field Descriptions (continued)

Bit	Field	Value	Description
3-0	CNTR1_CFG	0-Fh	Filter configuration for performance counter 1 register (PC1). Refer to Table 14-33 for details.

Table 14-33. Performance Counter Filter Configuration

Performance Counter Configuration Register (PCC) Bit			
CNTR*n*_CFG	**CNTR*n*_REGION_EN**	**CNTR*n*_MSTID_EN**	**Description**
0	0	0 or 1	**Counts the total number of READ/WRITE commands the external memory controller receives.** The size of counter increments are determines by the size of the transfer and the default burst size (DBS). The counter breaks up transfers into sizes according to DBS. Therefore, counter increments for transfers aligned to DBS are equal to the transfer size divided by the DBS.
1h	0	0	**Counts the total number of ACTIVATE commands the external memory controller issues to DDR2/mDDR memory.** The counter increments by a value of 1 for every request to read/write data to a closed bank in DDR2/mDDR memory by the external memory controller.
2h	0 or 1	0 or 1	**Counts the total number of READ commands (read accesses) the DDR2/mDDR memory controller receives.** Counter increments for transfers aligned to the default burst size (DBS) are equal to the transfer size divided by the DBS.
3h	0 or 1	0 or 1	**Counts the total number of WRITE commands the DDR2/mDDR memory controller receives.** Counter increments for transfers aligned to the default burst size (DBS) are equal to the transfer size of data written to the DDR2/mDDR memory controller divided by the DBS.
4h	0	0	**Counts the number of external memory controller cycles (DDR_CLK cycles) that the command FIFO is full.** Use the following to calculate the counter value as a percentage: *% = counter value / total DDR_CLK cycles in a sample period* As the value of this counter approaches 100%, the DDR2/mDDR memory controller is approaching a congestion point where the command FIFO is full 100% of the time and a command will have to wait at the SCR to be accepted in the command FIFO.
5h-7h	0	0	Reserved
8h	0 or 1	0 or 1	**Counts the number of commands (requests) in the command FIFO that require a priority elevation.** To avoid command starvation, the DDR2/mDDR memory controller can momentarily raise the priority of the oldest command in the command FIFO after a set number of transfers have been made. The PR_OLD_COUNT bit field in the peripheral bus burst priority register (PBBPR) sets the number of the transfers that must be made before the DDR2/mDDR memory controller will raise the priority of the oldest command.
9h	0	0	**Counts the number of DDR2/mDDR memory controller cycles (DDR_CLK cycles) that a command is pending in the command FIFO.** This counter increments every cycle the command FIFO is not empty. Use the following to calculate the counter value as a percentage: *% = counter value / total DDR_CLK cycles in sample period* As the value of this counter approaches 100%, the number of cycles the DDR2/mDDR memory controller has a command in the command FIFO to service approaches 100%.
Ah-Fh	0	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

14.4.11 Performance Counter Master Region Select Register (PCMRS)

The performance counter master region select register (PCMRS) is shown in Figure 14-30 and described in Table 14-34.

Figure 14-30. Performance Counter Master Region Select Register (PCMRS)

31	24	23	20	19	16
MST_ID2		Reserved		REGION_SEL2	
R/W-0		R-0		R/W-0	

15	8	7	4	3	0
MST_ID1		Reserved		REGION_SEL1	
R/W-0		R-0		R/W-0	

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

Table 14-34. Performance Counter Master Region Select Register (PCMRS) Field Descriptions

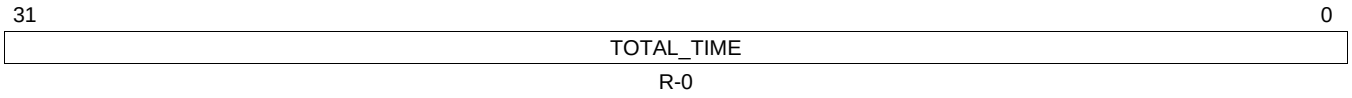
Bit	Field	Value	Description
31-24	MST_ID2	0-FFh	Master ID for performance counter 2 register (PC2). For the Master ID value for master peripherals in the device, see the *System Configuration (SYSCFG) Module* chapter.
23-20	Reserved	0	Any writes to these bit(s) must always have a value of 0.
19-16	REGION_SEL2	0-Fh	Region select for performance counter 2 register (PC2).
		0	PC2 counts total DDR2/mDDR accesses.
		1h-6h	Reserved
		7h	PC2 counts total DDR2/mDDR memory controller memory-mapped register accesses.
		8h-Fh	Reserved
15-8	MST_ID1	0-FFh	Master ID for performance counter 1 register (PC1). For the Master ID value for master peripherals in the device, see the *System Configuration (SYSCFG) Module* chapter.
7-4	Reserved	0	Any writes to these bit(s) must always have a value of 0.
3-0	REGION_SEL1	0-Fh	Region select for performance counter 1 register (PC1).
		0	PC1 counts total DDR2/mDDR accesses.
		1h-6h	Reserved
		7h	PC1 counts total DDR2/mDDR memory controller memory-mapped register accesses.
		8h-Fh	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

Performance Counter Time Register (PCT)

The performance counter time register (PCT) is shown in Figure 14-31 and described in Table 14-35.

Figure 14-31. Performance Counter Time Register (PCT)



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

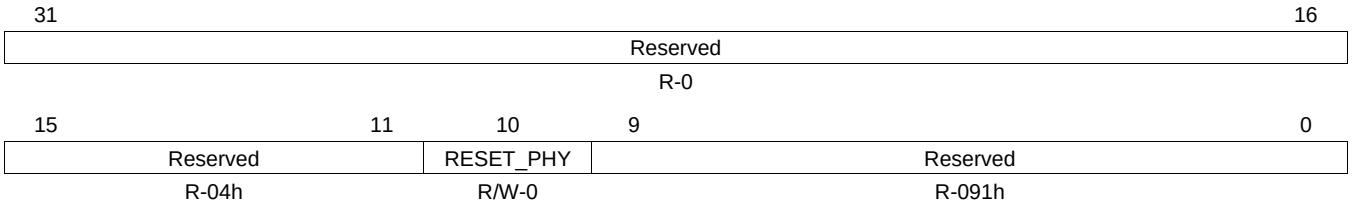
Table 14-35. Performance Counter Time Register (PCT) Field Description

Bit	Field	Value	Description
31-0	TOTAL_TIME	0-FFFF FFFFh	32-bit counter that continuously counts number for DDR_CLK cycles elapsed after the DDR2/mDDR memory controller is brought out of reset.

14.4.12 DDR PHY Reset Control Register (DRPYRCR)

The DDR PHY reset control register (DRPYRCR) is used to reset the DDR PHY. The DRPYRCR is shown in Figure 14-32 and described in Table 14-36.

Figure 14-32. DDR PHY Reset Control Register (DRPYRCR)



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

Table 14-36. DDR PHY Reset Control Register (DRPYRCR)

Bit	Field	Value	Description
31-11	Reserved	0000 04h	Always write the default value to these bits.
10	RESET_PHY		Reset DDR PHY.
		0	No effect.
		1	Resets DDR PHY.
9-0	Reserved	091h	Always write the default value to these bits.

Copyright © 2013–2016, Texas Instruments Incorporated

14.4.13 Interrupt Raw Register (IRR)

The interrupt raw register (IRR) displays the raw status of the interrupt. If the interrupt condition occurs, the corresponding bit in IRR is set independent of whether or not the interrupt is enabled. The IRR is shown in Figure 14-33 and described in Table 14-37.

Figure 14-33. Interrupt Raw Register (IRR)

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		LT	Reserved	
R-0		R/W1C-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing 0 has no effect); -*n* = value after reset

Table 14-37. Interrupt Raw Register (IRR) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	LT		Line trap. Write a 1 to clear LT and the LTM bit in the interrupt masked register (IMR); a write of 0 has no effect.
		0	A line trap condition has not occurred.
		1	Illegal memory access type. See Section 14.2.14 for more details.
1-0	Reserved	0	Reserved

14.4.14 Interrupt Masked Register (IMR)

The interrupt masked register (IMR) displays the status of the interrupt when it is enabled. If the interrupt condition occurs and the corresponding bit in the interrupt mask set register (IMSR) is set, then the IMR bit is set. The IMR bit is not set if the interrupt is not enabled in IMSR. The IMR is shown in Figure 14-34 and described in Table 14-38.

Figure 14-34. Interrupt Masked Register (IMR)

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		LTM	Reserved	
R-0		R/W1C-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing 0 has no effect); -*n* = value after reset

Table 14-38. Interrupt Masked Register (IMR) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	LTM		Line trap masked. Write a 1 to clear LTM and the LT bit in the interrupt raw register (IRR); a write of 0 has no effect.
		0	A line trap condition has not occurred.
		1	Illegal memory access type (only set if the LTMSET bit in IMSR is set). See Section 14.2.14 for more details.
1-0	Reserved	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

14.4.15 Interrupt Mask Set Register (IMSR)

The interrupt mask set register (IMSR) enables the DDR2/mDDR memory controller interrupt. The IMSR is shown in Figure 14-35 and described in Table 14-39.

NOTE: If the LTMSET bit in IMSR is set concurrently with the LTMCLR bit in the interrupt mask clear register (IMCR), the interrupt is not enabled and neither bit is set to 1.

Figure 14-35. Interrupt Mask Set Register (IMSR)

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		LTMSET	Reserved	
R-0		R/W-0	R-0	

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

Table 14-39. Interrupt Mask Set Register (IMSR) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	LTMSET		Line trap interrupt set. Write a 1 to set LTMSET and the LTMCLR bit in the interrupt mask clear register (IMCR); a write of 0 has no effect.
		0	Line trap interrupt is not enabled; a write of 1 to the LTMCLR bit in IMCR occurred.
		1	Line trap interrupt is enabled.
1-0	Reserved	0	Reserved

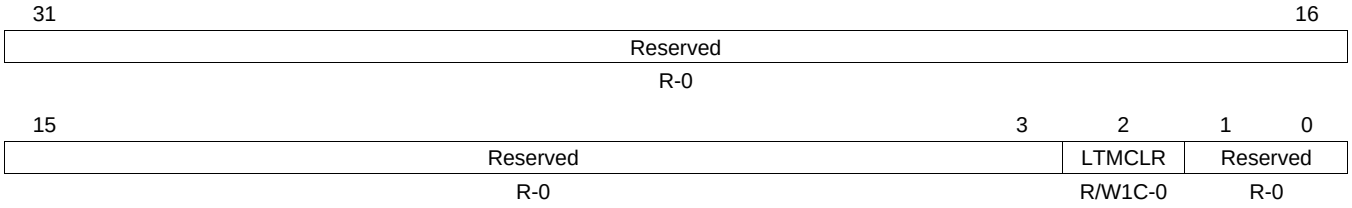
Copyright © 2013–2016, Texas Instruments Incorporated

14.4.16 Interrupt Mask Clear Register (IMCR)

The interrupt mask clear register (IMCR) disables the DDR2/mDDR memory controller interrupt. Once an interrupt is enabled, it may be disabled by writing a 1 to the IMCR bit. The IMCR is shown in Figure 14-36 and described in Table 14-40.

NOTE: If the LTMCLR bit in IMCR is set concurrently with the LTMSET bit in the interrupt mask set register (IMSR), the interrupt is not enabled and neither bit is set to 1.

Figure 14-36. Interrupt Mask Clear Register (IMCR)



LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing 0 has no effect); -*n* = value after reset

Table 14-40. Interrupt Mask Clear Register (IMCR) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	LTMCLR		Line trap interrupt clear. Write a 1 to clear LTMCLR and the LTMSET bit in the interrupt mask set register (IMSR); a write of 0 has no effect.
		0	Line trap interrupt is not enabled.
		1	Line trap interrupt is enabled; a write of 1 to the LTMSET bit in IMSR occurred.
1-0	Reserved	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

14.4.17 DDR PHY Control Register (DRPYC1R)

The DDR PHY control register 1 (DRPYC1R) configures the DDR2/mDDR memory controller read latency. The DRPYC1R is shown in Figure 14-37 and described in Table 14-41.

Figure 14-37. DDR PHY Control Register 1 (DRPYC1R)

31	16
Reserved	
R-0	

15	14 12	11 8	7	6	5 3	2 0
Rsvd	CONFIG_DLL_MODE	Reserved	EXT_STRBEN	PWRDNEN	Reserved	RL
R-0	R/W-0	R-0	R/W-0	R/W-1	R-0	R/W-6h

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

Table 14-41. DDR PHY Control Register 1 (DRPYC1R) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved	0	Reserved
14-12	CONFIG_DLL_MODE		DLL configuration. Controls the value assigned to the config_dll_mode input.
		0-1h	DLL REFCLK is enabled.
		2h	DLL REFCLK is disabled.
		3h-7h	Reserved
11-8	Reserved	0	Reserved
7	EXT_STRBEN		Internal/External strobe gating.
		0	Internal strobe gating mode.
		1	External strobe gating mode.
6	PWRDNEN		Power down receivers.
		0	Receivers powered up when idle.
		1	Receivers powered down when idle.
5-3	Reserved	0	Reserved
2-0	RL	0-7h	Read latency. Read latency is equal to CAS latency plus round trip board delay for data minus 1. The maximum value of read latency that is supported is CAS latency plus 2. The minimum read latency value that is supported is CAS latency plus 1. The read latency value is defined in number of MCLK/DDR_CLK cycles.

Copyright © 2013–2016, Texas Instruments Incorporated

Enhanced Capture (eCAP) Module

The enhanced capture (eCAP) module is essential in systems where accurate timing of external events is important. This chapter describes the eCAP module.

Topic **Page**

15.1 Introduction **409**
15.2 Architecture **410**
15.3 Applications **419**
15.4 Registers **435**

Copyright © 2013–2016, Texas Instruments Incorporated

15.1 Introduction

15.1.1 Purpose of the Peripheral

Uses for eCAP include:

- Sample rate measurements of audio inputs
- 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

15.1.2 Features

The eCAP module includes the following features:

- 32-bit time base counter
- 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

Copyright © 2013–2016, Texas Instruments Incorporated

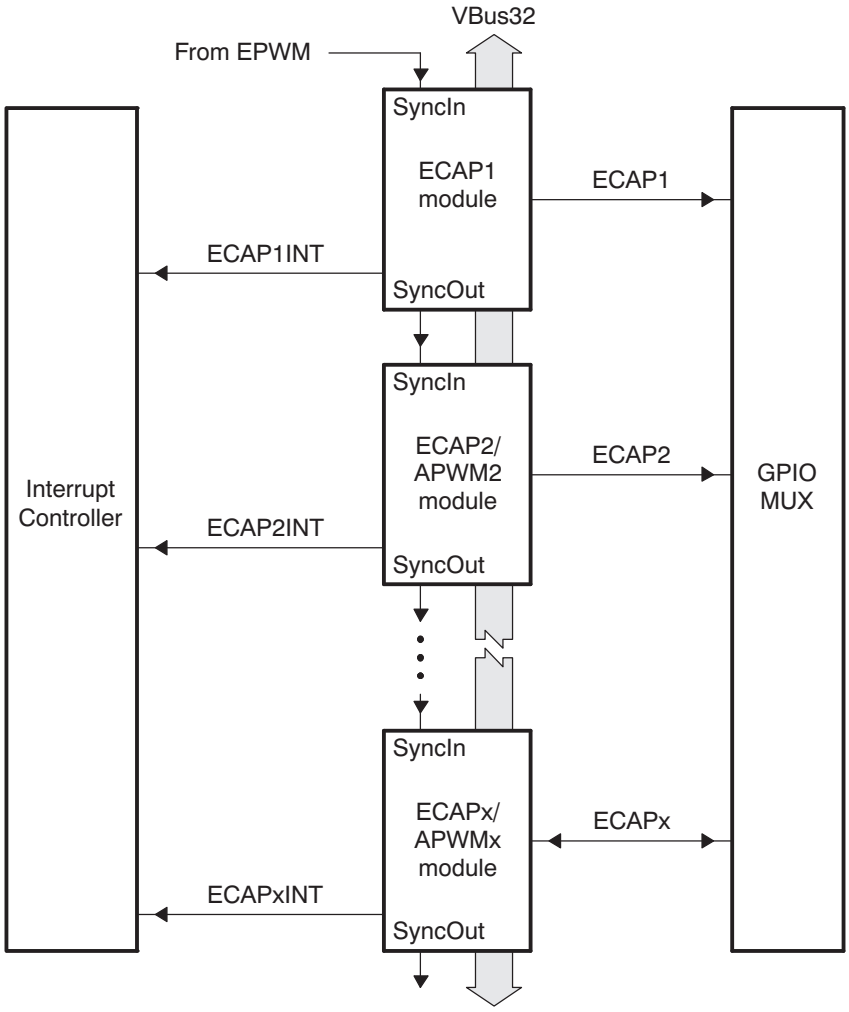
15.2 Architecture

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 × 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

Multiple identical eCAP modules can be contained in a system as shown in Figure 15-1. The number of modules is device-dependent and is based on target application needs. In this chapter, the letter x within a signal or module name is used to indicate a generic eCAP instance on a device.

Figure 15-1. Multiple eCAP Modules

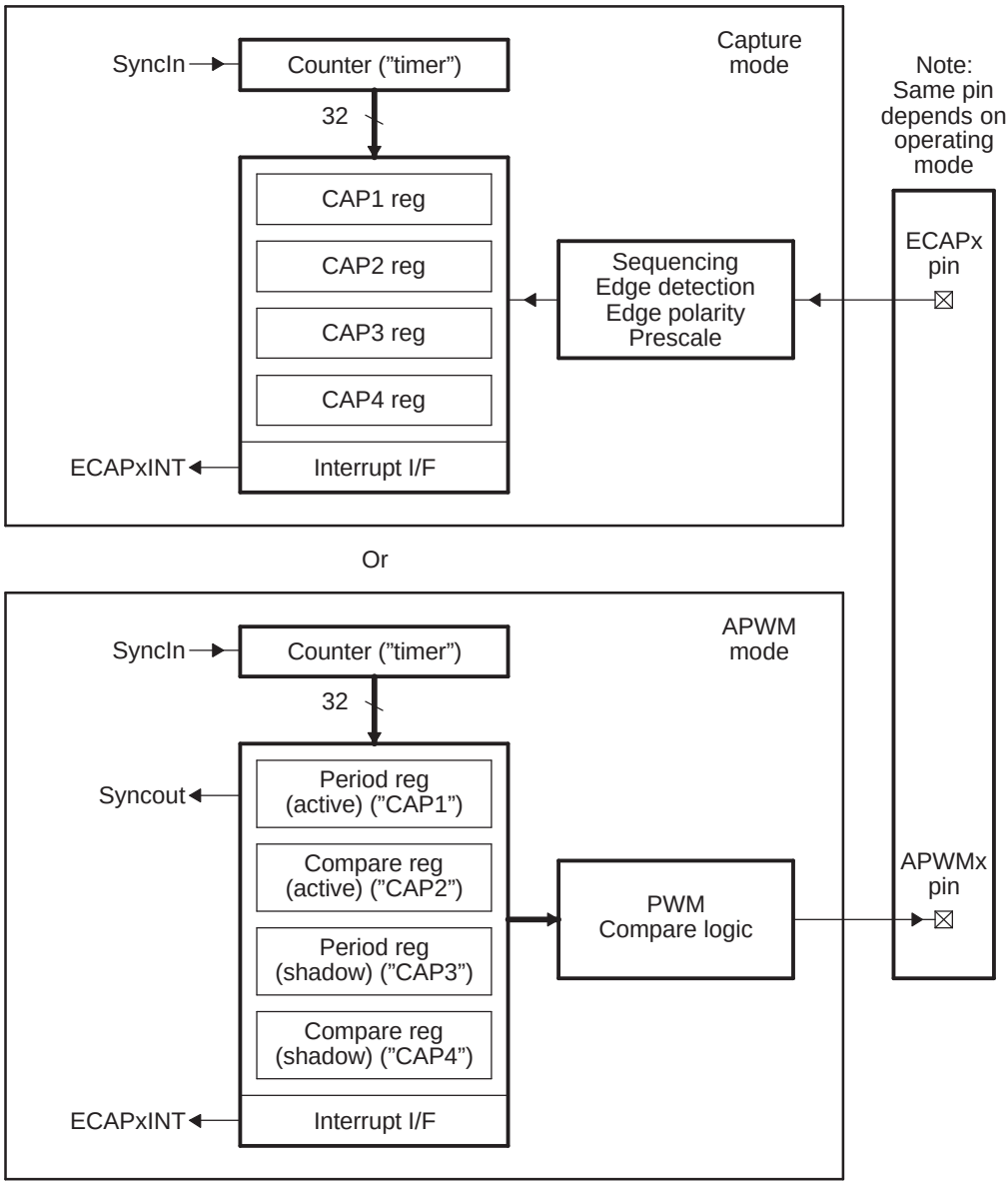


Copyright © 2013–2016, Texas Instruments Incorporated

15.2.1 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 15-2 is a high-level view of both the capture and auxiliary pulse-width modulator (APWM) modes of operation.

Figure 15-2. Capture and APWM Modes of Operation



(1) A single pin is shared between CAP and APWM functions. In capture mode, it is an input; in APWM mode, it is an output.

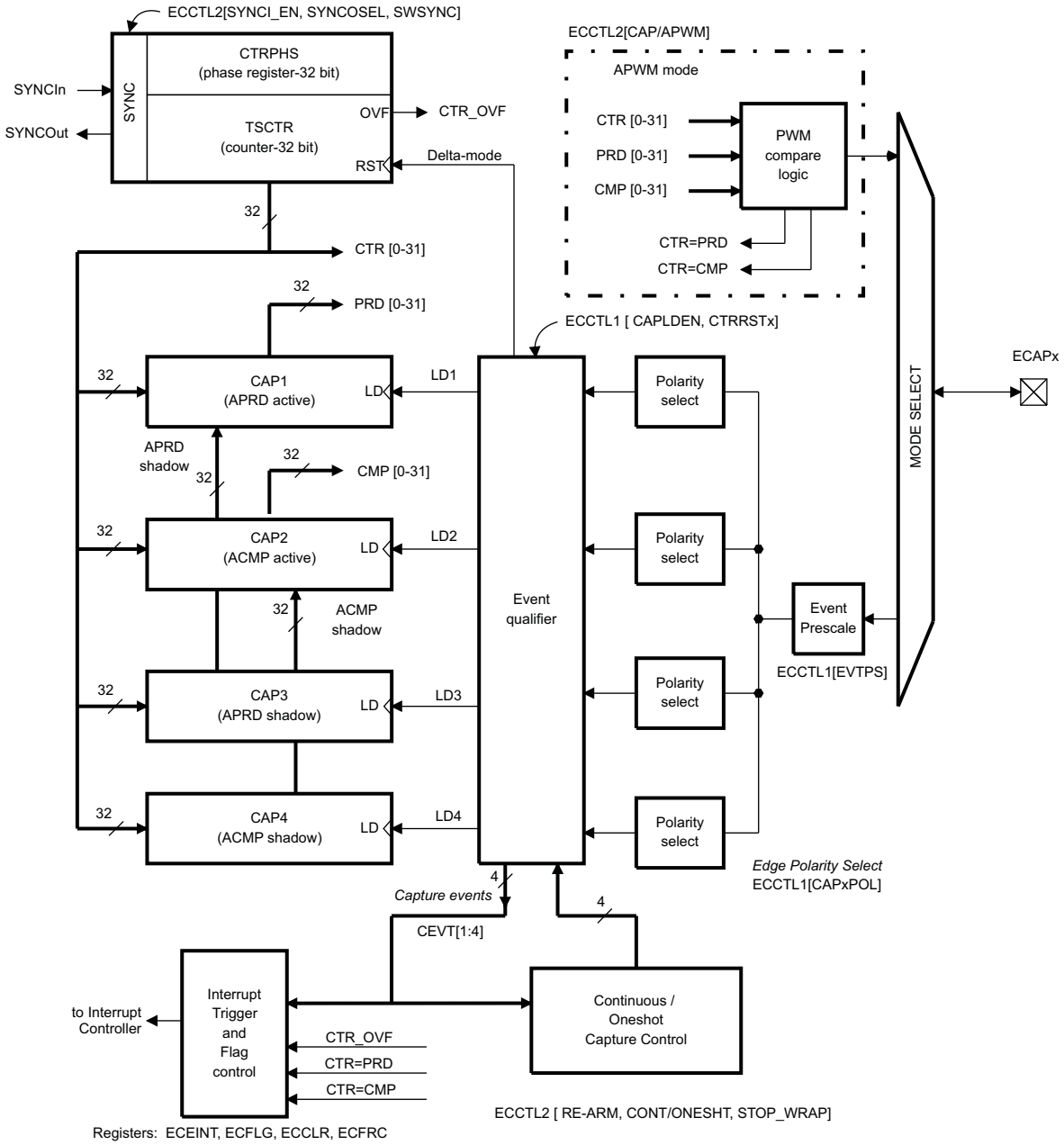
(2) In APWM mode, writing any value to CAP1/CAP2 active registers also writes the same value to the corresponding shadow registers CAP3/CAP4. This emulates immediate mode. Writing to the shadow registers CAP3/CAP4 invokes the shadow mode.

Copyright © 2013–2016, Texas Instruments Incorporated

15.2.2 Capture Mode Description

Figure 15-3 shows the various components that implement the capture function.

Figure 15-3. Capture Function Diagram

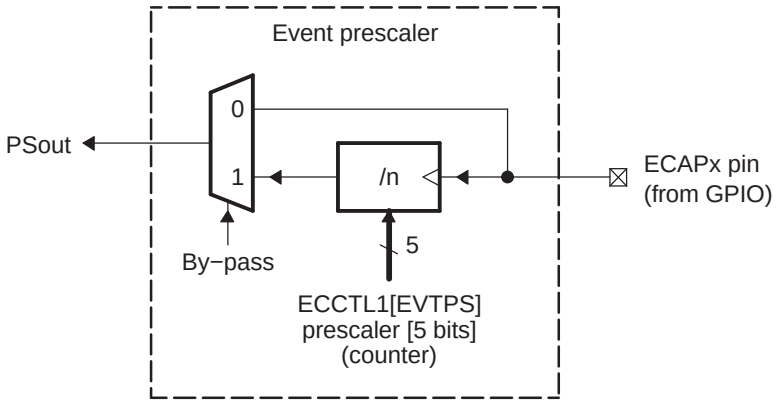


Copyright © 2013–2016, Texas Instruments Incorporated

15.2.2.1 Event Prescaler

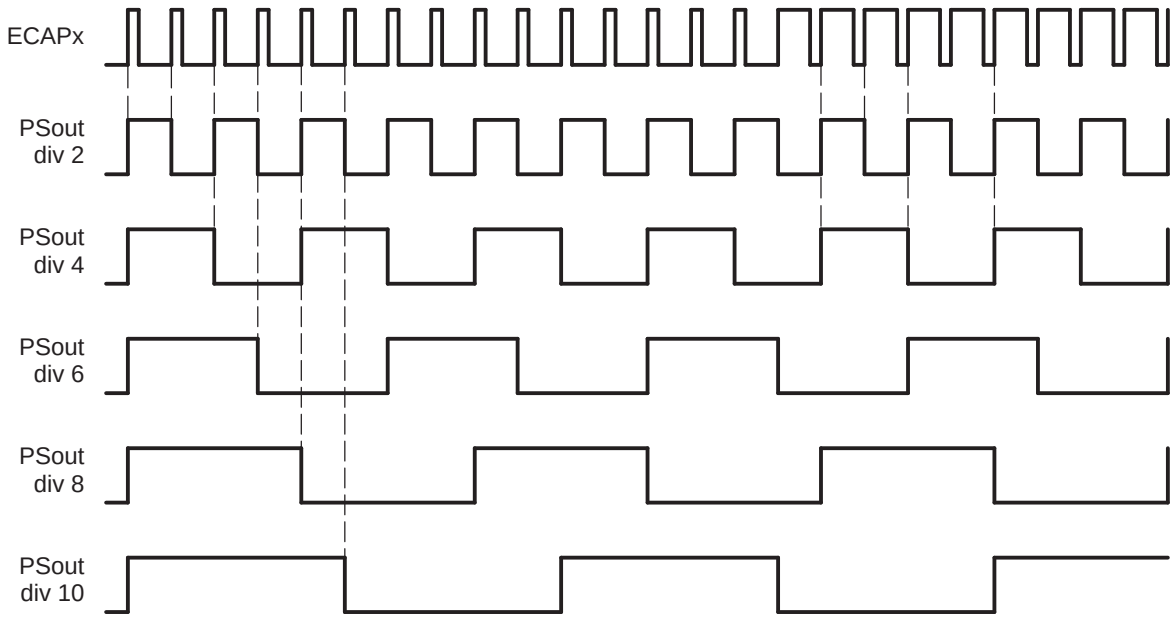
An input capture signal (pulse train) can be prescaled by N = 2-62 (in multiples of 2) or can bypass the prescaler. This is useful when very high frequency signals are used as inputs. Figure 15-4 shows a functional diagram and Figure 15-5 shows the operation of the prescale function.

Figure 15-4. Event Prescale Control



(1) When a prescale value of 1 is chosen (ECCTL1[13:9] = 0000) the input capture signal by-passes the prescale logic completely.

Figure 15-5. Prescale Function Waveforms



Copyright © 2013–2016, Texas Instruments Incorporated

15.2.2.2 Edge Polarity Select and Qualifier

- Four independent edge polarity (rising edge/falling edge) selection multiplexers 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 CAP*n* register by the Mod4 counter. The CAP*n* register is loaded on the falling edge.

15.2.2.3 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.

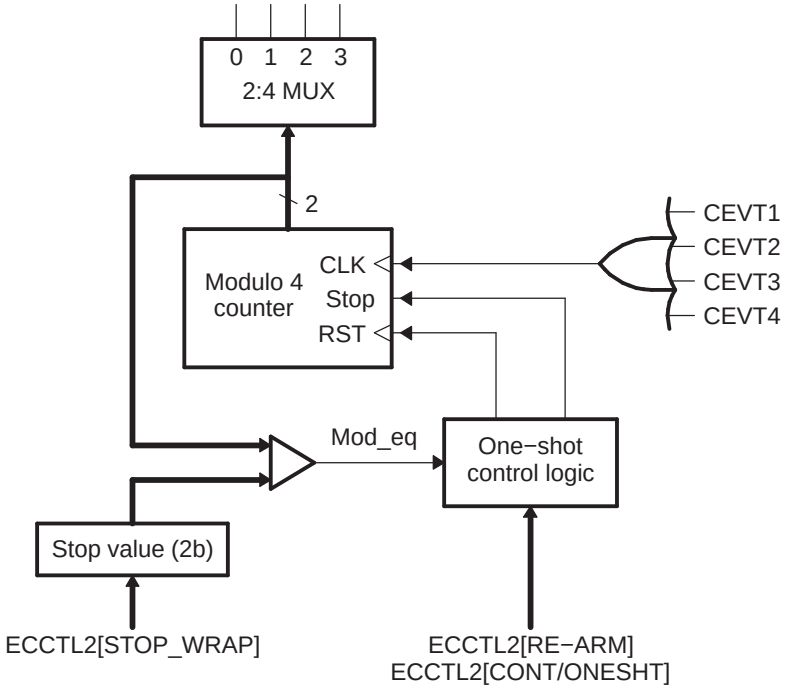
The continuous/one-shot block (Figure 15-6) controls the start/stop and reset (zero) functions of the Mod4 counter via a mono-shot type of action that can be triggered by the stop-value comparator and re-armed via software control.

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 15-6. Continuous/One-shot Block Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

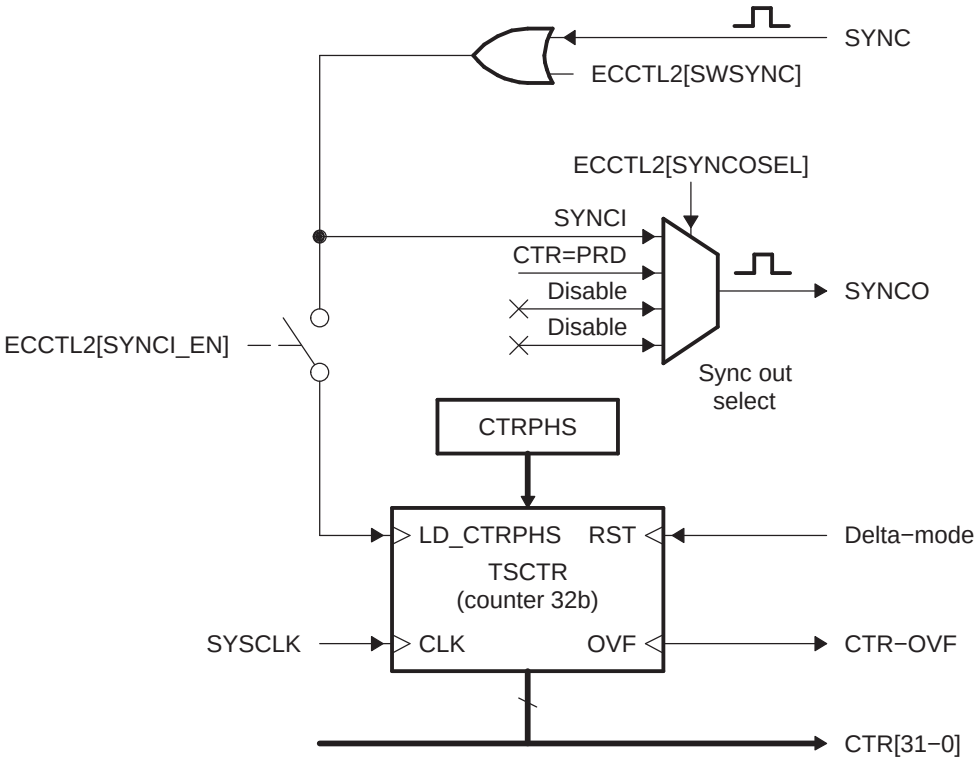
15.2.2.4 32-Bit Counter and Phase Control

This counter (Figure 15-7) 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.

Figure 15-7. Counter and Synchronization Block Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

15.2.2.5 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.

15.2.2.6 Interrupt Control

An Interrupt can be generated on capture events (CEVT1-CEVT4, CTROVF) or APWM events (CTR = PRD, CTR = CMP). See Figure 15-8.

A counter overflow event (FFFF FFFFh->0000 0000h) is also provided as an interrupt source (CTROVF).

The capture events are edge and sequencer qualified (that is, ordered in time) by the polarity select and Mod4 gating, respectively.

One of these events can be selected as the interrupt source (from the eCAP*n* module) going to the interrupt controller.

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 interrupt controller 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.

15.2.2.7 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.

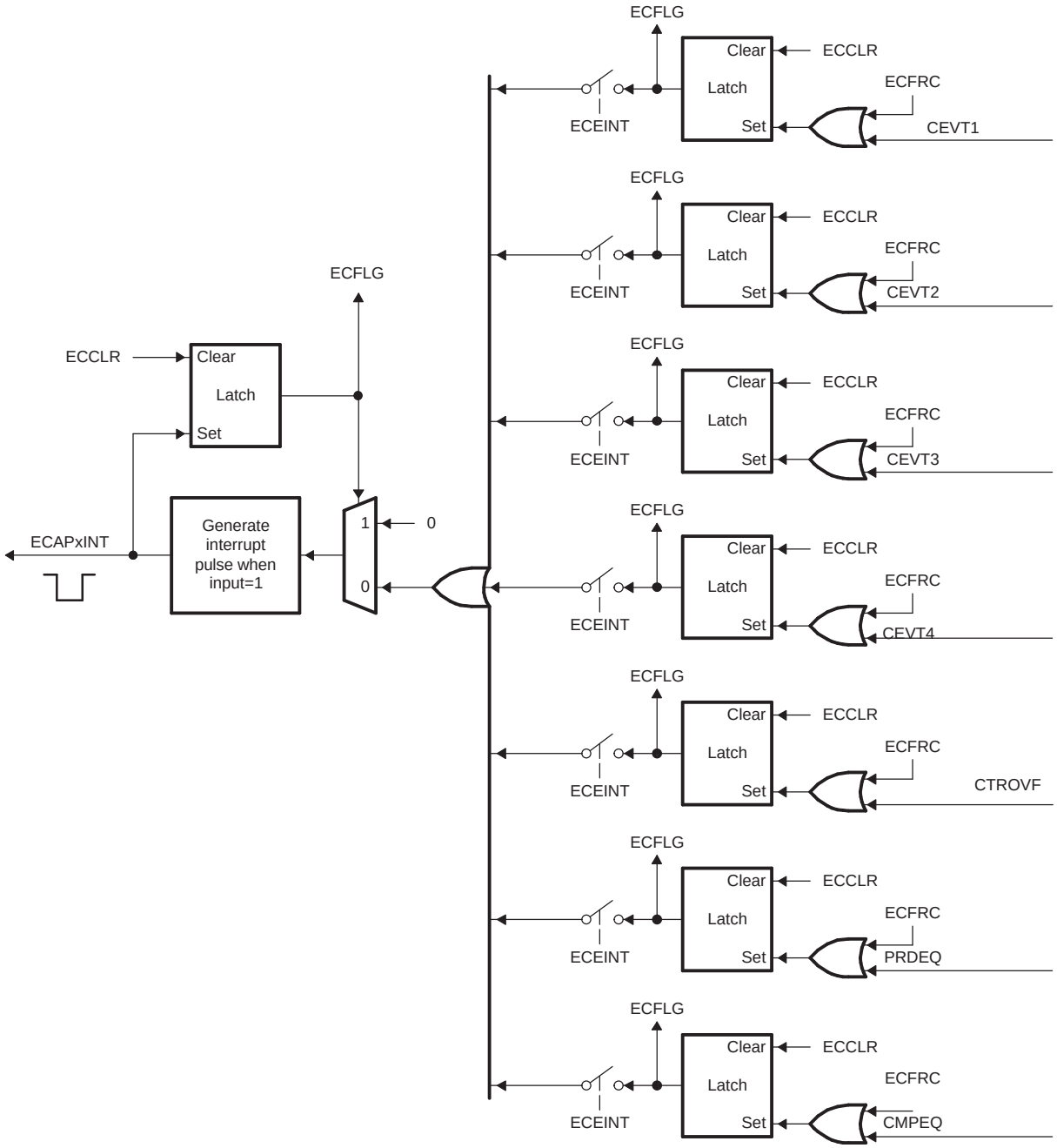
In APWM mode, shadow loading is active and two choices are permitted:

- 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]

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.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 15-8. Interrupts in eCAP Module



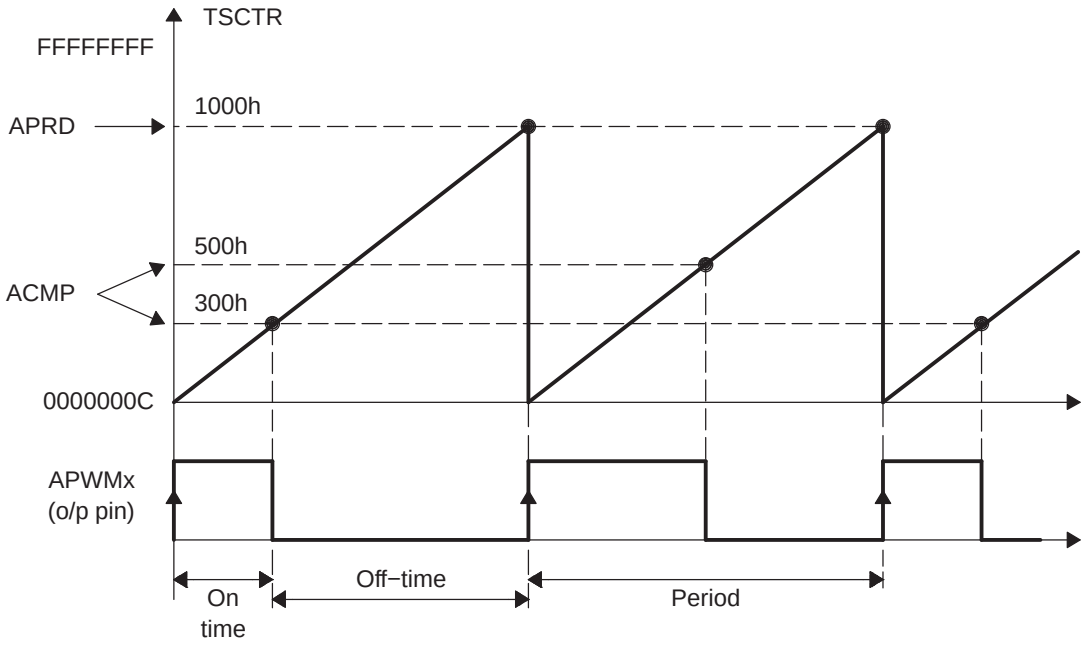
Copyright © 2013–2016, Texas Instruments Incorporated

15.2.2.8 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 15-9. PWM Waveform Details Of APWM Mode Operation



The behavior of APWM active-high mode (APWMPOL == 0) is:

```
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:

```
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)
```


Copyright © 2013–2016, Texas Instruments Incorporated

```
CMP > PERIOD+1, output low for complete period
```

15.3 Applications

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, below are useful #defines which will help in the understanding of the examples.

```
// ECCTL1 ( ECAP Control Reg 1)
//==========================
// 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
```

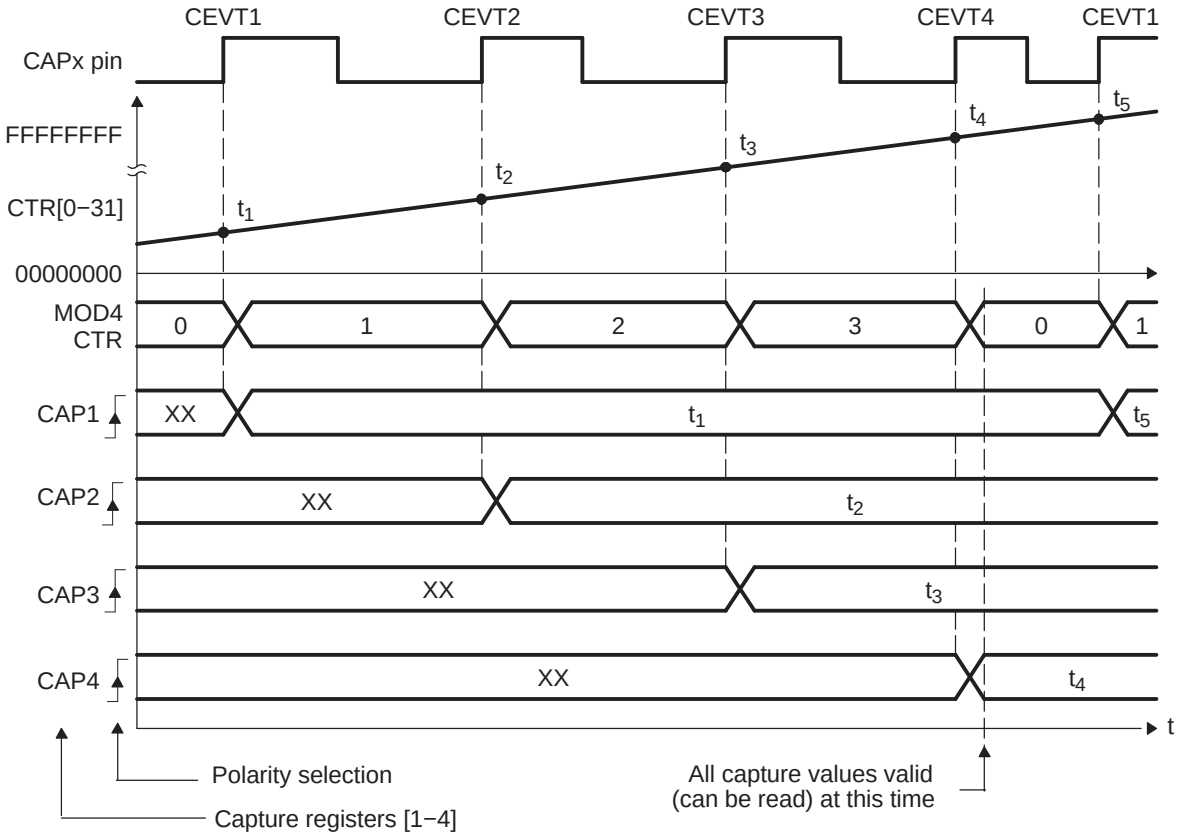

Copyright © 2013–2016, Texas Instruments Incorporated

15.3.1 Absolute Time-Stamp Operation Rising Edge Trigger Example

Figure 15-10 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 FFFF FFFFh (maximum value), it wraps around to 0000 0000h (not shown in Figure 15-10), 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 CAP*n* registers.

Figure 15-10. Capture Sequence for Absolute Time-Stamp, Rising Edge Detect



Copyright © 2013–2016, Texas Instruments Incorporated

Table 15-1. ECAP Initialization for CAP Mode Absolute Time, Rising Edge Trigger

Register	Bit	Value
ECCTL1	CAP1POL	EC_RISING
ECCTL1	CAP2POL	EC_RISING
ECCTL1	CAP3POL	EC_RISING
ECCTL1	CAP4POL	EC_RISING
ECCTL1	CTRRST1	EC_ABS_MODE
ECCTL1	CTRRST2	EC_ABS_MODE
ECCTL1	CTRRST3	EC_ABS_MODE
ECCTL1	CTRRST4	EC_ABS_MODE
ECCTL1	CAPLDEN	EC_ENABLE
ECCTL1	PRESCALE	EC_DIV1
ECCTL2	CAP_APWM	EC_CAP_MODE
ECCTL2	CONT_ONESHT	EC_CONTINUOUS
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	SYNCI_EN	EC_DISABLE
ECCTL2	TSCTRSTOP	EC_RUN

Example 15-1. Code Snippet for CAP Mode Absolute Time, Rising Edge Trigger

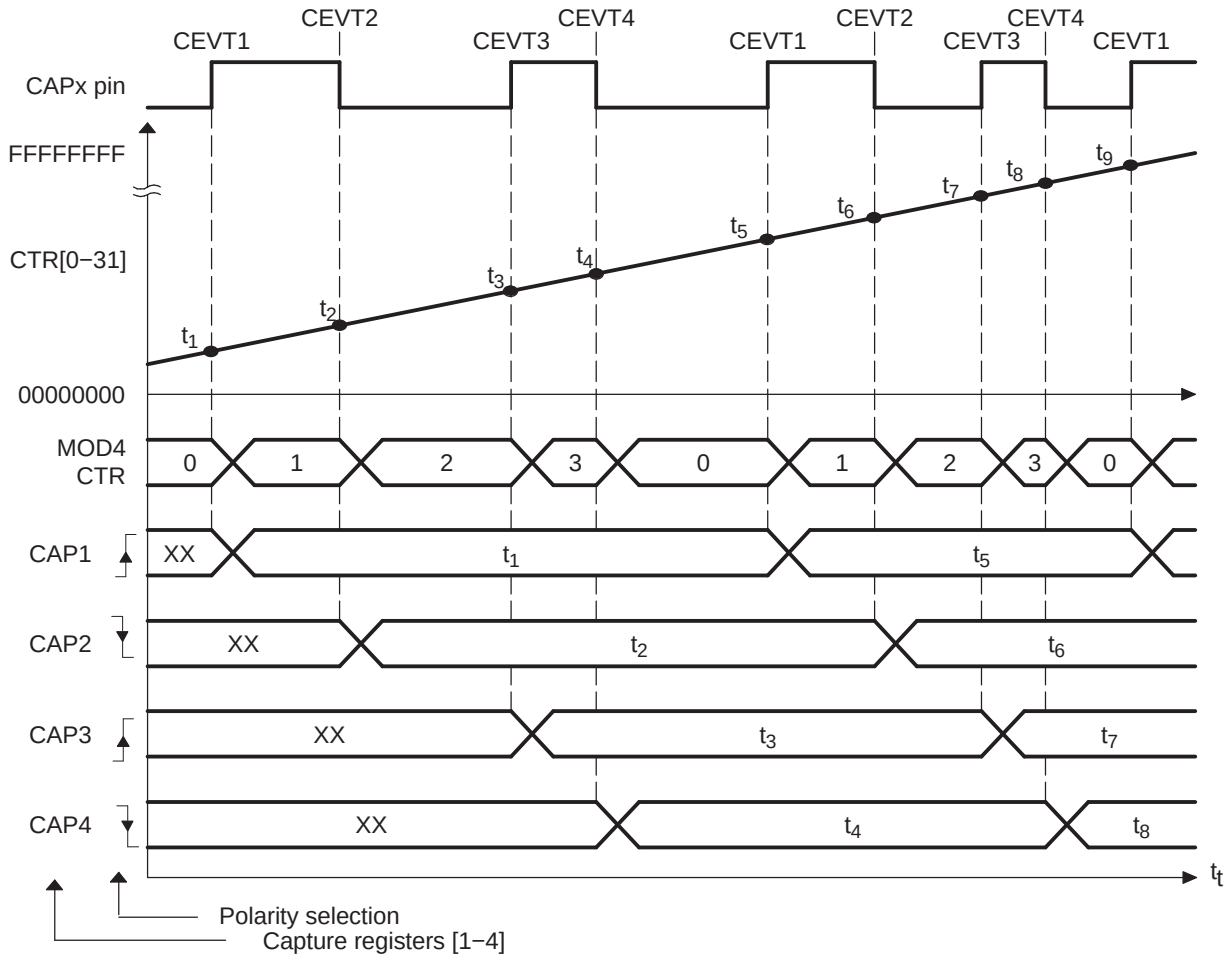
```
// Code snippet for CAP mode Absolute Time, Rising edge trigger

// Run Time ( e.g. CEVT4 triggered ISR call)
//=========================================
    TSt1 = ECAPxRegs.CAP1;        // Fetch Time-Stamp captured at t1
    TSt2 = ECAPxRegs.CAP2;        // Fetch Time-Stamp captured at t2
    TSt3 = ECAPxRegs.CAP3;        // Fetch Time-Stamp captured at t3
    TSt4 = ECAPxRegs.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 © 2013–2016, Texas Instruments Incorporated

15.3.2 Absolute Time-Stamp Operation Rising and Falling Edge Trigger Example

In Figure 15-11 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: 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 15-11. Capture Sequence for Absolute Time-Stamp, Rising and Falling Edge Detect



Copyright © 2013–2016, Texas Instruments Incorporated

Table 15-2. ECAP Initialization for CAP Mode Absolute Time, Rising and Falling Edge Trigger

Register	Bit	Value
ECCTL1	CAP1POL	EC_RISING
ECCTL1	CAP2POL	EC_FALLING
ECCTL1	CAP3POL	EC_RISING
ECCTL1	CAP4POL	EC_FALLING
ECCTL1	CTRRST1	EC_ABS_MODE
ECCTL1	CTRRST2	EC_ABS_MODE
ECCTL1	CTRRST3	EC_ABS_MODE
ECCTL1	CTRRST4	EC_ABS_MODE
ECCTL1	CAPLDEN	EC_ENABLE
ECCTL1	PRESCALE	EC_DIV1
ECCTL2	CAP_APWM	EC_CAP_MODE
ECCTL2	CONT_ONESHT	EC_CONTINUOUS
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	SYNCI_EN	EC_DISABLE
ECCTL2	TSCTRSTOP	EC_RUN

Example 15-2. Code Snippet for CAP Mode Absolute Time, Rising and Falling Edge Trigger

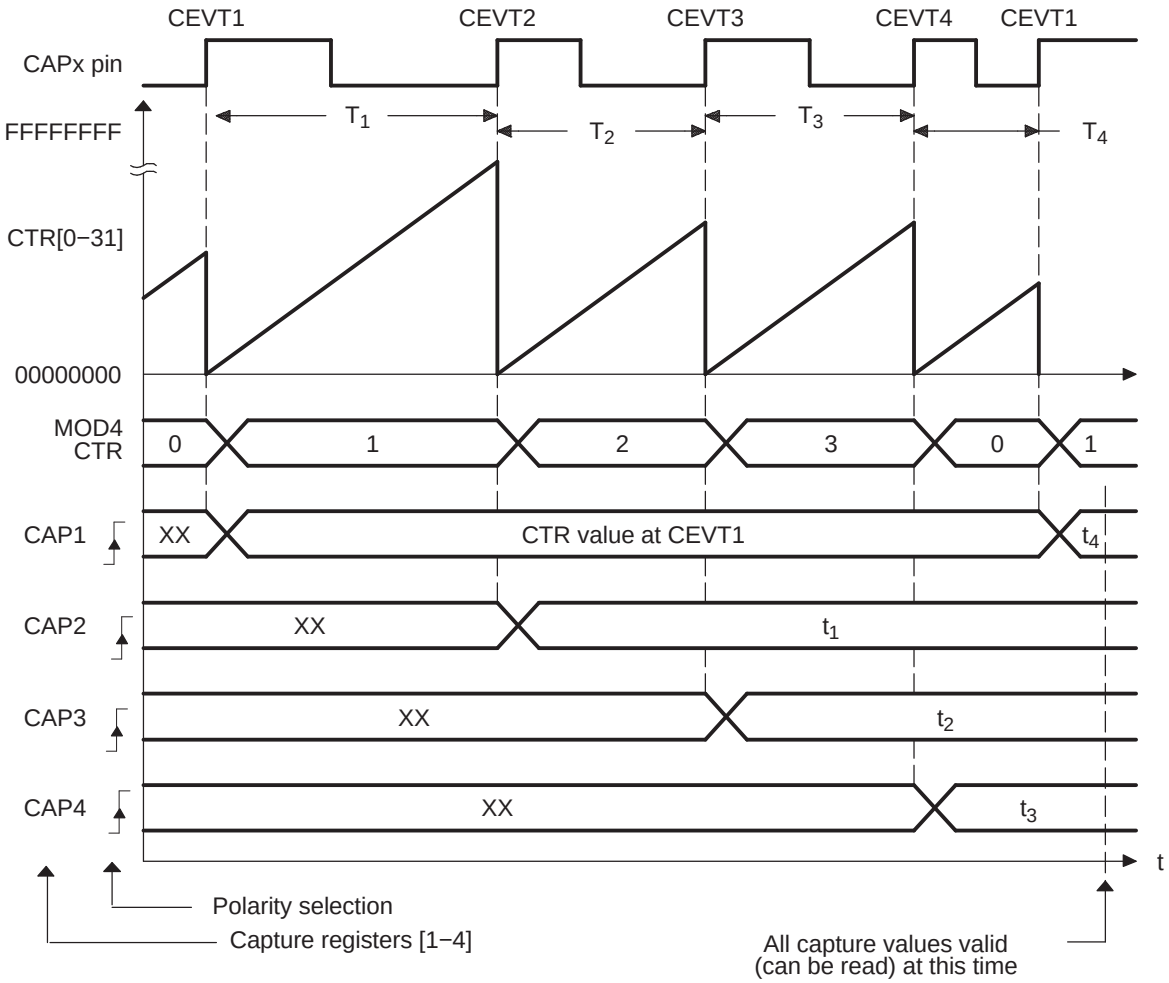
```
// Code snippet for CAP mode Absolute Time, Rising & Falling edge triggers

// Run Time ( e.g. CEVT4 triggered ISR call)
//=========================================
    TSt1 = ECAPxRegs.CAP1;        // Fetch Time-Stamp captured at t1
    TSt2 = ECAPxRegs.CAP2;        // Fetch Time-Stamp captured at t2
    TSt3 = ECAPxRegs.CAP3;        // Fetch Time-Stamp captured at t3
    TSt4 = ECAPxRegs.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 © 2013–2016, Texas Instruments Incorporated

15.3.3 Time Difference (Delta) Operation Rising Edge Trigger Example

Figure 15-12 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 FFFF FFFFh (maximum value), before the next event, it wraps around to 0000 0000h and continues, a CNTOVF (counter overflow) Flag is set, and an Interrupt (if enabled) occurs. The advantage of Delta-time Mode is that the CAP*n* contents directly give timing data without the need for CPU calculations: Period1 = T_1, Period2 = T_2,…etc. As shown in Figure 15-12, 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 15-12. Capture Sequence for Delta Mode Time-Stamp, Rising Edge Detect



Copyright © 2013–2016, Texas Instruments Incorporated

Table 15-3. ECAP Initialization for CAP Mode Delta Time, Rising Edge Trigger

Register	Bit	Value
ECCTL1	CAP1POL	EC_RISING
ECCTL1	CAP2POL	EC_RISING
ECCTL1	CAP3POL	EC_RISING
ECCTL1	CAP4POL	EC_RISING
ECCTL1	CTRRST1	EC_DELTA_MODE
ECCTL1	CTRRST2	EC_DELTA_MODE
ECCTL1	CTRRST3	EC_DELTA_MODE
ECCTL1	CTRRST4	EC_DELTA_MODE
ECCTL1	CAPLDEN	EC_ENABLE
ECCTL1	PRESCALE	EC_DIV1
ECCTL2	CAP_APWM	EC_CAP_MODE
ECCTL2	CONT_ONESHT	EC_CONTINUOUS
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	SYNCI_EN	EC_DISABLE
ECCTL2	TSCTRSTOP	EC_RUN

Example 15-3. Code Snippet for CAP Mode Delta Time, Rising Edge Trigger

```
// Code snippet for CAP mode Delta Time, Rising edge trigger

// Run Time ( e.g. CEVT1 triggered ISR call)
//=========================================
// Note: here Time-stamp directly represents the Period value.
    Period4 = ECAPxRegs.CAP1;    // Fetch Time-Stamp captured at T1
    Period1 = ECAPxRegs.CAP2;    // Fetch Time-Stamp captured at T2
    Period2 = ECAPxRegs.CAP3;    // Fetch Time-Stamp captured at T3
    Period3 = ECAPxRegs.CAP4;    // Fetch Time-Stamp captured at T4
```

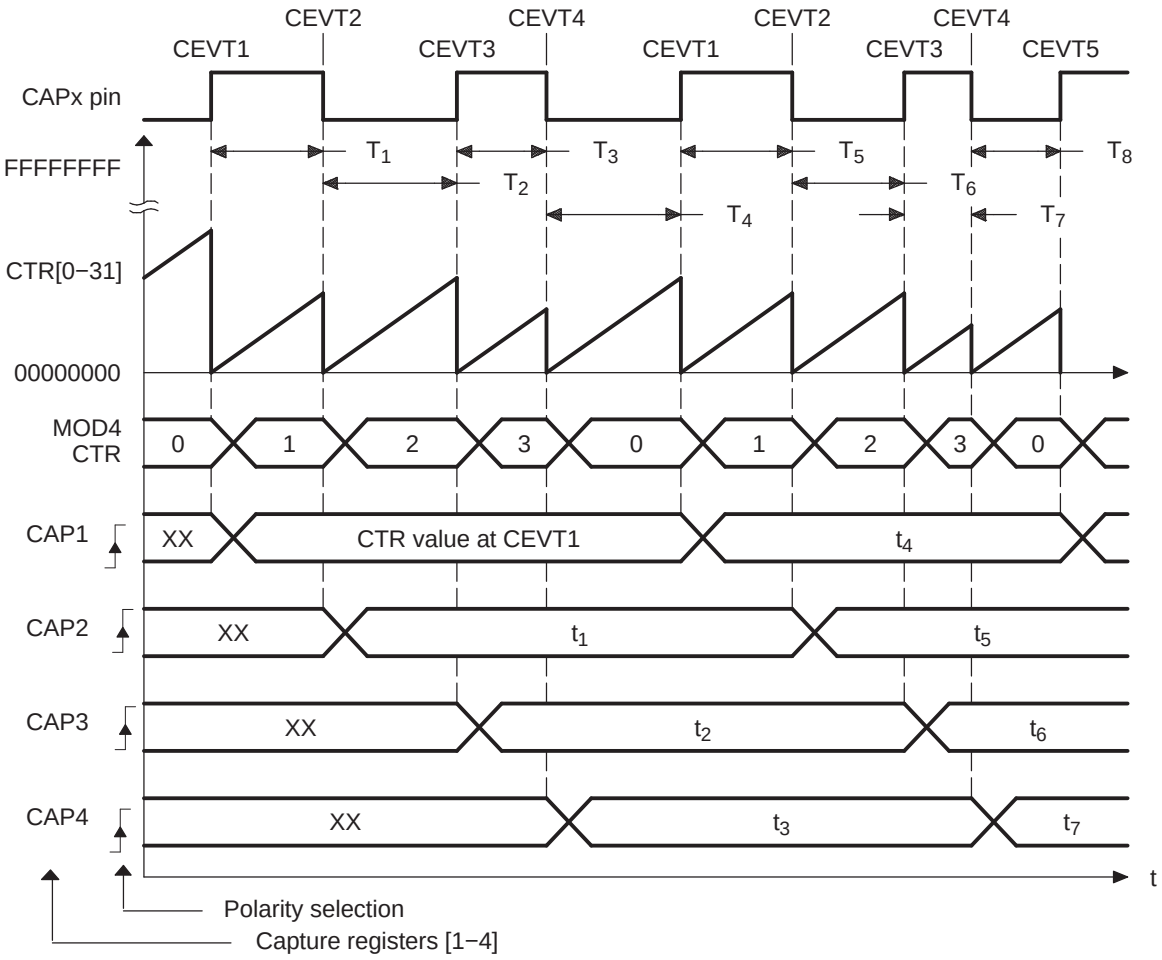

Copyright © 2013–2016, Texas Instruments Incorporated

15.3.4 Time Difference (Delta) Operation Rising and Falling Edge Trigger Example

In Figure 15-13 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: Period1 = T_1 + T_2, Period2 = T_3 + T_4, …etc Duty Cycle1 (on-time %) = T_1 / Period1 × 100%, etc Duty Cycle1 (off-time %) = T_2 / Period1 × 100%, etc

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, that is, during run-time, only the shadow registers must be used.

Figure 15-13. Capture Sequence for Delta Mode Time-Stamp, Rising and Falling Edge Detect



Copyright © 2013–2016, Texas Instruments Incorporated

Table 15-4. ECAP Initialization for CAP Mode Delta Time, Rising and Falling Edge Triggers

Register	Bit	Value
ECCTL1	CAP1POL	EC_RISING
ECCTL1	CAP2POL	EC_FALLING
ECCTL1	CAP3POL	EC_RISING
ECCTL1	CAP4POL	EC_FALLING
ECCTL1	CTRRST1	EC_DELTA_MODE
ECCTL1	CTRRST2	EC_DELTA_MODE
ECCTL1	CTRRST3	EC_DELTA_MODE
ECCTL1	CTRRST4	EC_DELTA_MODE
ECCTL1	CAPLDEN	EC_ENABLE
ECCTL1	PRESCALE	EC_DIV1
ECCTL2	CAP_APWM	EC_CAP_MODE
ECCTL2	CONT_ONESHT	EC_CONTINUOUS
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	SYNCI_EN	EC_DISABLE
ECCTL2	TSCTRSTOP	EC_RUN

Example 15-4. Code Snippet for CAP Mode Delta Time, Rising and Falling Edge Triggers

```
// Code snippet for CAP mode Delta Time, Rising and Falling edge triggers

// Run Time ( e.g. CEVT1 triggered ISR call)
//=========================================
// Note: here Time-stamp directly represents the Duty cycle values.
    DutyOnTime1 =  ECAPxRegs.CAP2;    // Fetch Time-Stamp captured at T2
    DutyOffTime1 = ECAPxRegs.CAP3;    // Fetch Time-Stamp captured at T3
    DutyOnTime2 =  ECAPxRegs.CAP4;    // Fetch Time-Stamp captured at T4
    DutyOffTime2 = ECAPxRegs.CAP1;    // Fetch Time-Stamp captured at T1

    Period1 = DutyOnTime1 + DutyOffTime1;
    Period2 = DutyOnTime2 + DutyOffTime2;
```

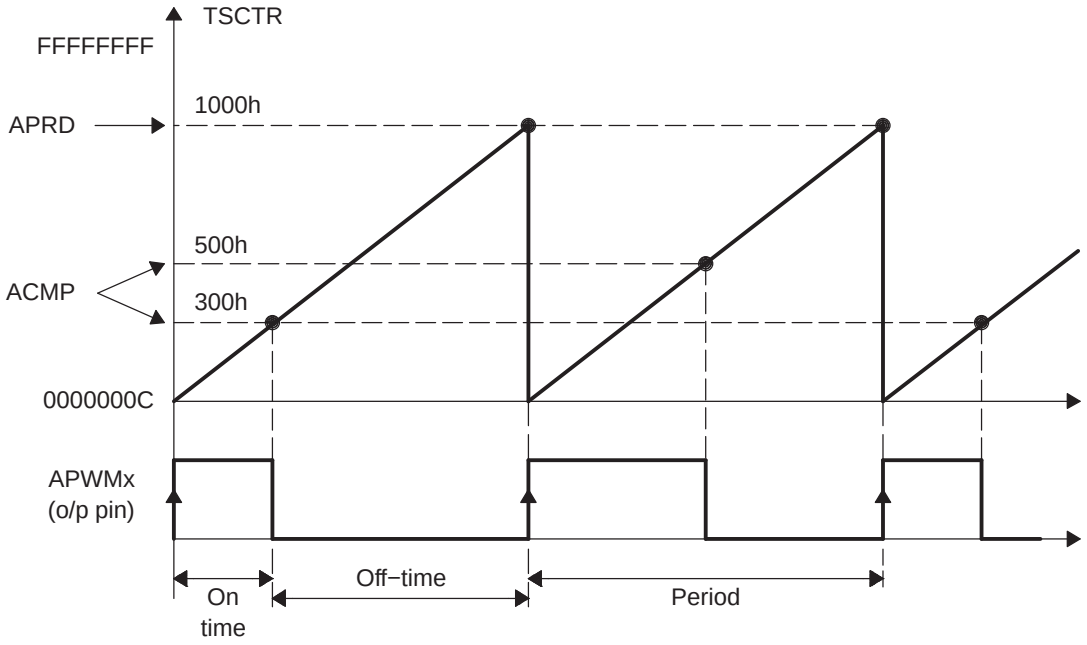

Copyright © 2013–2016, Texas Instruments Incorporated

15.3.5 Application of the APWM Mode

15.3.5.1 Simple PWM Generation (Independent Channel/s) Example

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 APWM*n*. 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.

Figure 15-14. PWM Waveform Details of APWM Mode Operation



Copyright © 2013–2016, Texas Instruments Incorporated

Table 15-5. ECAP Initialization for APWM Mode

Register	Bit	Value
CAP1	CAP1	0x1000
CTRPHS	CTRPHS	0x0
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCI_EN	EC_DISABLE
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	TSCTRSTOP	EC_RUN

Example 15-5. Code Snippet for APWM Mode

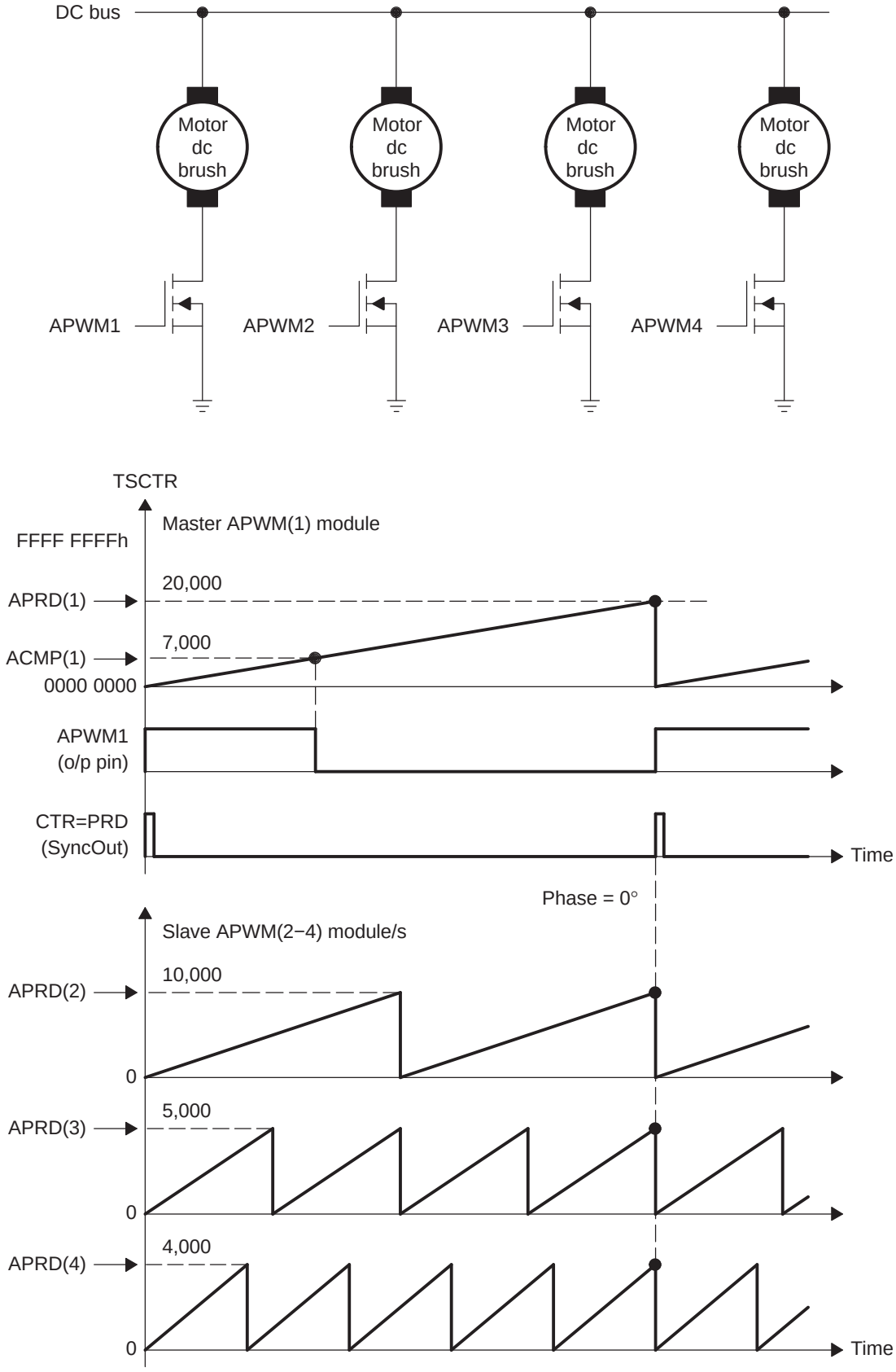
```
// Code snippet for APWM mode Example 1

// Run Time (Instant 1, e.g. ISR call)
//======================
   ECAPxRegs.CAP2 = 0x300;      // Set Duty cycle i.e. compare value

// Run Time (Instant 2, e.g. another ISR call)
//======================
   ECAPxRegs.CAP2 = 0x500;      // Set Duty cycle i.e. compare value
```

15.3.5.2 Multichannel PWM Generation with Synchronization Example

Figure 15-15 takes advantage of the synchronization feature between eCAP modules. Here 4 independent PWM channels are required with different frequencies, but at integer multiples of each other to avoid "beat" frequencies. Hence one eCAP module is configured as the Master and the remaining 3 are Slaves all receiving their synch pulse (CTR = PRD) from the master. Note the Master is chosen to have the lower frequency (F1 = 1/20,000) requirement. Here Slave2 Freq = 2 × F1, Slave3 Freq = 4 × F1 and Slave4 Freq = 5 × F1. Note here values are in decimal notation. Also, only the APWM1 output waveform is shown.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 15-15. Multichannel PWM Example Using 4 eCAP Modules



Copyright © 2013–2016, Texas Instruments Incorporated

Table 15-6. ECAP1 Initialization for Multichannel PWM Generation with Synchronization

Register	Bit	Value
CAP1	CAP1	20000
CTRPHS	CTRPHS	0
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCI_EN	EC_DISABLE
ECCTL2	SYNCO_SEL	EC_CTR_PRD
ECCTL2	TSCTRSTOP	EC_RUN

Table 15-7. ECAP2 Initialization for Multichannel PWM Generation with Synchronization

Register	Bit	Value
CAP1	CAP1	10000
CTRPHS	CTRPHS	0
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCI_EN	EC_ENABLE
ECCTL2	SYNCO_SEL	EC_SYNCI
ECCTL2	TSCTRSTOP	EC_RUN

Table 15-8. ECAP3 Initialization for Multichannel PWM Generation with Synchronization

Register	Bit	Value
CAP1	CAP1	5000
CTRPHS	CTRPHS	0
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCI_EN	EC_ENABLE
ECCTL2	SYNCO_SEL	EC_SYNCI
ECCTL2	TSCTRSTOP	EC_RUN

Table 15-9. ECAP4 Initialization for Multichannel PWM Generation with Synchronization

Register	Bit	Value
CAP1	CAP1	4000
CTRPHS	CTRPHS	0
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCI_EN	EC_ENABLE
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	TSCTRSTOP	EC_RUN

Copyright © 2013–2016, Texas Instruments Incorporated

Example 15-6. Code Snippet for Multichannel PWM Generation with Synchronization

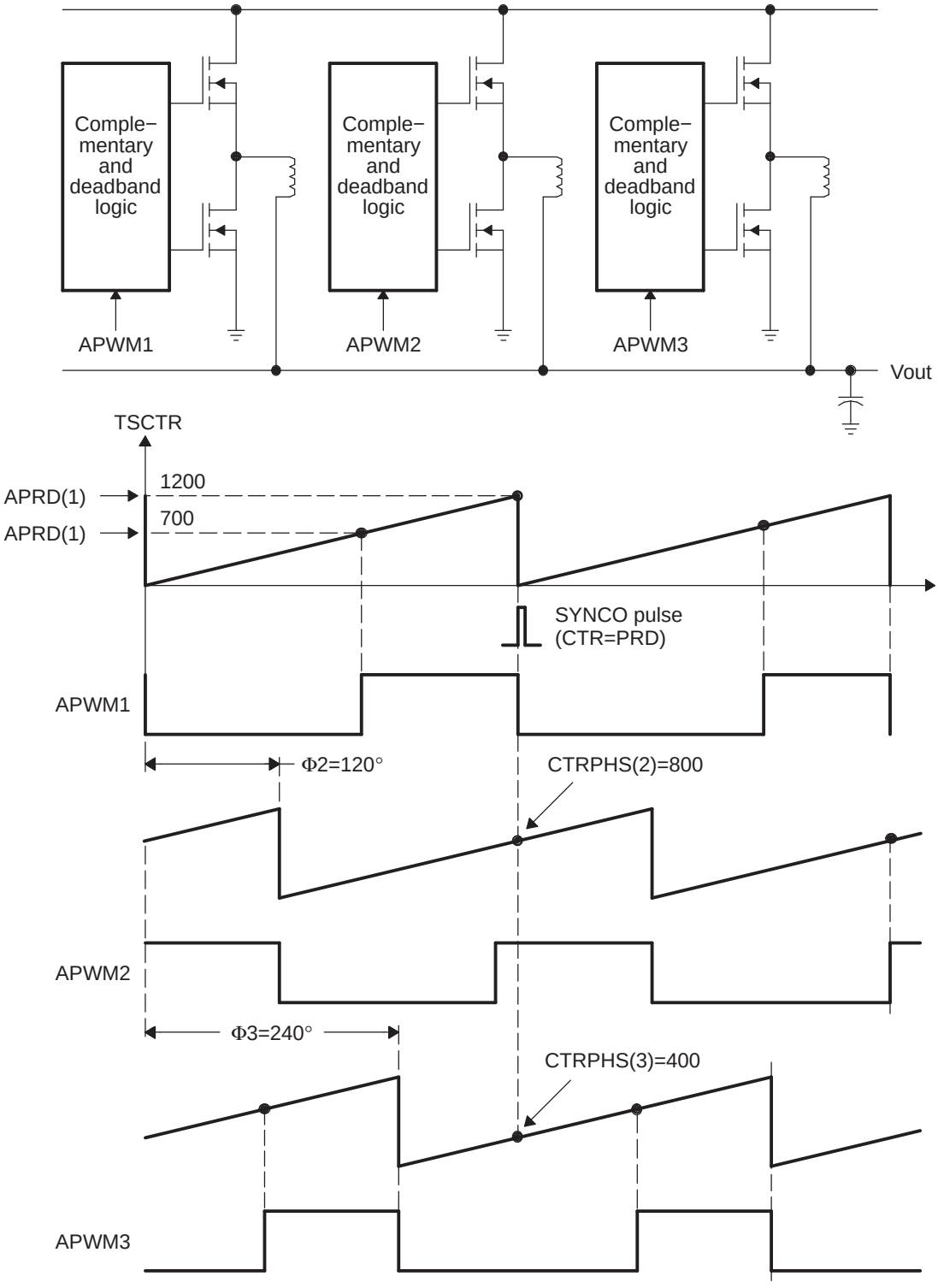
```
// Code snippet for APWM mode Example 2

// Run Time (Note: Example execution of one run-time instant)
//=============================================================
   ECAP1Regs.CAP2 = 7000;    // Set Duty cycle i.e., compare value = 7000
   ECAP2Regs.CAP2 = 2000;    // Set Duty cycle i.e., compare value = 2000
   ECAP3Regs.CAP2 = 550;     // Set Duty cycle i.e., compare value = 550
   ECAP4Regs.CAP2 = 6500;    // Set Duty cycle i.e., compare value = 6500
```

15.3.5.3 Multichannel PWM Generation with Phase Control Example

In Figure 15-16, the Phase control feature of the APWM mode is used to control a 3 phase Interleaved DC/DC converter topology. This topology requires each phase to be off-set by 120° from each other. Hence if “Leg” 1 (controlled by APWM1) is the reference Leg (or phase), that is, 0°, then Leg 2 need 120° off-set and Leg 3 needs 240° off-set. The waveforms in Figure 15-16 show the timing relationship between each of the phases (Legs). Note eCAP1 module is the Master and issues a sync out pulse to the slaves (modules 2, 3) whenever TSCTR = Period value.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 15-16. Multiphase (channel) Interleaved PWM Example Using 3 eCAP Modules



Copyright © 2013–2016, Texas Instruments Incorporated

Table 15-10. ECAP1 Initialization for Multichannel PWM Generation with Phase Control

Register	Bit	Value
CAP1	CAP1	1200
CTRPHS	CTRPHS	0
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCI_EN	EC_DISABLE
ECCTL2	SYNCO_SEL	EC_CTR_PRD
ECCTL2	TSCTRSTOP	EC_RUN

Table 15-11. ECAP2 Initialization for Multichannel PWM Generation with Phase Control

Register	Bit	Value
CAP1	CAP1	1200
CTRPHS	CTRPHS	800
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCI_EN	EC_ENABLE
ECCTL2	SYNCO_SEL	EC_SYNCI
ECCTL2	TSCTRSTOP	EC_RUN

Table 15-12. ECAP3 Initialization for Multichannel PWM Generation with Phase Control

Register	Bit	Value
CAP1	CAP1	1200
CTRPHS	CTRPHS	400
ECCTL2	CAP_APWM	EC_APWM_MODE
ECCTL2	APWMPOL	EC_ACTV_HI
ECCTL2	SYNCI_EN	EC_ENABLE
ECCTL2	SYNCO_SEL	EC_SYNCO_DIS
ECCTL2	TSCTRSTOP	EC_RUN

Example 15-7. Code Snippet for Multichannel PWM Generation with Phase Control

```
// Code snippet for APWM mode Example 3


// Run Time (Note: Example execution of one run-time instant)
//=============================================================
// All phases are set to the same duty cycle
  ECAP1Regs.CAP2 = 700;     // Set Duty cycle i.e. compare value = 700
  ECAP2Regs.CAP2 = 700;     // Set Duty cycle i.e. compare value = 700
  ECAP3Regs.CAP2 = 700;     // Set Duty cycle i.e. compare value = 700
```


Copyright © 2013–2016, Texas Instruments Incorporated

15.4 Registers

Table 15-13 shows the eCAP module control and status register set. All 32-bit registers are aligned on even address boundaries and are organized in little-endian mode. The 16 least-significant bits of a 32-bit register are located on lowest address (even address).

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.

Table 15-13. Control and Status Register Set

Offset	Acronym	Description	Size (×16)	Section
0h	TSCTR	Time-Stamp Counter Register	2	Section 15.4.1
4h	CTRPHS	Counter Phase Offset Value Register	2	Section 15.4.2
8h	CAP1	Capture 1 Register	2	Section 15.4.3
Ch	CAP2	Capture 2 Register	2	Section 15.4.4
10h	CAP3	Capture 3 Register	2	Section 15.4.5
14h	CAP4	Capture 4 Register	2	Section 15.4.6
28h	ECCTL1	Capture Control Register 1	1	Section 15.4.7
2Ah	ECCTL2	Capture Control Register 2	1	Section 15.4.8
2Ch	ECEINT	Capture Interrupt Enable Register	1	Section 15.4.9
2Eh	ECFLG	Capture Interrupt Flag Register	1	Section 15.4.10
30h	ECCLR	Capture Interrupt Clear Register	1	Section 15.4.11
32h	ECFRC	Capture Interrupt Force Register	1	Section 15.4.12
5Ch	REVID	Revision ID Register	2	Section 15.4.13

15.4.1 Time-Stamp Counter Register (TSCTR)

The time-stamp counter register (TSCTR) is shown in Figure 15-17 and described in Table 15-14.

Figure 15-17. Time-Stamp Counter Register (TSCTR)

31	0
TSCTR	
R/W-0	

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

Table 15-14. Time-Stamp Counter Register (TSCTR) Field Descriptions

Bit	Field	Value	Description
31-0	TSCTR	0-FFFF FFFFh	Active 32-bit counter register that is used as the capture time-base

Copyright © 2013–2016, Texas Instruments Incorporated

15.4.2 Counter Phase Control Register (CTRPHS)

The counter phase control register (CTRPHS) is shown in Figure 15-18 and described in Table 15-15.

Figure 15-18. Counter Phase Control Register (CTRPHS)

31	0
CTRPHS	
R/W-0	

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

Table 15-15. Counter Phase Control Register (CTRPHS) Field Descriptions

Bit	Field	Value	Description
31-0	CTRPHS	0-FFFF FFFFh	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.

15.4.3 Capture 1 Register (CAP1)

The capture 1 register (CAP1) is shown in Figure 15-19 and described in Table 15-16.

Figure 15-19. Capture 1 Register (CAP1)

31	0
CAP1	
R/W-0	

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

Table 15-16. Capture 1 Register (CAP1) Field Descriptions

Bit	Field	Value	Description
31-0	CAP1	0-FFFF FFFFh	This register can be loaded (written) by: • Time-Stamp (i.e., counter value) during a capture event • Software - may be useful for test purposes • APRD active register when used in APWM mode

Copyright © 2013–2016, Texas Instruments Incorporated

15.4.4 Capture 2 Register (CAP2)

The capture 2 register (CAP2) is shown in Figure 15-20 and described in Table 15-17.

Figure 15-20. Capture 2 Register (CAP2)

31	0
CAP2	
R/W-0	

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

Table 15-17. Capture 2 Register (CAP2) Field Descriptions

Bit	Field	Value	Description
31-0	CAP2	0-FFFF FFFFh	This register can be loaded (written) by: • Time-Stamp (i.e., counter value) during a capture event • Software - may be useful for test purposes • ACMP active register when used in APWM mode

15.4.5 Capture 3 Register (CAP3)

The capture 3 register (CAP3) is shown in Figure 15-21 and described in Table 15-18.

Figure 15-21. Capture 3 Register (CAP3)

31	0
CAP3	
R/W-0	

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

Table 15-18. Capture 3 Register (CAP3) Field Descriptions

Bit	Field	Value	Description
31-0	CAP3	0-FFFF FFFFh	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 shadows CAP1.

Copyright © 2013–2016, Texas Instruments Incorporated

15.4.6 Capture 4 Register (CAP4)

The capture 4 register (CAP4) is shown in Figure 15-22 and described in Table 15-19.

Figure 15-22. Capture 4 Register (CAP4)

31	0
CAP4	
R/W-0	

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

Table 15-19. Capture 4 Register (CAP4) Field Descriptions

Bit	Field	Value	Description
31-0	CAP4	0-FFFF FFFFh	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 through this register. In this mode, CAP4 shadows CAP2.

15.4.7 ECAP Control Register 1 (ECCTL1)

The ECAP control register 1 (ECCTL1) is shown in Figure 15-23 and described in Table 15-20.

Figure 15-23. ECAP Control Register 1 (ECCTL1)

15	14	13	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; -*n* = value after reset

Table 15-20. ECAP Control Register 1 (ECCTL1) Field Descriptions

Bit	Field	Value	Description
15-14	FREE/SOFT	0-3h	Emulation Control
		0	TSCTR counter stops immediately on emulation suspend
		1h	TSCTR counter runs until = 0
		2h-3h	TSCTR counter is unaffected by emulation suspend (Run Free)
13-9	PRESCALE	0-1Fh	Event Filter prescale select
		0	Divide by 1 (i.e,. no prescale, by-pass the prescaler)
		1	Divide by 2
		2h	Divide by 4
		3h	Divide by 6
		4h	Divide by 8
		5h	Divide by 10
		...	
		1Eh	Divide by 60
		1Fh	Divide by 62
8	CAPLDEN		Enable Loading of CAP1-4 registers on a capture event
		0	Disable CAP1-4 register loads at capture event time.
		1	Enable CAP1-4 register loads at capture event time.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 15-20. ECAP Control Register 1 (ECCTL1) Field Descriptions (continued)

Bit	Field	Value	Description
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)

Copyright © 2013–2016, Texas Instruments Incorporated

15.4.8 ECAP Control Register 2 (ECCTL2)

The ECAP control register 2 (ECCTL2) is shown in Figure 15-24 and described in Table 15-21.

Figure 15-24. ECAP Control Register 2 (ECCTL2)

15	14	13	12	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	RE-ARM	STOP_WRAP		CONT/ONESHT
R/W-0		R/W-0	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 15-21. ECAP Control Register 2 (ECCTL2) Field Descriptions

Bit	Field	Value	Description
15-11	Reserved	0	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 • ECAP*n*/APWM*n* 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 • ECAP*n*/APWM*n* 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	0-3h	Sync-Out Select
		0	Select sync-in event to be the sync-out signal (pass through)
		1h	Select CTR = PRD event to be the sync-out signal
		2h	Disable sync out signal
		3h	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

Copyright © 2013–2016, Texas Instruments Incorporated

Table 15-21. ECAP Control Register 2 (ECCTL2) Field Descriptions (continued)

Bit	Field	Value	Description
3	RE-ARM		One-Shot Re-Arming Control, that is, 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	0-3h	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.
		0	Stop after Capture Event 1 in one-shot mode. Wrap after Capture Event 1 in continuous mode.
		1h	Stop after Capture Event 2 in one-shot mode. Wrap after Capture Event 2 in continuous mode.
		2h	Stop after Capture Event 3 in one-shot mode. Wrap after Capture Event 3 in continuous mode.
		3h	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

15.4.9 ECAP Interrupt Enable Register (ECEINT)

The ECAP interrupt enable register (ECEINT) is shown in Figure 15-25 and described in Table 15-22.

The interrupt enable bits (CEVT*n*) 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:

1. Disable global interrupts
2. Stop eCAP counter
3. Disable eCAP interrupts
4. Configure peripheral registers
5. Clear spurious eCAP interrupt flags
6. Enable eCAP interrupts
7. Start eCAP counter
8. Enable global interrupts

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 15-25. ECAP Interrupt Enable Register (ECEINT)

15							8
Reserved							
R-0							

7	6	5	4	3	2	1	0
CTR=CMP	CTR=PRD	CTROVF	CEVT4	CEVT3	CEVT2	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 15-22. ECAP Interrupt Enable Register (ECEINT) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	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	Disable 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	Enable Capture Event 4 as an Interrupt source
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	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

15.4.10 ECAP Interrupt Flag Register (ECFLG)

The ECAP interrupt flag register (ECFLG) is shown in Figure 15-26 and described in Table 15-23.

Figure 15-26. 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 = Read only; -*n* = value after reset

Table 15-23. ECAP Interrupt Flag Register (ECFLG) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved
7	CTR=CMP		Compare Equal Compare Status Flag. This flag is only active 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 0xFFFFFFFF to 0x00000000
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 ECAP*n* 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 ECAP*n* 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 ECAP*n* 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 ECAP*n* pin.
0	INT		Global Interrupt Status Flag
		0	Indicates no interrupt generated.
		1	Indicates that an interrupt was generated.

Copyright © 2013–2016, Texas Instruments Incorporated

15.4.11 ECAP Interrupt Clear Register (ECCLR)

The ECAP interrupt clear register (ECCLR) is shown in Figure 15-27 and described in Table 15-24.

Figure 15-27. 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

Table 15-24. ECAP Interrupt Clear Register (ECCLR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved
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 CEVT3 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
		0	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.

Copyright © 2013–2016, Texas Instruments Incorporated

15.4.12 ECAP Interrupt Forcing Register (ECFRC)

The ECAP interrupt forcing register (ECFRC) is shown in Figure 15-28 and described in Table 15-25.

Figure 15-28. 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 15-25. ECAP Interrupt Forcing Register (ECFRC) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved
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.
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
		0	No effect. Always reads back a 0.
		1	Writing a 1 sets the CEVT1 flag bit.
0	Reserved	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

15.4.13 Revision ID Register (REVID)

The revision ID register (REVID) is shown in Figure 15-29 and described in Table 15-26.

Figure 15-29. Revision ID Register (REVID)

31	0
REV	
R-44D2 2100h	

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

Table 15-26. Revision ID Register (REVID) Field Descriptions

Bit	Field	Value	Description
31-0	REV	44D2 2100h	Revision ID.

Copyright © 2013–2016, Texas Instruments Incorporated

Enhanced High-Resolution Pulse-Width Modulator (eHRPWM)

This chapter describes the enhanced high-resolution pulse-width modulator (eHRPWM).

Topic **Page**

16.1 Introduction 448
16.2 Architecture 453
16.3 Applications to Power Topologies 512
16.4 Registers 536

Copyright © 2013–2016, Texas Instruments Incorporated

16.1 Introduction

16.1.1 Introduction

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 and 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.

In this chapter, 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.

16.1.2 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 16-1. 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 Section 16.2.10. See your 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:

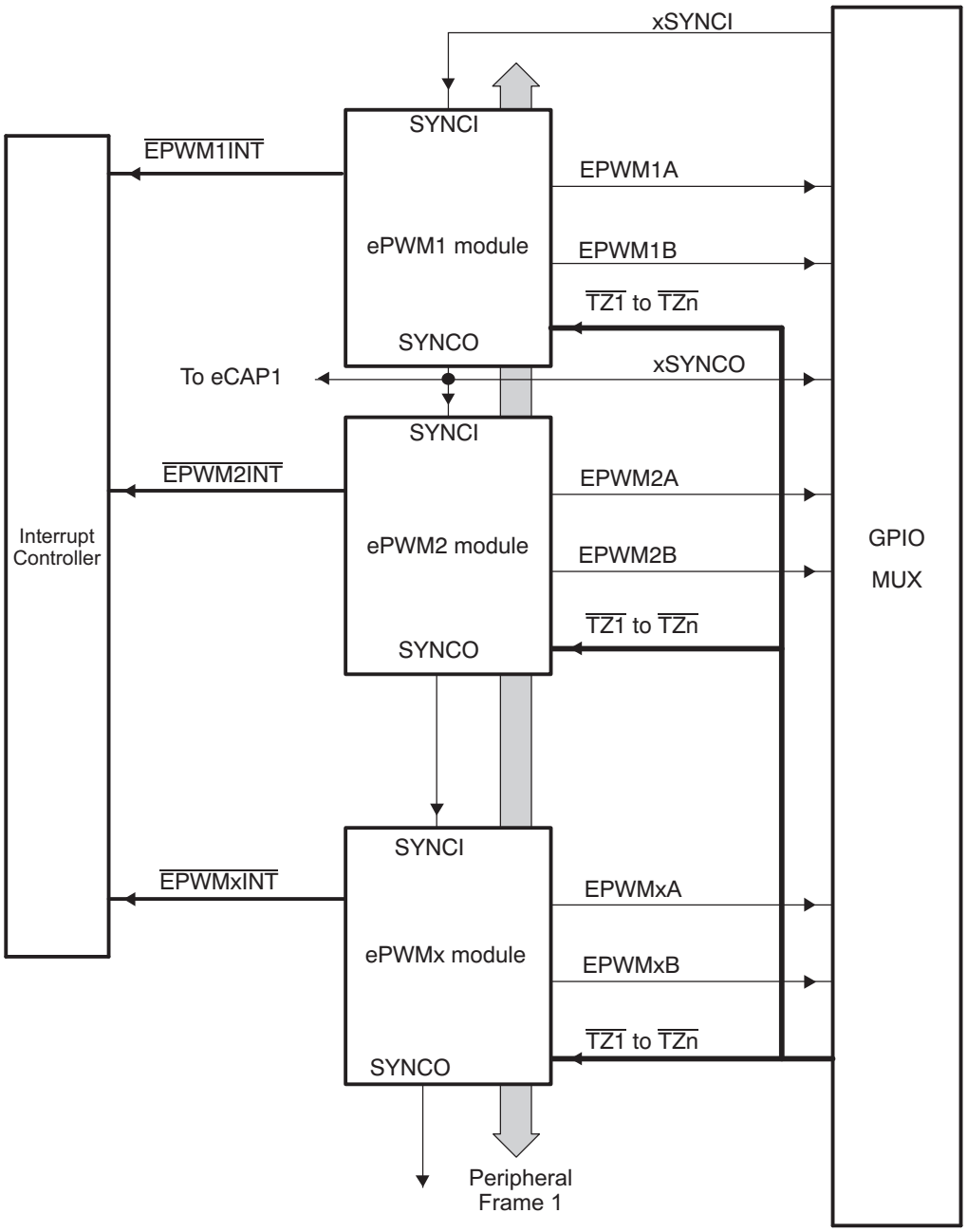
- Dedicated 16-bit time-base counter with period and frequency control
- Two PWM outputs (EPWMxA and EPWMxB) that can be used in the following configurations::
  - Two independent PWM outputs with single-edge operation
  - Two independent PWM outputs with dual-edge symmetric operation
  - One independent 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.
- 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 16-1. The signals are described in detail in subsequent sections.

The order in which the ePWM modules are connected may differ from what is shown in Figure 16-1. See Section 16.2.3.3.2 for the synchronization scheme for a particular device. Each ePWM module consists of seven submodules and is connected within a system via the signals shown in Figure 16-2.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 16-1. Multiple ePWM Modules



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 16-2. Submodules and Signal Connections for an ePWM Module

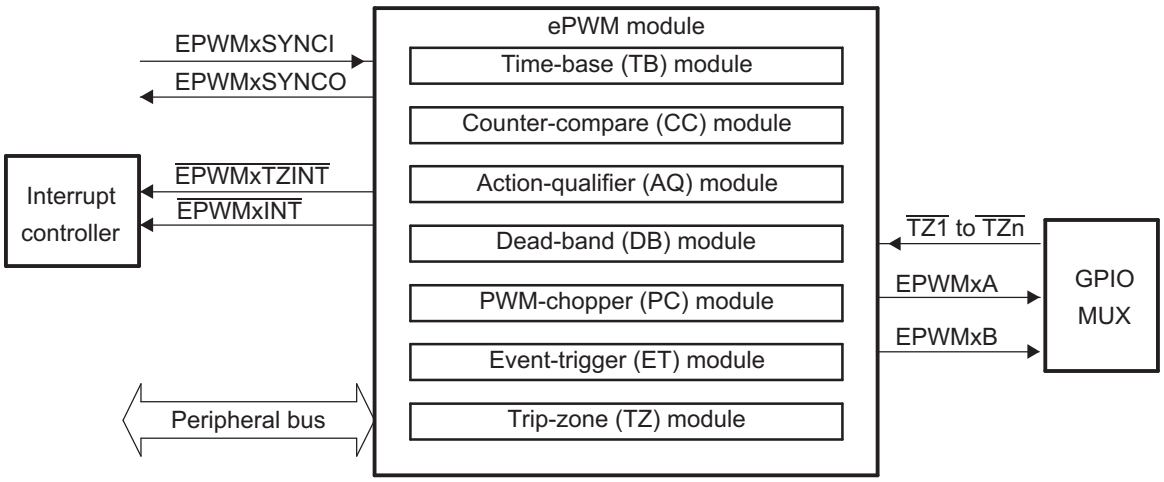


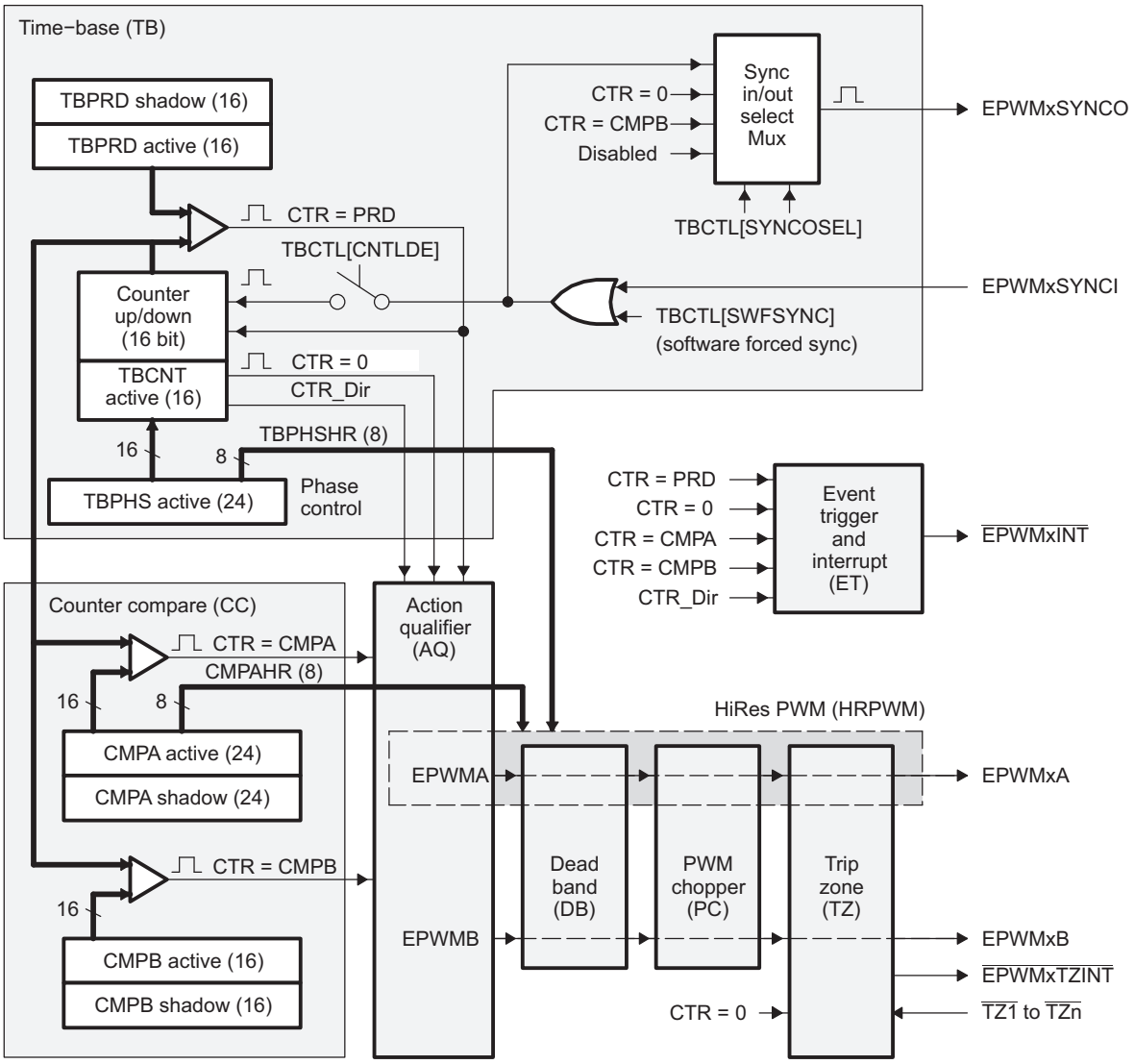
Figure 16-3 shows more internal details of a single ePWM module. The main signals used by the ePWM module are:

- **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 ($\overline{\text{TZ1}}$ to $\overline{\text{TZn}}$).** These input signals alert the ePWM module of an external fault condition. Each module on a device can be configured to either use or ignore any of the trip-zone signals. The trip-zone signal can be configured as an asynchronous input through the GPIO peripheral. See your device-specific data manual to determine how many trip-zone pins are available in the device.
- **Time-base synchronization input (EPWMxSYNCI) and output (EPWMxSYNCO) signals.** The synchronization signals daisy chain the ePWM modules together. Each module can be configured 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).
- **Peripheral Bus.** The peripheral bus is 32-bits wide and allows both 16-bit and 32-bit writes to the ePWM register file.

Figure 16-3 also shows the key internal submodule interconnect signals. Each submodule is described in detail in Section 16.2.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 16-3. ePWM Submodules and Critical Internal Signal Interconnects



Copyright © 2013–2016, Texas Instruments Incorporated

16.1.3 Register Mapping

Table 16-1 shows the complete ePWM module control and status register set grouped by submodule. 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.

Table 16-1. ePWM Module Control and Status Registers Grouped by Submodule

Acronym	Offset[1]	Size (×16)	Shadow	Register Description
Time-Base Submodule Registers				
TBCTL	0h	1	No	Time-Base Control Register
TBSTS	2h	1	No	Time-Base Status Register
TBPHSHR	4h	1	No	Extension for HRPWM Phase Register [2]
TBPHS	6h	1	No	Time-Base Phase Register
TBCNT	8h	1	No	Time-Base Counter Register
TBPRD	Ah	1	Yes	Time-Base Period Register
Counter-Compare Submodule Registers				
CMPCTL	Eh	1	No	Counter-Compare Control Register
CMPAHR	10h	1	No	Extension for HRPWM Counter-Compare A Register [2]
CMPA	12h	1	Yes	Counter-Compare A Register
CMPB	14h	1	Yes	Counter-Compare B Register
Action-Qualifier Submodule Registers				
AQCTLA	16h	1	No	Action-Qualifier Control Register for Output A (EPWMxA)
AQCTLB	18h	1	No	Action-Qualifier Control Register for Output B (EPWMxB)
AQSFRC	1Ah	1	No	Action-Qualifier Software Force Register
AQCSFRC	1Ch	1	Yes	Action-Qualifier Continuous S/W Force Register Set
Dead-Band Generator Submodule Registers				
DBCTL	1Eh	1	No	Dead-Band Generator Control Register
DBRED	20h	1	No	Dead-Band Generator Rising Edge Delay Count Register
DBFED	22h	1	No	Dead-Band Generator Falling Edge Delay Count Register
PWM-Chopper Submodule Registers				
PCCTL	3Ch	1	No	PWM-Chopper Control Register
Trip-Zone Submodule Registers				
TZSEL	24h	1	No	Trip-Zone Select Register
TZCTL	28h	1	No	Trip-Zone Control Register
TZEINT	2Ah	1	No	Trip-Zone Enable Interrupt Register
TZFLG	2Ch	1	No	Trip-Zone Flag Register
TZCLR	2Eh	1	No	Trip-Zone Clear Register
TZFRC	30h	1	No	Trip-Zone Force Register
Event-Trigger Submodule Registers				
ETSEL	32h	1	No	Event-Trigger Selection Register
ETPS	34h	1	No	Event-Trigger Pre-Scale Register
ETFLG	36h	1	No	Event-Trigger Flag Register
ETCLR	38h	1	No	Event-Trigger Clear Register
ETFRC	3Ah	1	No	Event-Trigger Force Register
High-Resolution PWM (HRPWM) Submodule Registers				
HRCNFG	1040h	1	No	HRPWM Configuration Register [2]

(1) Locations not shown are reserved.

(2) These registers are only available on ePWM instances that include the high-resolution PWM (HRPWM) extension; otherwise, these locations are reserved. See your device-specific data manual to determine which instances include the HRPWM.

Copyright © 2013–2016, Texas Instruments Incorporated

16.2 Architecture

Seven submodules are included in every ePWM peripheral. There are some instances that include a high-resolution submodule that allows more precise control of the PWM outputs. Each of these submodules performs specific tasks that can be configured by software.

16.2.1 Overview

Table 16-2 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 16.2.4 for relevant details.

Table 16-2. Submodule Configuration Parameters

Submodule	Configuration Parameter or Option	Reference
Time-base (TB)	• Scale the time-base clock (TBCLK) relative to the system clock (SYSCLKOUT). • Configure the PWM time-base counter (TBCNT) 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. • 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.	Section 16.2.3
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	Section 16.2.4
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	Section 16.2.5
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.	Section 16.2.6
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.	Section 16.2.7

Copyright © 2013–2016, Texas Instruments Incorporated

Table 16-2. Submodule Configuration Parameters (continued)

Submodule	Configuration Parameter or Option	Reference
Trip-zone (TZ)	• Configure the ePWM module to react to one, all, or none of the trip-zone pins. • 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 pin: – One-shot – Cycle-by-cycle • Enable the trip-zone to initiate an interrupt. • Bypass the trip-zone module entirely.	Section 16.2.8
Event-trigger (ET)	• Enable the ePWM events that will trigger an interrupt. • Specify the rate at which events cause triggers (every occurrence or every second or third occurrence) • Poll, set, or clear event flags	Section 16.2.9
High-Resolution PWM (HRPWM)	• Enable extended time resolution capabilities • Configure finer time granularity control or edge positioning	Section 16.2.10

Copyright © 2013–2016, Texas Instruments Incorporated

Code examples are provided in the remainder of this chapter that show how to implement various ePWM module configurations. These examples use the constant definitions shown in Example 16-1.

Example 16-1. Constant Definitions Used in the Code Examples

```
// TBCTL (Time-Base Control)
// = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
// TBCNT MODE bit
#define         TB_COUNT_UP        0x0
#define         TB_COUNT_DOWN      0x1
#define         TB_COUNT_UPDOWN    0x2
#define         TB_FREEZE          0x3
// PHSEN bit
#define         TB_DISABLE         0x0
#define         TB_ENABLE          0x1
// PRDLD bit
#define         TB_SHADOW          0x0
#define         TB_IMMEDIATE       0x1
// SYNCOSEL bit
#define         TB_SYNC_IN         0x0
#define         TB_CTR_ZERO        0x1
#define         TB_CTR_CMPB        0x2
#define         TB_SYNC_DISABLE    0x3
// HSPCLKDIV and CLKDIV bits
#define         TB_DIV1            0x0
#define         TB_DIV2            0x1
#define         TB_DIV4            0x2
// PHSDIR bit
#define         TB_DOWN            0x0
#define         TB_UP              0x1
// CMPCTL (Compare Control)
// = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
// LOADAMODE and LOADBMODE bits
#define         CC_CTR_ZERO        0x0
#define         CC_CTR_PRD         0x1
#define         CC_CTR_ZERO_PRD    0x2
#define         CC_LD_DISABLE      0x3
// SHDWAMODE and SHDWBMODE bits
#define         CC_SHADOW          0x0
#define         CC_IMMEDIATE       0x1
// AQCTLA and AQCTLB (Action-qualifier Control)
// = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
// ZRO, PRD, CAU, CAD, CBU, CBD bits
#define         AQ_NO_ACTION       0x0
#define         AQ_CLEAR           0x1
#define         AQ_SET             0x2
#define         AQ_TOGGLE          0x3
// DBCTL (Dead-Band Control)
// = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
// MODE bit
#define         DB_DISABLE         0x0
#define         DBA_ENABLE         0x1
#define         DBB_ENABLE         0x2
#define         DB_FULL_ENABLE     0x3
// POLSEL bit
#define         DB_ACTV_HI         0x0
#define         DB_ACTV_LOC        0x1
#define         DB_ACTV_HIC        0x2
#define         DB_ACTV_LO         0x3
// PCCTL (chopper control)
// = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
// CHPEN bit
#define         CHP_DISABLE        0x0
#define         CHP_ENABLE         0x1
```


Copyright © 2013–2016, Texas Instruments Incorporated

Example 16-1. Constant Definitions Used in the Code Examples (continued)

```
// CHPFREQ bit
#define         CHP_DIV1           0x0
#define         CHP_DIV2           0x1
#define         CHP_DIV3           0x2
#define         CHP_DIV4           0x3
#define         CHP_DIV5           0x4
#define         CHP_DIV6           0x5
#define         CHP_DIV7           0x6
#define         CHP_DIV8           0x7
// CHPDUTY bit
#define         CHP1_8TH           0x0
#define         CHP2_8TH           0x1
#define         CHP3_8TH           0x2
#define         CHP4_8TH           0x3
#define         CHP5_8TH           0x4
#define         CHP6_8TH           0x5
#define         CHP7_8TH           0x6
// TZSEL (Trip-zone Select)
// = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
// CBCn and OSHTn bits
#define         TZ_DISABLE         0x0
#define         TZ_ENABLE          0x1
// TZCTL (Trip-zone Control)
// = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
// TZA and TZB bits
#define         TZ_HIZ             0x0
#define         TZ_FORCE_HI        0x1
#define         TZ_FORCE_LO        0x2
#define         TZ_NONE            0x3
// ETSEL (Event-trigger Select)
// = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
// INTSEL bit
#define         ET_CTR_ZERO        0x1
#define         ET_CTR_PRD         0x2
#define         ET_CTRU_CMPA       0x4
#define         ET_CTRD_CMPA       0x5
#define         ET_CTRU_CMPB       0x6
#define         ET_CTRD_CMPB       0x7
// ETPS (Event-trigger Prescale)
// = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
// INTPRD bit
#define         ET_DISABLE         0x0
#define         ET_1ST             0x1
#define         ET_2ND             0x2
#define         ET_3RD             0x3
```

16.2.2 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:

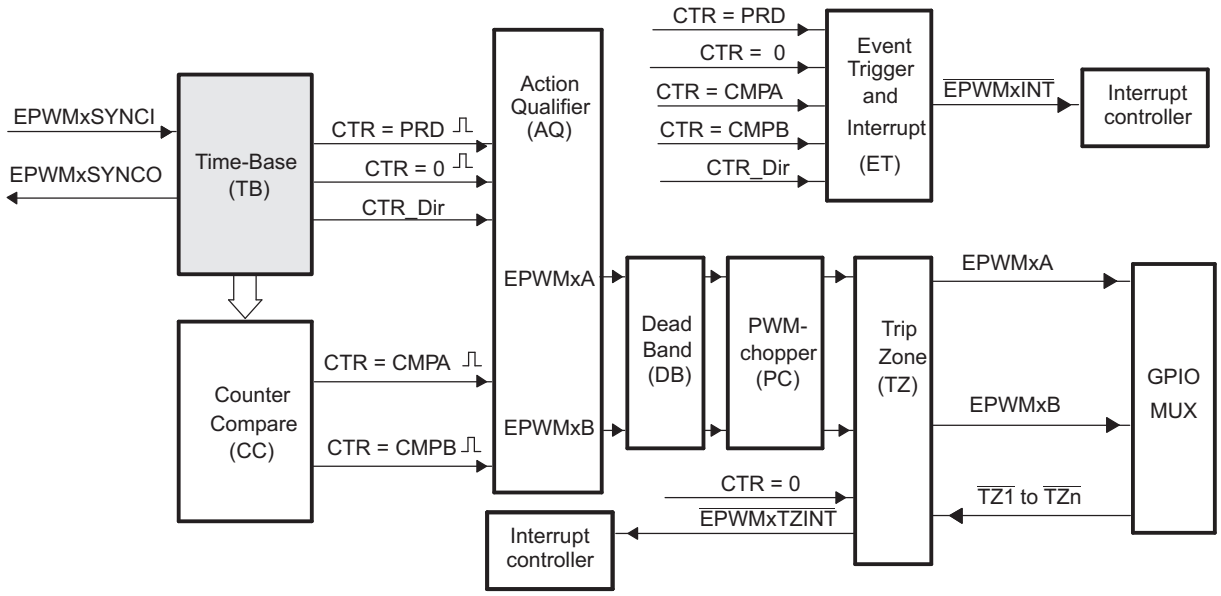
1. Disable global interrupts (CPU INTM flag)
2. Disable ePWM interrupts
3. Initialize peripheral registers
4. Clear any spurious ePWM flags
5. Enable ePWM interrupts
6. Enable global interrupts

Copyright © 2013–2016, Texas Instruments Incorporated

16.2.3 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 16-4 illustrates the time-base module's place within the ePWM.

Figure 16-4. Time-Base Submodule Block Diagram



16.2.3.1 Purpose of the Time-Base Submodule

You can configure the time-base submodule for the following:

- Specify the ePWM time-base counter (TBCNT) 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 (TBCNT = TBPRD) .
  - CTR = 0: Time-base counter equal to zero (TBCNT = 0000h).
- 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 © 2013–2016, Texas Instruments Incorporated

16.2.3.2 Controlling and Monitoring the Time-Base Submodule

Table 16-3 lists the registers used to control and monitor the time-base submodule.

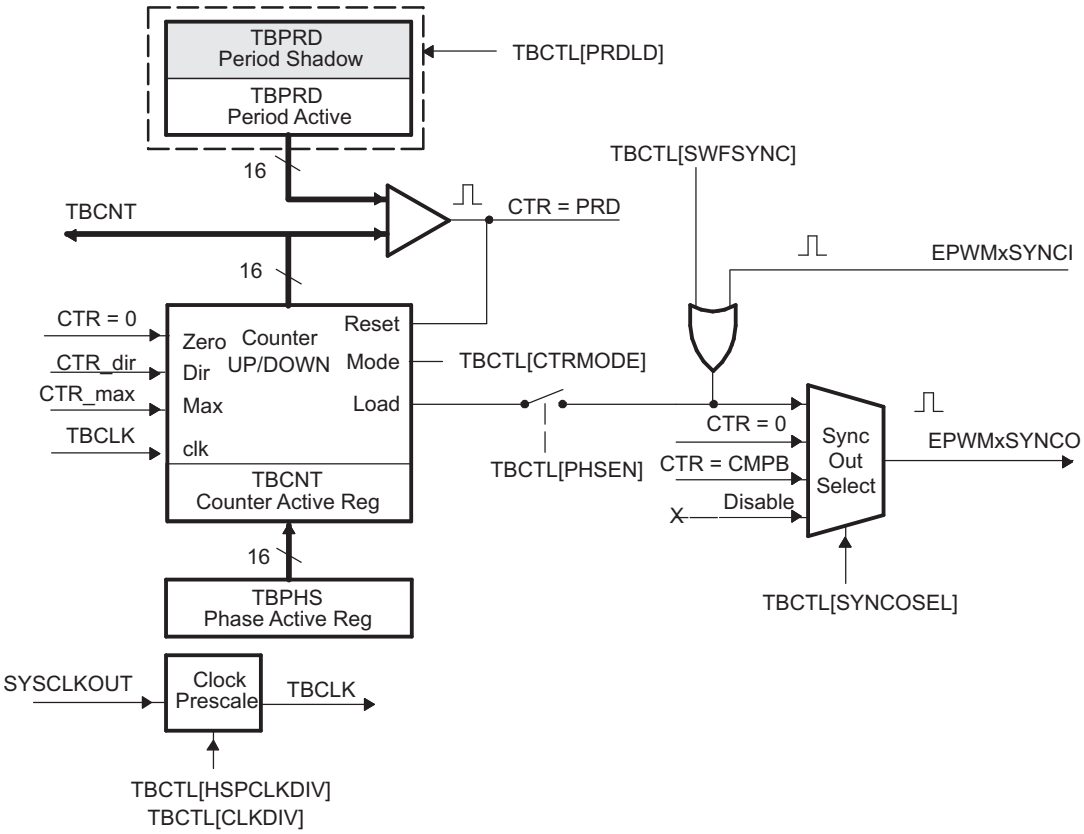
Table 16-3. Time-Base Submodule Registers

Acronym	Register Description	Address Offset	Shadowed
TBCTL	Time-Base Control Register	0h	No
TBSTS	Time-Base Status Register	2h	No
TBPHSHR	HRPWM extension Phase Register (1)	4h	No
TBPHS	Time-Base Phase Register	6h	No
TBCNT	Time-Base Counter Register	8h	No
TBPRD	Time-Base Period Register	Ah	Yes

(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. See your device-specific data manual to determine which ePWM instances include this feature.

Figure 16-5 shows the critical signals and registers of the time-base submodule. Table 16-4 provides descriptions of the key signals associated with the time-base submodule.

Figure 16-5. Time-Base Submodule Signals and Registers



Copyright © 2013–2016, Texas Instruments Incorporated

Table 16-4. 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. 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 16.2.3.3.2 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 = 0: The time-base counter equal to zero (TBCNT = 0000h). 3. CTR = CMPB: The time-base counter equal to the counter-compare B (TBCNT = 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 TBCNT = TBPRD.
CTR = 0	Time-base counter equal to zero.
	This signal is generated whenever the counter value is zero. That is when TBCNT equals 0000h.
CTR = CMPB	Time-base counter equal to active counter-compare B register (TBCNT = 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. (TBCNT = FFFFh)
	Generated event when the TBCNT 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.

16.2.3.3 Calculating PWM Period and Frequency

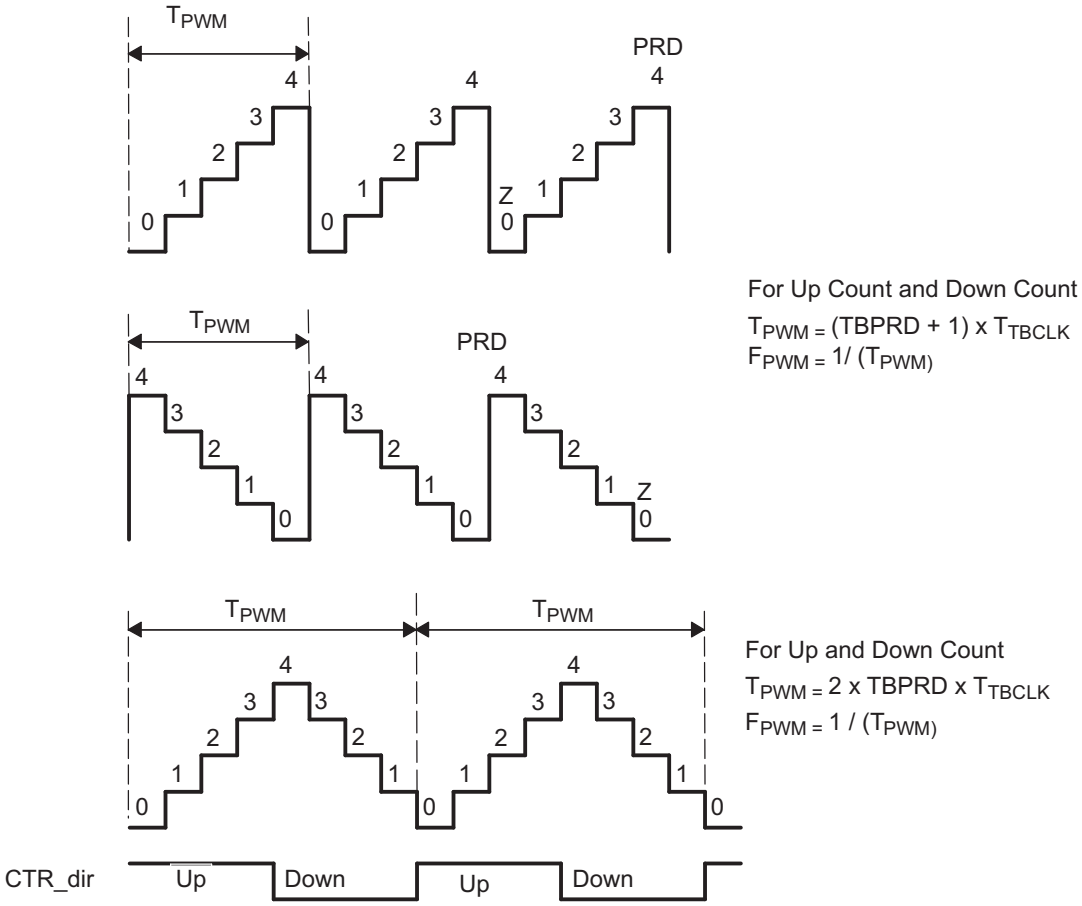
The frequency of PWM events is controlled by the time-base period (TBPRD) register and the mode of the time-base counter. Figure 16-6 shows the period (T_{pwm}) and frequency (F_{pwm}) relationships for the up-count, down-count, and up-down-count time-base counter modes when when the period is set to 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 © 2013–2016, Texas Instruments Incorporated

Figure 16-6. Time-Base Frequency and Period



16.2.3.3.1 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:

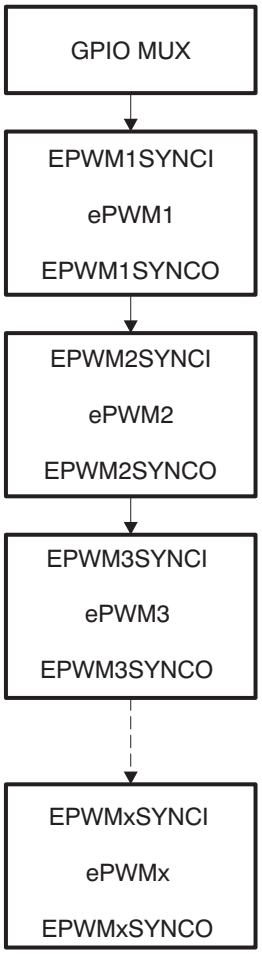
- **Time-Base Period Shadow Mode:** The TBPRD shadow register is enabled when TBCTL[PRDLD] = 0. Reads from and writes to the TBPRD memory address go to the shadow register. The shadow register contents are transferred to the active register (TBPRD (Active) ← TBPRD (shadow)) when the time-base counter equals zero (TBCNT = 0000h). By default the TBPRD shadow register is enabled.
- **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.

Copyright © 2013–2016, Texas Instruments Incorporated

16.2.3.3.2 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. The possible synchronization connections for the remaining ePWM modules is shown in Figure 16-7.

Figure 16-7. Time-Base Counter Synchronization Scheme 1



Copyright © 2013–2016, Texas Instruments Incorporated

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 (TBCNT) 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 → TBCNT). This operation occurs on the next valid time-base clock (TBCLK) edge.
- **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.

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 TBPHS bit is ignored in count-up or count-down modes. See Figure 16-8 through Figure 16-11 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.

16.2.3.4 Phase Locking the Time-Base Clocks of Multiple ePWM Modules

The TBCLKSYNC bit in the chip configuration register 1 (CFGCHIP1) in the System Module can be used to globally synchronize the time-base clocks of all enabled ePWM modules on a device. The TBCLKSYNC bit is part of the chip configuration registers and is described in the device-specific data 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 ePWM module clocks.
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.

16.2.3.5 Time-Base Counter Modes and Timing Waveforms

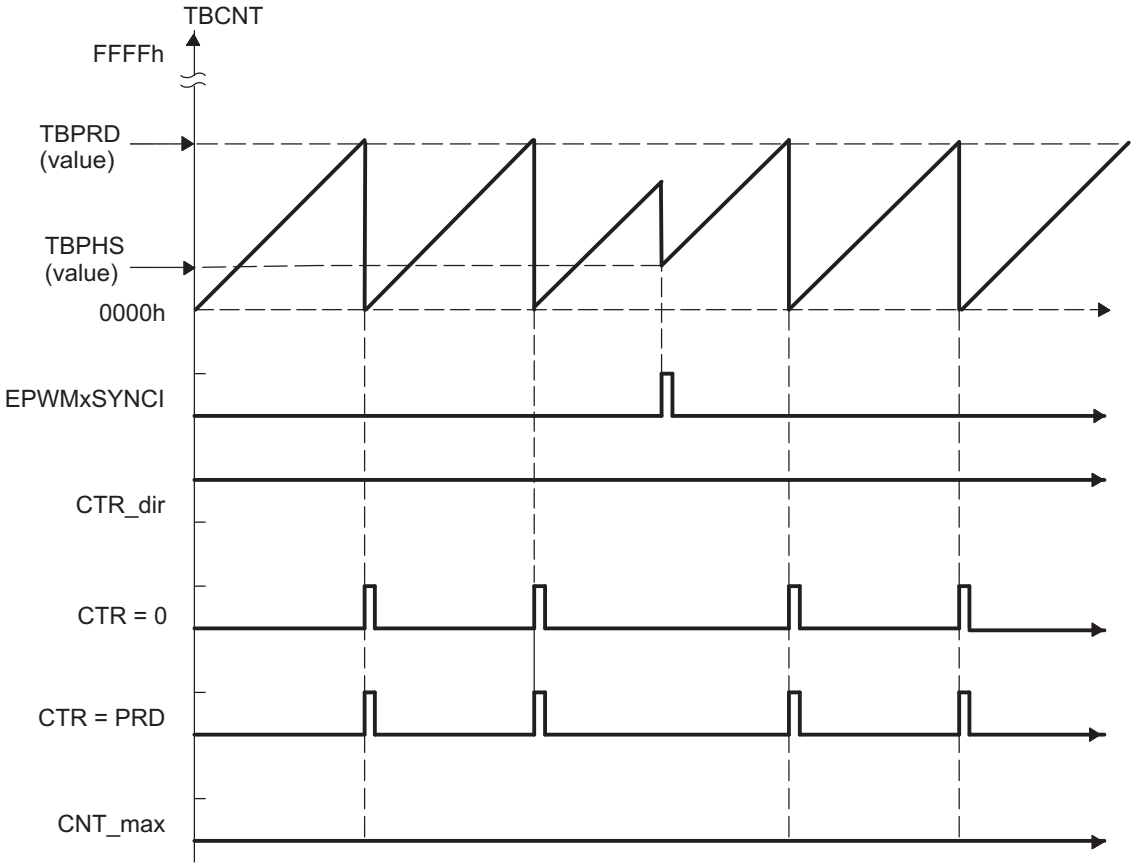
The time-base counter operates in one of four modes:

- 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, Figure 16-8 to Figure 16-11 show when events are generated and how the time-base responds to an EPWMxSYNCI signal.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 16-8. Time-Base Up-Count Mode Waveforms



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 16-9. Time-Base Down-Count Mode Waveforms

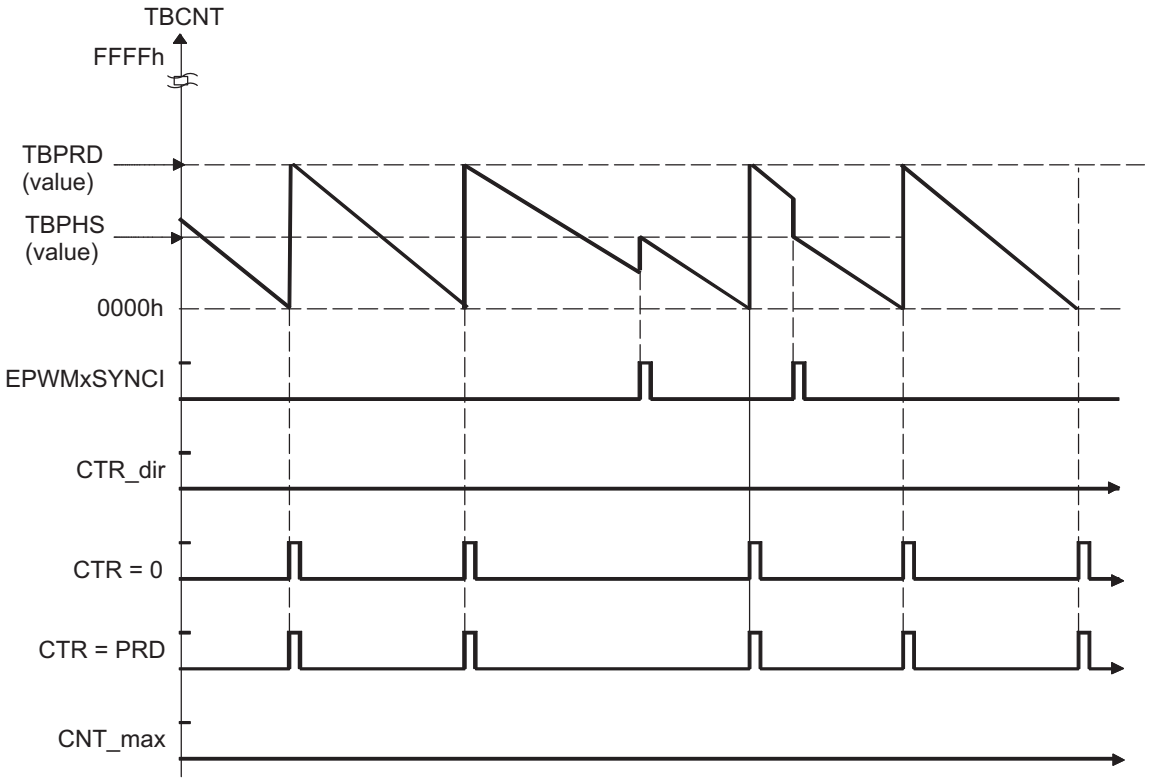
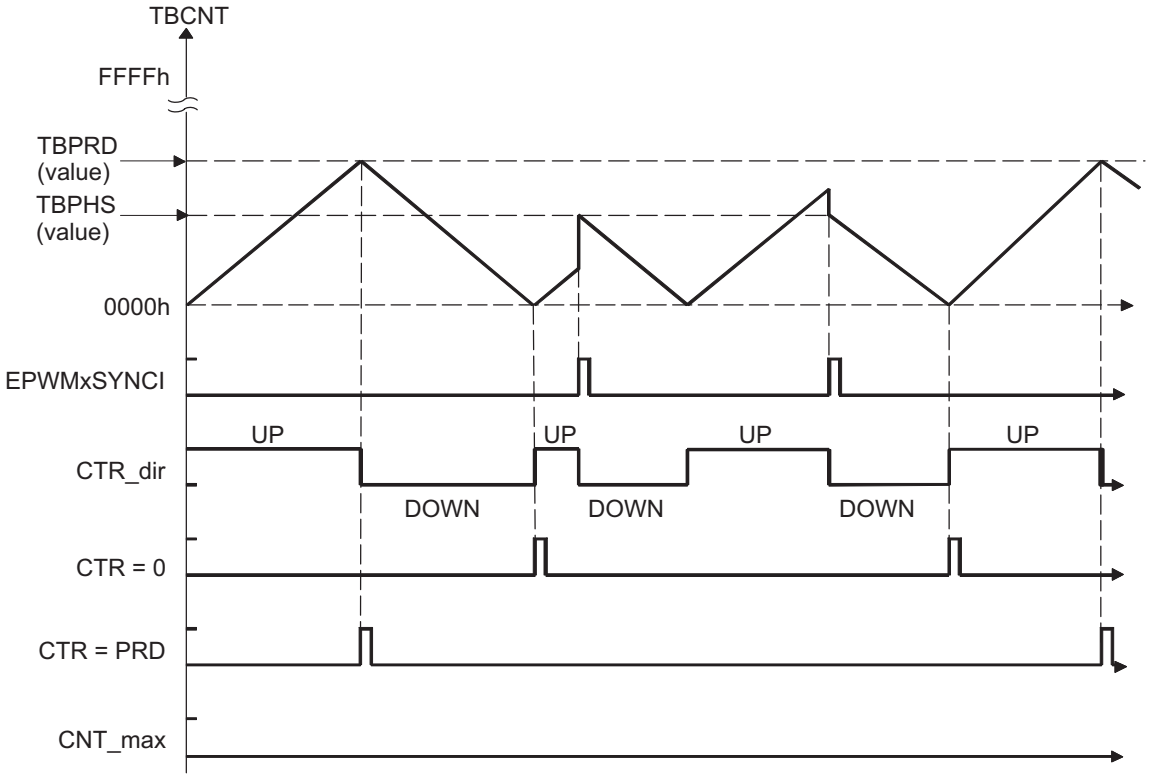
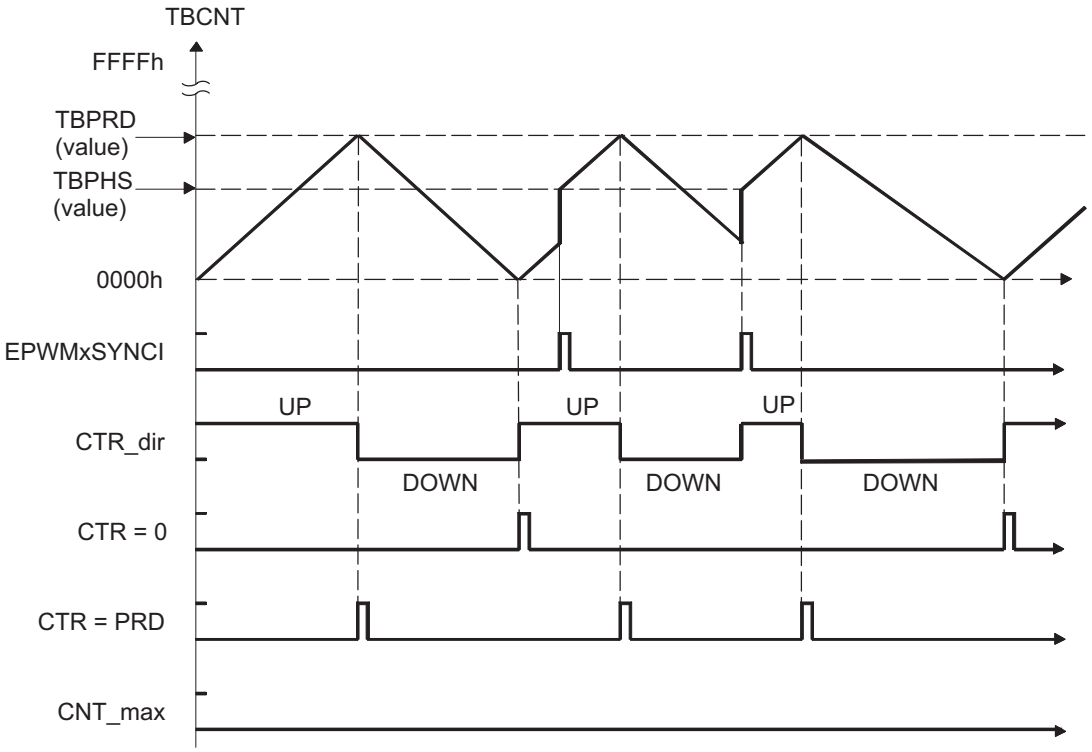


Figure 16-10. Time-Base Up-Down-Count Waveforms, TBCTL[PHSDIR = 0] Count Down on Synchronization Event



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 16-11. Time-Base Up-Down Count Waveforms, TBCTL[PHSDIR = 1] Count Up on Synchronization Event



Copyright © 2013–2016, Texas Instruments Incorporated

16.2.4 Counter-Compare (CC) Submodule

Figure 16-12 illustrates the counter-compare submodule within the ePWM. Figure 16-13 shows the basic structure of the counter-compare submodule.

Figure 16-12. Counter-Compare Submodule

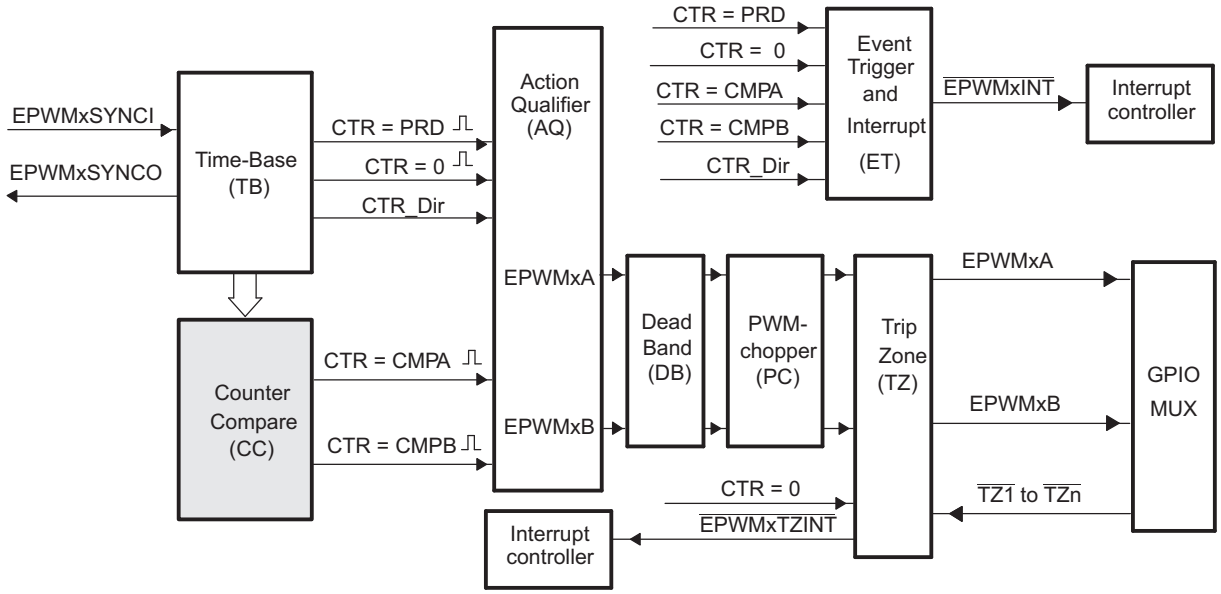
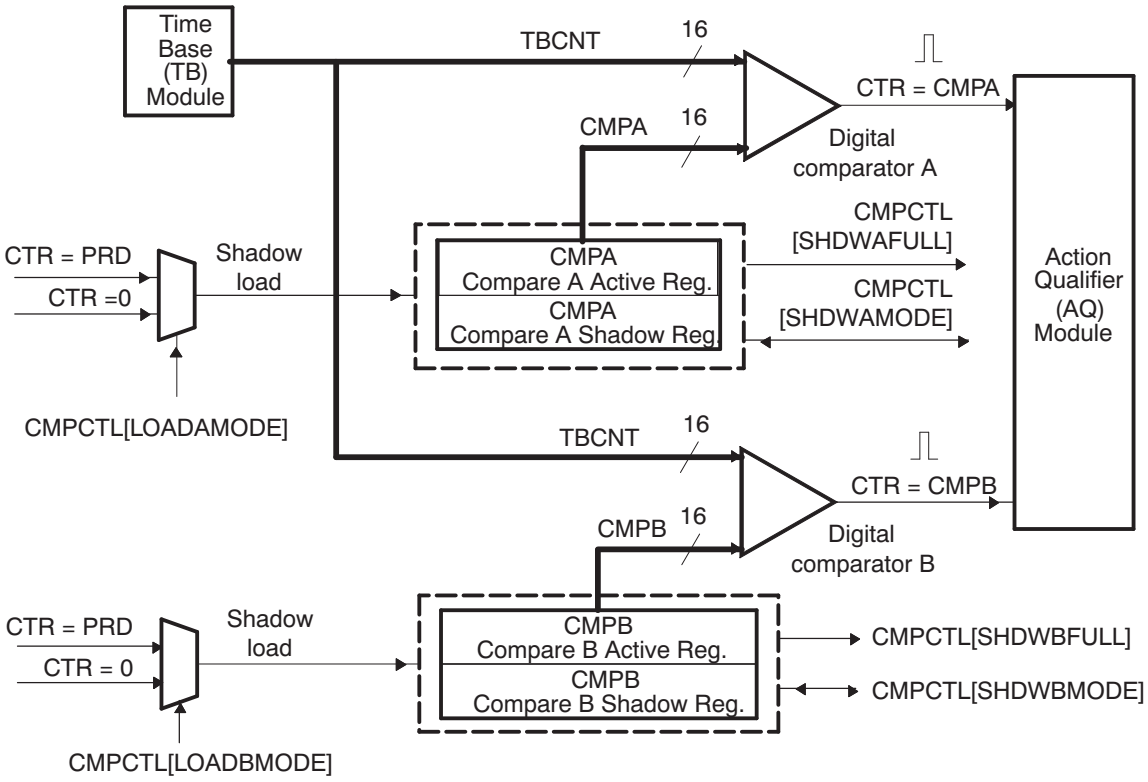


Figure 16-13. Counter-Compare Submodule Signals and Registers



Copyright © 2013–2016, Texas Instruments Incorporated

16.2.4.1 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) and counter-compare B (CMPB) 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 submodule:

- Generates events based on programmable time stamps using the CMPA and CMPB registers
  - CTR = CMPA: Time-base counter equals counter-compare A register (TBCNT = CMPA).
  - CTR = CMPB: Time-base counter equals counter-compare B register (TBCNT = CMPB)
- Controls the PWM duty cycle if the action-qualifier submodule is configured appropriately
- Shadows new compare values to prevent corruption or glitches during the active PWM cycle

16.2.4.2 Controlling and Monitoring the Counter-Compare Submodule

Table 16-5 lists the registers used to control and monitor the counter-compare submodule. Table 16-6 lists the key signals associated with the counter-compare submodule.

Table 16-5. Counter-Compare Submodule Registers

Acronym	Register Description	Address Offset	Shadowed
CMPCTL	Counter-Compare Control Register.	Eh	No
CMPAHR	HRPWM Counter-Compare A Extension Register (1)	10h	Yes
CMPA	Counter-Compare A Register	12h	Yes
CMPB	Counter-Compare B Register	14h	Yes

(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. Refer to the device-specific data manual to determine which ePWM instances include this feature.

Table 16-6. Counter-Compare Submodule Key Signals

Signal	Description of Event	Registers Compared
CTR = CMPA	Time-base counter equal to the active counter-compare A value	TBCNT = CMPA
CTR = CMPB	Time-base counter equal to the active counter-compare B value	TBCNT = CMPB
CTR = PRD	Time-base counter equal to the active period. Used to load active counter-compare A and B registers from the shadow register	TBCNT = TBPRD
CTR = 0	Time-base counter equal to zero. Used to load active counter-compare A and B registers from the shadow register	TBCNT = 0000h

Copyright © 2013–2016, Texas Instruments Incorporated

16.2.4.3 Operational Highlights for the Counter-Compare Submodule

The counter-compare submodule is responsible for generating two independent compare events based on two compare registers:

1. CTR = CMPA: Time-base counter equal to counter-compare A register (TBCNT = CMPA).
2. CTR = CMPB: Time-base counter equal to counter-compare B register (TBCNT = 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 0000h and TBPRD; and occurs once per cycle, if the compare value is equal to 0000h 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 16.2.5.1 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 occurs 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:

  – CTR = PRD: Time-base counter equal to the period (TBCNT = TBPRD).
  – CTR = 0: Time-base counter equal to zero (TBCNT = 0000h)
  – Both CTR = PRD and CTR = 0

  Which of these three events is specified by the CMPCTL[LOADAMODE] and CMPCTL[LOADBMODE] register bits. 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 (TBCTL[SHADWAMODE] = 1 or TBCTL[SHADWBMODE] = 1), then a read from or a write to the register will go directly to the active register.

16.2.4.4 Count Mode Timing Waveforms

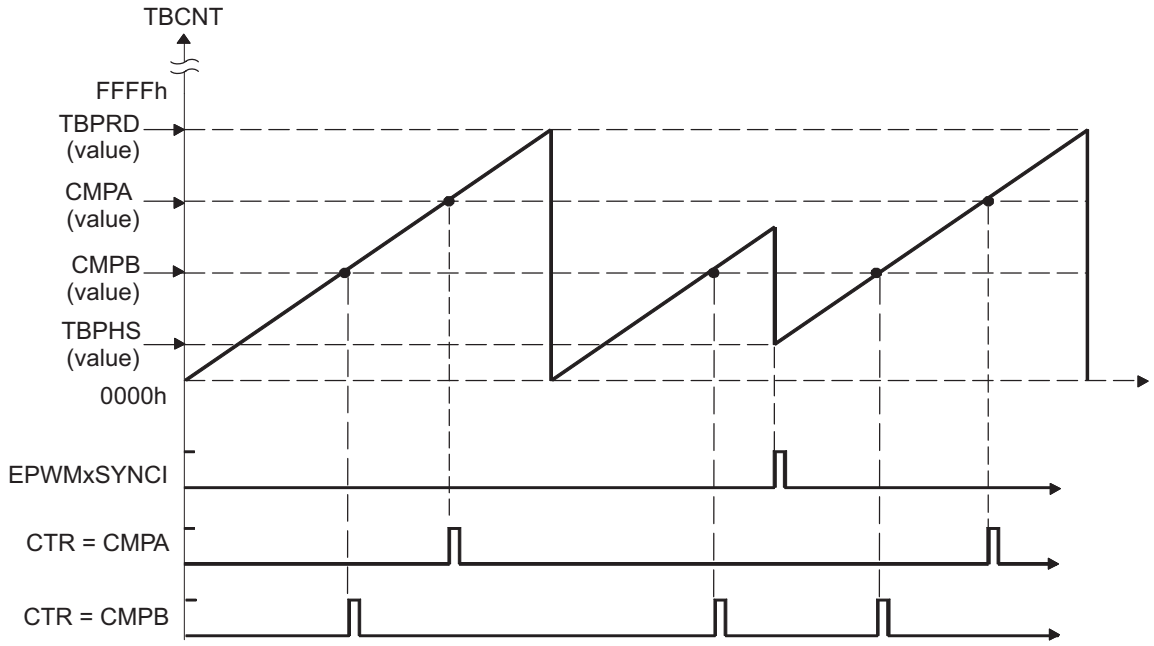
The counter-compare module can generate compare events in all three count modes:

- 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 16-14 to Figure 16-17 show when events are generated and how the EPWMxSYNCI signal interacts.

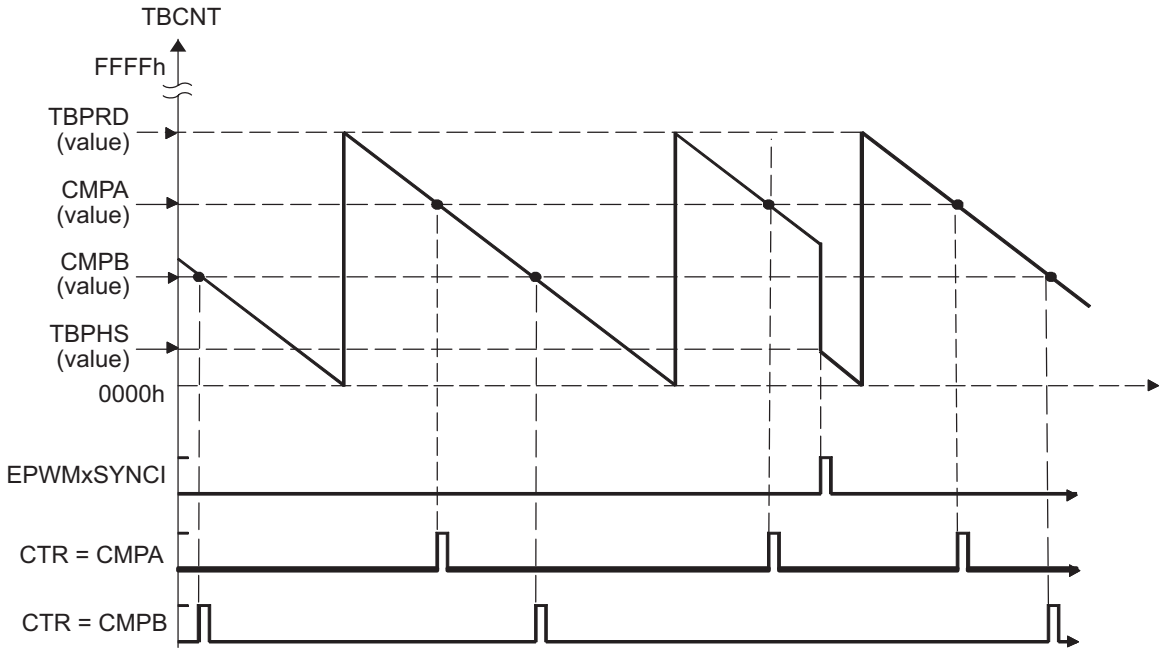
Copyright © 2013–2016, Texas Instruments Incorporated

Figure 16-14. Counter-Compare Event Waveforms in Up-Count Mode



NOTE: An EPWMxSYNCI external synchronization event can cause a discontinuity in the TBCNT count sequence. This can lead to a compare event being skipped. This skipping is considered normal operation and must be taken into account.

Figure 16-15. Counter-Compare Events in Down-Count Mode



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 16-16. Counter-Compare Events in Up-Down-Count Mode, TBCTL[PHSDIR = 0] Count Down on Synchronization Event

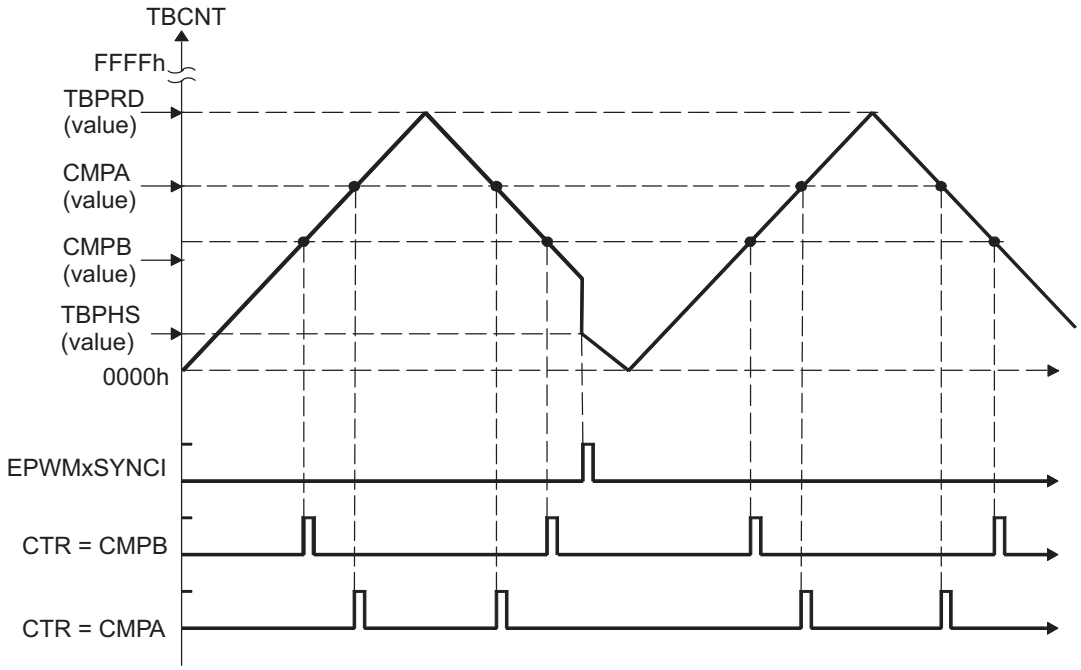
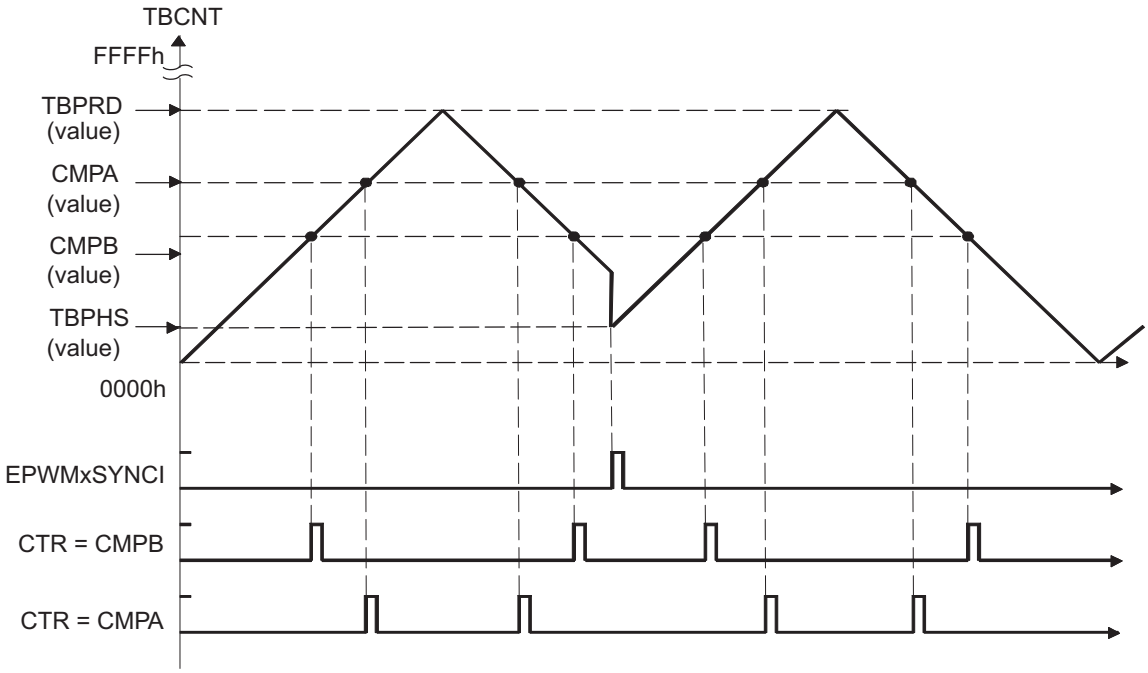


Figure 16-17. Counter-Compare Events in Up-Down-Count Mode, TBCTL[PHSDIR = 1] Count Up on Synchronization Event

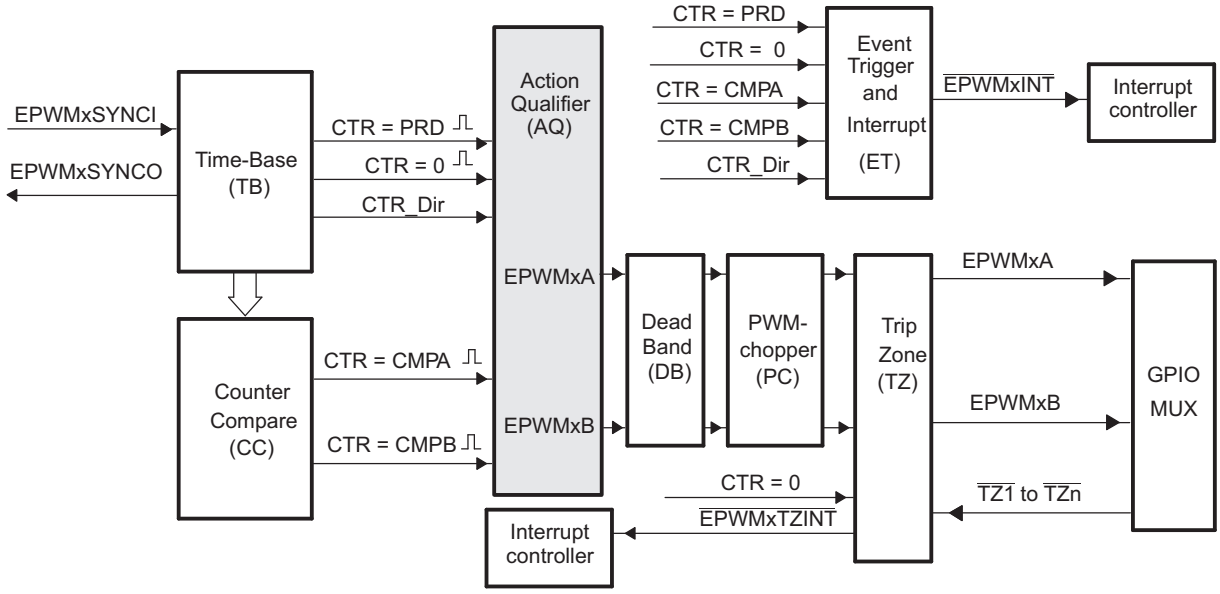


Copyright © 2013–2016, Texas Instruments Incorporated

16.2.5 Action-Qualifier (AQ) Submodule

Figure 16-18 shows the action-qualifier (AQ) submodule (see shaded block) in the ePWM system. 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.

Figure 16-18. Action-Qualifier Submodule



16.2.5.1 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 (TBCNT = TBPRD)
  - CTR = 0: Time-base counter equal to zero (TBCNT = 0000h)
  - CTR = CMPA: Time-base counter equal to the counter-compare A register (TBCNT = CMPA)
  - CTR = CMPB: Time-base counter equal to the counter-compare B register (TBCNT = 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.

16.2.5.2 Controlling and Monitoring the Action-Qualifier Submodule

Table 16-7 lists the registers used to control and monitor the action-qualifier submodule.

Table 16-7. Action-Qualifier Submodule Registers

Acronym	Register Description	Address Offset	Shadowed
AQCTLA	Action-Qualifier Control Register For Output A (EPWMxA)	16h	No
AQCTLB	Action-Qualifier Control Register For Output B (EPWMxB)	18h	No
AQSFRC	Action-Qualifier Software Force Register	1Ah	No
AQCSFRC	Action-Qualifier Continuous Software Force	1Ch	Yes

Copyright © 2013–2016, Texas Instruments Incorporated

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 Figure 16-19. The possible input events are summarized again in Table 16-8.

Figure 16-19. Action-Qualifier Submodule Inputs and Outputs

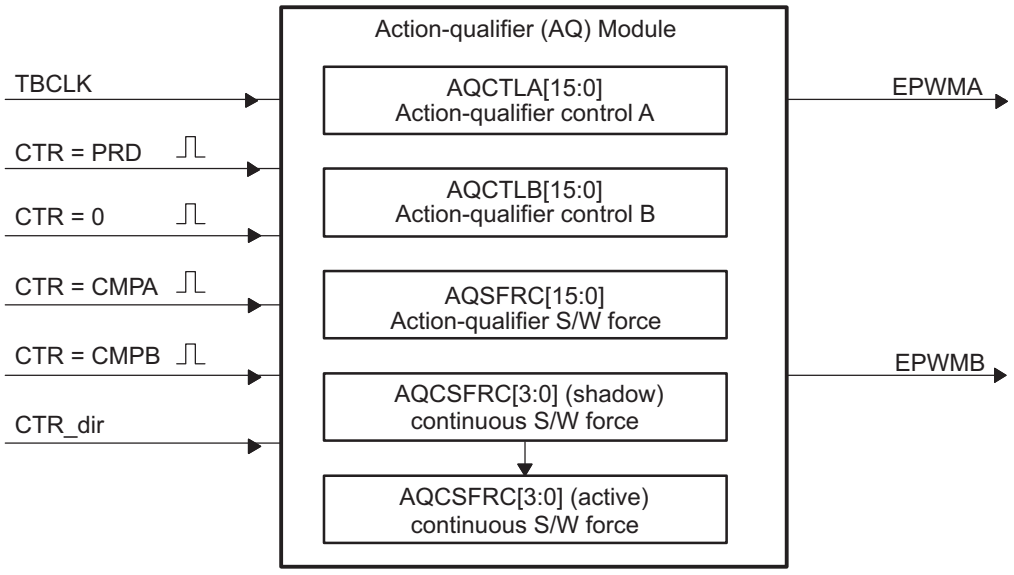


Table 16-8. Action-Qualifier Submodule Possible Input Events

Signal	Description	Registers Compared
CTR = PRD	Time-base counter equal to the period value	TBCNT = TBPRD
CTR = 0	Time-base counter equal to zero	TBCNT = 0000h
CTR = CMPA	Time-base counter equal to the counter-compare A	TBCNT = CMPA
CTR = CMPB	Time-base counter equal to the counter-compare B	TBCNT = 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:

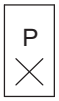
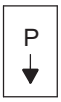
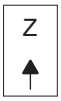
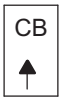
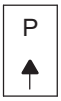
- **Set High:** Set output EPWMxA or EPWMxB to a high level.
- **Clear Low:** Set output EPWMxA or EPWMxB to a low level.
- **Toggle:** If EPWMxA or EPWMxB is currently pulled high, then pull the output low. If EPWMxA or EPWMxB is currently pulled low, then pull the output high.
- **Do Nothing:** Keep 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. See the event-trigger submodule description in Section 16.2.9 for details.

Copyright © 2013–2016, Texas Instruments Incorporated

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 chapter use a set of symbolic actions. These symbols are summarized in Figure 16-20. 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 16-20. Possible Action-Qualifier Actions for EPWMxA and EPWMxB Outputs

S/W force	TB Counter equals:				Actions
	Zero	Comp A	Comp B	Period	
					Do Nothing
					Clear Low
					Set High
					Toggle

Copyright © 2013–2016, Texas Instruments Incorporated

16.2.5.3 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 16-9. A priority level of 1 is the highest priority and level 7 is the lowest. The priority changes slightly depending on the direction of TBCNT.

Table 16-9. Action-Qualifier Event Priority for Up-Down-Count Mode

Priority Level	Event if TBCNT is Incrementing TBCNT = 0 up to TBCNT = TBPRD	Event if TBCNT is Decrementing TBCNT = TBPRD down to TBCNT = 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) (1)	Counter equals CMPB on up-count (CBU) (1)
6 (Lowest)	Counter equals CMPA on down-count (CAD) (1)	Counter equals CMPA on up-count (CBU) (1)

(1) To maintain symmetry for up-down-count mode, both up-events (CAU/CBU) and down-events (CAD/CBD) can be generated for TBPRD. Otherwise, up-events can occur only when the counter is incrementing and down-events can occur only when the counter is decrementing.

Table 16-10 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 16-10. 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 16-11 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 16-11. 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)

Copyright © 2013–2016, Texas Instruments Incorporated

It is possible to set the compare value greater than the period. In this case the action will take place as shown in Table 16-12.

Table 16-12. Behavior if CMPA/CMPB is Greater than the Period

Counter Mode	Compare on Up-Count Event CAU/CBU	Compare on Down-Count Event CAU/CBU
Up-Count Mode	If CMPA/CMPB ≤ TBPRD period, then the event occurs on a compare match (TBCNT = CMPA or CMPB). If CMPA/CMPB > TBPRD, then the event will not occur.	Never occurs.
Down-Count Mode	Never occurs.	If CMPA/CMPB < TBPRD, the event will occur on a compare match (TBCNT = CMPA or CMPB). If CMPA/CMPB ≥ TBPRD, the event will occur on a period match (TBCNT = TBPRD).
Up-Down-Count Mode	If CMPA/CMPB < TBPRD and the counter is incrementing, the event occurs on a compare match (TBCNT = CMPA or CMPB). If CMPA/CMPB is ≥ TBPRD, the event will occur on a period match (TBCNT = TBPRD).	If CMPA/CMPB < TBPRD and the counter is decrementing, the event occurs on a compare match (TBCNT = CMPA or CMPB). If CMPA/CMPB ≥ TBPRD, the event occurs on a period match (TBCNT = TBPRD).

16.2.5.4 Waveforms for Common Configurations

NOTE: The waveforms in this chapter 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:

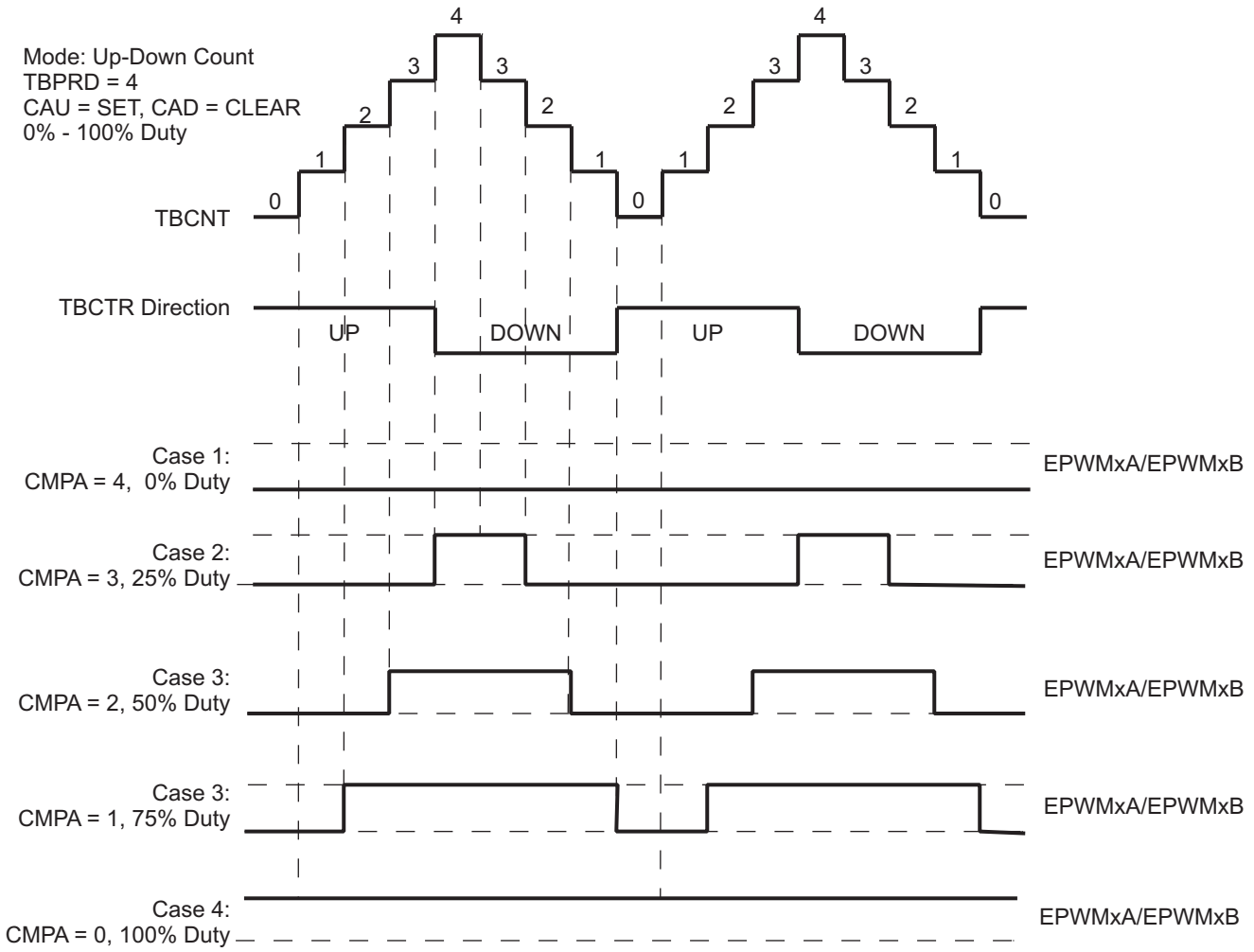
- To achieve 0-100% asymmetric PWM use the following configuration: Load CMPA/CMPB on TBPRD. Use the Zero action to set the PWM and a compare-up action to clear the PWM. Modulate the compare value from 0 to TBPRD+1 to achieve 0-100% PWM duty.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 16-21 shows how a symmetric PWM waveform can be generated using the up-down-count mode of the TBCNT. 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.

Figure 16-21. Up-Down-Count Mode Symmetrical Waveform



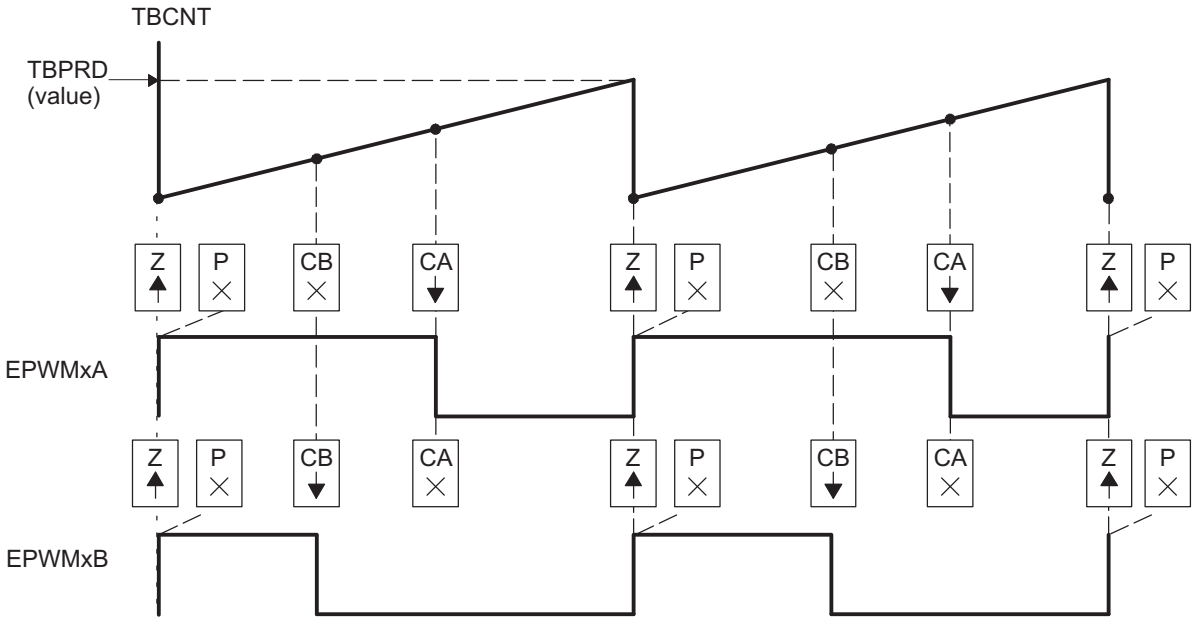
Copyright © 2013–2016, Texas Instruments Incorporated

The PWM waveforms in Figure 16-22 through Figure 16-27 show some common action-qualifier configurations. Some conventions used in the figures 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

Table 16-13 and Table 16-14 contains initialization and runtime register configurations for the waveforms in Figure 16-22.

Figure 16-22. Up, Single Edge Asymmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB—Active High



(1) PWM period = (TBPRD + 1) × T_{TBCLK}

(2) Duty modulation for EPWMxA is set by CMPA, and is active high (that is, high time duty proportional to CMPA).

(3) Duty modulation for EPWMxB is set by CMPB and is active high (that is, high time duty proportional to CMPB).

(4) The "Do Nothing" actions (X) are shown for completeness, but will not be shown on subsequent diagrams.

(5) Actions at zero and period, although appearing to occur concurrently, are actually separated by one TBCLK period. TBCNT wraps from period to 0000h.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 16-13. EPWMx Initialization for Figure 16-22

Register	Bit	Value	Comments
TBPRD	TBPRD	600 (258h)	Period = 601 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCNT	TBCNT	0	Clear TB counter
TBCTL	CTRMODE	TB_UP	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
	HSPCLKDIV	TB_DIV1	TBCLK = SYSCLK
	CLKDIV	TB_DIV1	
CMPA	CMPA	350 (15Eh)	Compare A = 350 TBCLK counts
CMPB	CMPB	200 (C8h)	Compare B = 200 TBCLK counts
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	ZRO	AQ_SET	
	CAU	AQ_CLEAR	
AQCTLB	ZRO	AQ_SET	
	CBU	AQ_CLEAR	

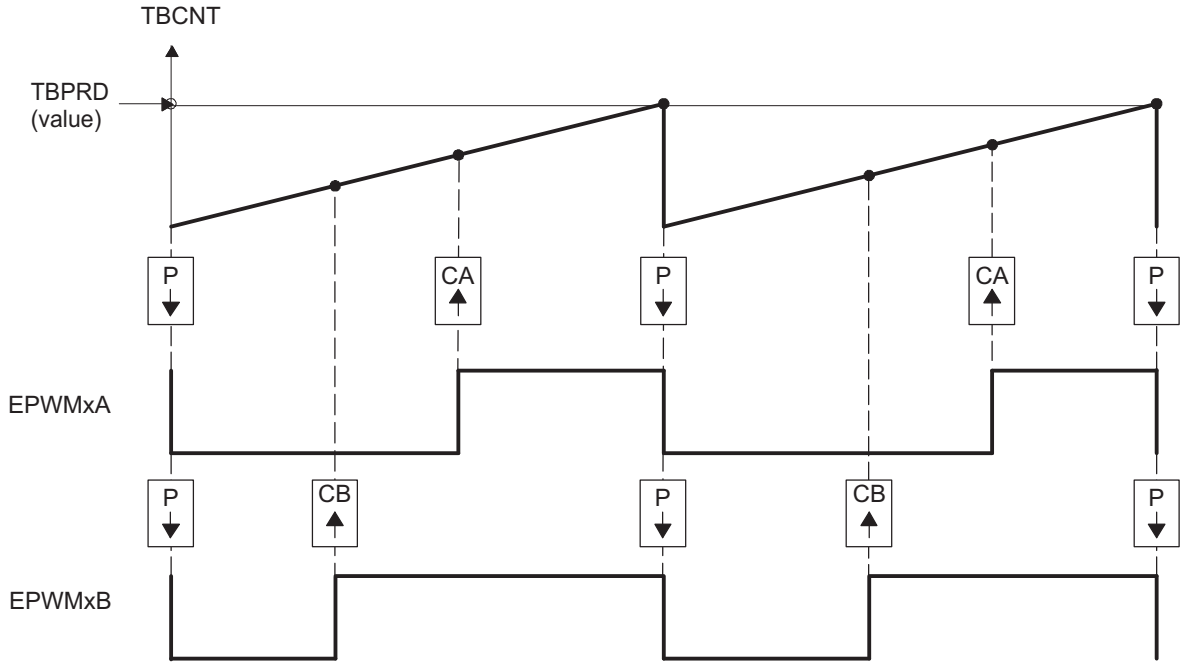
Table 16-14. EPWMx Run Time Changes for Figure 16-22

Register	Bit	Value	Comments
CMPA	CMPA	Duty1A	Adjust duty for output EPWM1A
CMPB	CMPB	Duty1B	Adjust duty for output EPWM1B

Copyright © 2013–2016, Texas Instruments Incorporated

Table 16-15 and Table 16-16 contains initialization and runtime register configurations for the waveforms in Figure 16-23.

Figure 16-23. Up, Single Edge Asymmetric Waveform With Independent Modulation on EPWMxA and EPWMxB—Active Low



(1) PWM period = (TBPRD + 1) × T_{TBCLK}

(2) Duty modulation for EPWMxA is set by CMPA, and is active low (that is, the low time duty is proportional to CMPA).

(3) Duty modulation for EPWMxB is set by CMPB and is active low (that is, the low time duty is proportional to CMPB).

(4) The Do Nothing actions (X) are shown for completeness here, but will not be shown on subsequent diagrams.

(5) Actions at zero and period, although appearing to occur concurrently, are actually separated by one TBCLK period. TBCNT wraps from period to 0000h.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 16-15. EPWMx Initialization for Figure 16-23

Register	Bit	Value	Comments
TBPRD	TBPRD	600 (258h)	Period = 601 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCNT	TBCNT	0	Clear TB counter
TBCTL	CTRMODE	TB_UP	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
	HSPCLKDIV	TB_DIV1	TBCLK = SYSCLK
	CLKDIV	TB_DIV1	
CMPA	CMPA	350 (15Eh)	Compare A = 350 TBCLK counts
CMPB	CMPB	200 (C8h)	Compare B = 200 TBCLK counts
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	PRD	AQ_CLEAR	
	CAU	AQ_SET	
AQCTLB	PRD	AQ_CLEAR	
	CBU	AQ_SET	

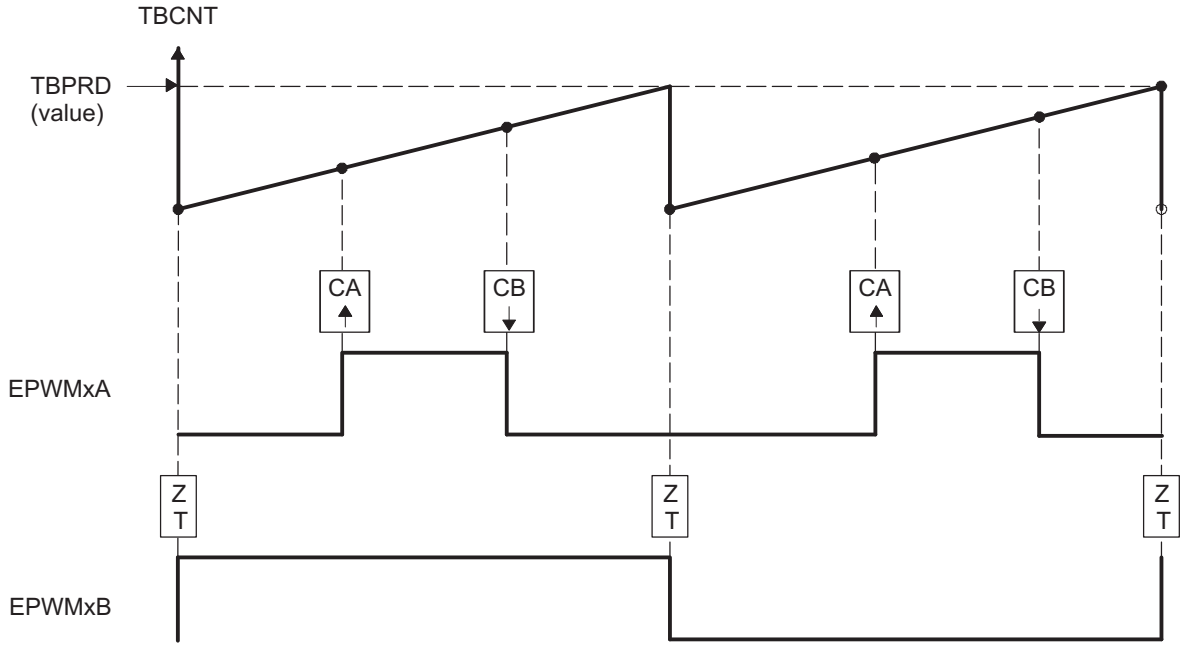
Table 16-16. EPWMx Run Time Changes for Figure 16-23

Register	Bit	Value	Comments
CMPA	CMPA	Duty1A	Adjust duty for output EPWM1A
CMPB	CMPB	Duty1B	Adjust duty for output EPWM1B

Copyright © 2013–2016, Texas Instruments Incorporated

Table 16-17 and Table 16-18 contains initialization and runtime register configurations for the waveforms Figure 16-24. Use the code in Example 16-1 to define the headers.

Figure 16-24. Up-Count, Pulse Placement Asymmetric Waveform With Independent Modulation on EPWMxA



(1) PWM frequency = 1/((TBPRD + 1) × T_{TBCLK})

(2) Pulse can be placed anywhere within the PWM cycle (0000h - TBPRD)

(3) High time duty proportional to (CMPB - CMPA)

(4) EPWMxB can be used to generate a 50% duty square wave with frequency = 1/2 × ((TBPRD + 1) × TBCLK)

Copyright © 2013–2016, Texas Instruments Incorporated

Table 16-17. EPWMx Initialization for Figure 16-24

Register	Bit	Value	Comments
TBPRD	TBPRD	600 (258h)	Period = 601 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCNT	TBCNT	0	Clear TB counter
TBCTL	CTRMODE	TB_UP	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
	HSPCLKDIV	TB_DIV1	TBCLK = SYSCLK
	CLKDIV	TB_DIV1	
CMPA	CMPA	200 (C8h)	Compare A = 200 TBCLK counts
CMPB	CMPB	400 (190h)	Compare B = 400 TBCLK counts
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	CAU	AQ_SET	
	CBU	AQ_CLEAR	
AQCTLB	ZRO	AQ_TOGGLE	

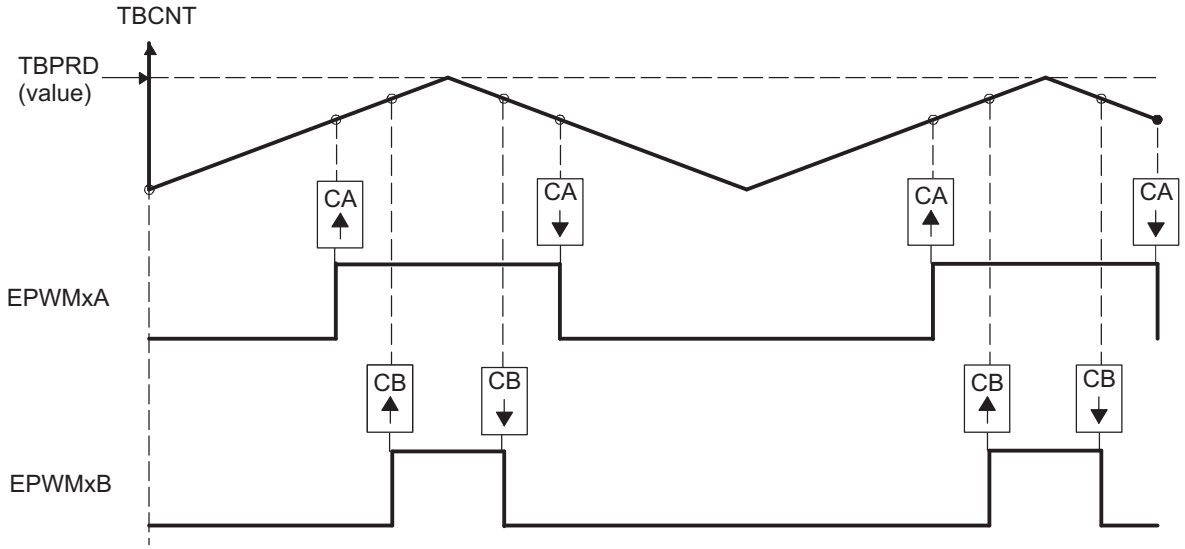
Table 16-18. EPWMx Run Time Changes for Figure 16-24

Register	Bit	Value	Comments
CMPA	CMPA	EdgePosA	Adjust duty for output EPWM1A
CMPB	CMPB	EdgePosB	

Copyright © 2013–2016, Texas Instruments Incorporated

Table 16-19 and Table 16-20 contains initialization and runtime register configurations for the waveforms in Figure 16-25. Use the code in Example 16-1 to define the headers.

Figure 16-25. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Active Low



(1) PWM period = 2 x TBPRD × T_{TBCLK}

(2) Duty modulation for EPWMxA is set by CMPA, and is active low (that is, the low time duty is proportional to CMPA).

(3) Duty modulation for EPWMxB is set by CMPB and is active low (that is, the low time duty is proportional to CMPB).

(4) Outputs EPWMxA and EPWMxB can drive independent power switches

Copyright © 2013–2016, Texas Instruments Incorporated

Table 16-19. EPWMx Initialization for Figure 16-25

Register	Bit	Value	Comments
TBPRD	TBPRD	600 (258h)	Period = 601 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCNT	TBCNT	0	Clear TB counter
TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
	HSPCLKDIV	TB_DIV1	TBCLK = SYSCLK
	CLKDIV	TB_DIV1	
CMPA	CMPA	400 (190h)	Compare A = 400 TBCLK counts
CMPB	CMPB	500 (1F4h)	Compare B = 500 TBCLK counts
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	CAU	AQ_SET	
	CAD	AQ_CLEAR	
AQCTLB	CBU	AQ_SET	
	CBD	AQ_CLEAR	

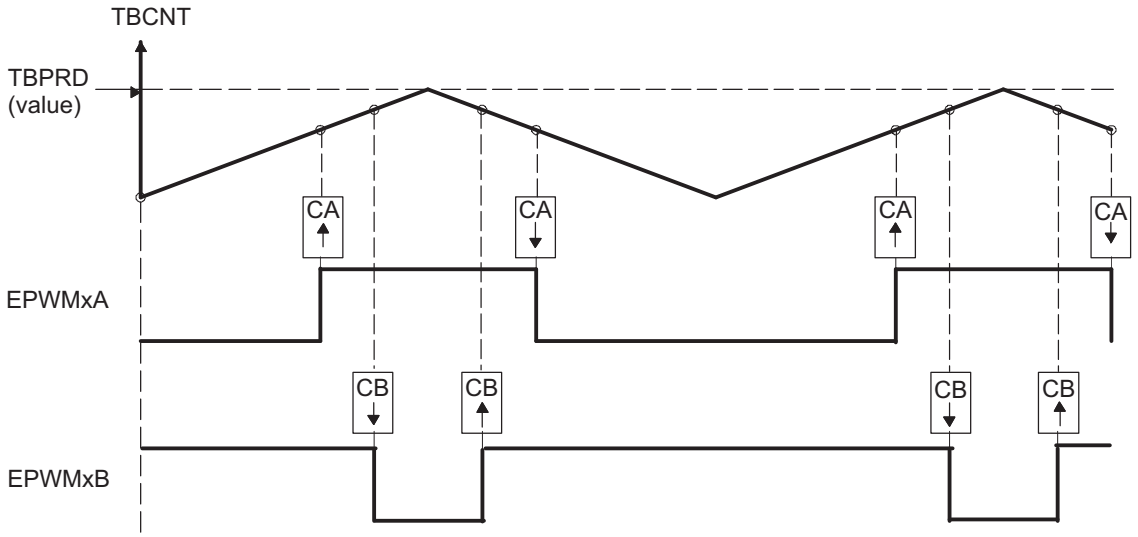
Table 16-20. EPWMx Run Time Changes for Figure 16-25

Register	Bit	Value	Comments
CMPA	CMPA	Duty1A	Adjust duty for output EPWM1A
CMPB	CMPB	Duty1B	Adjust duty for output EPWM1B

Copyright © 2013–2016, Texas Instruments Incorporated

Table 16-21 and Table 16-22 contains initialization and runtime register configurations for the waveforms in Figure 16-26. Use the code in Example 16-1 to define the headers.

Figure 16-26. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Complementary



(1) PWM period = 2 × TBPRD × T_{TBCLK}

(2) Duty modulation for EPWMxA is set by CMPA, and is active low, i.e., low time duty proportional to CMPA

(3) Duty modulation for EPWMxB is set by CMPB and is active high, i.e., high time duty proportional to CMPB

(4) Outputs EPWMx can drive upper/lower (complementary) power switches

(5) 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.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 16-21. EPWMx Initialization for Figure 16-26

Register	Bit	Value	Comments
TBPRD	TBPRD	600 (258h)	Period = 601 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCNT	TBCNT	0	Clear TB counter
TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
	HSPCLKDIV	TB_DIV1	TBCLK = SYSCLK
	CLKDIV	TB_DIV1	
CMPA	CMPA	350 (15Eh)	Compare A = 350 TBCLK counts
CMPB	CMPB	400 (190h)	Compare B = 400 TBCLK counts
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	CAU	AQ_SET	
	CAD	AQ_CLEAR	
AQCTLB	CBU	AQ_CLEAR	
	CBD	AQ_SET	

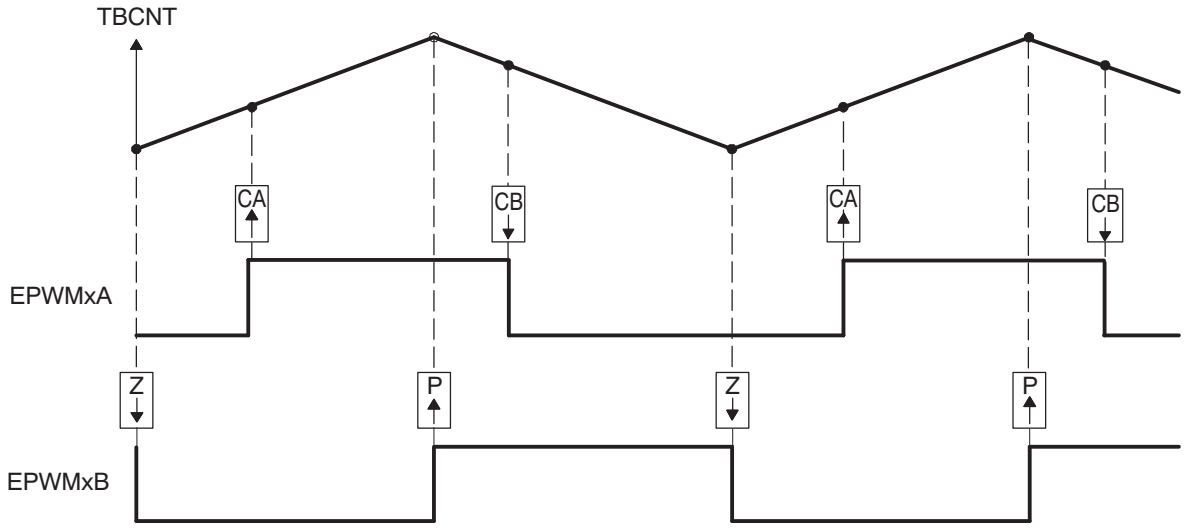
Table 16-22. EPWMx Run Time Changes for Figure 16-26

Register	Bit	Value	Comments
CMPA	CMPA	Duty1A	Adjust duty for output EPWM1A
CMPB	CMPB	Duty1B	Adjust duty for output EPWM1B

Copyright © 2013–2016, Texas Instruments Incorporated

Table 16-23 and Table 16-24 contains initialization and runtime register configurations for the waveforms in Figure 16-27. Use the code in Example 16-1 to define the headers.

Figure 16-27. Up-Down-Count, Dual Edge Asymmetric Waveform, With Independent Modulation on EPWMxA—Active Low



(1) PWM period = 2 × TBPRD × TBCLK

(2) Rising edge and falling edge can be asymmetrically positioned within a PWM cycle. This allows for pulse placement techniques.

(3) Duty modulation for EPWMxA is set by CMPA and CMPB.

(4) Low time duty for EPWMxA is proportional to (CMPA + CMPB).

(5) To change this example to active high, CMPA and CMPB actions need to be inverted (i.e., Set ! Clear and Clear Set).

(6) Duty modulation for EPWMxB is fixed at 50% (utilizes spare action resources for EPWMxB)

Copyright © 2013–2016, Texas Instruments Incorporated

Table 16-23. EPWMx Initialization for Figure 16-27

Register	Bit	Value	Comments
TBPRD	TBPRD	600 (258h)	Period = 601 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCNT	TBCNT	0	Clear TB counter
TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
	HSPCLKDIV	TB_DIV1	TBCLK = SYSCLK
	CLKDIV	TB_DIV1	
CMPA	CMPA	250 (FAh)	Compare A = 250 TBCLK counts
CMPB	CMPB	450 (1C2h)	Compare B = 450 TBCLK counts
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	CAU	AQ_SET	
	CBD	AQ_CLEAR	
AQCTLB	ZRO	AQ_CLEAR	
	PRD	AQ_SET	

Table 16-24. EPWMx Run Time Changes for Figure 16-27

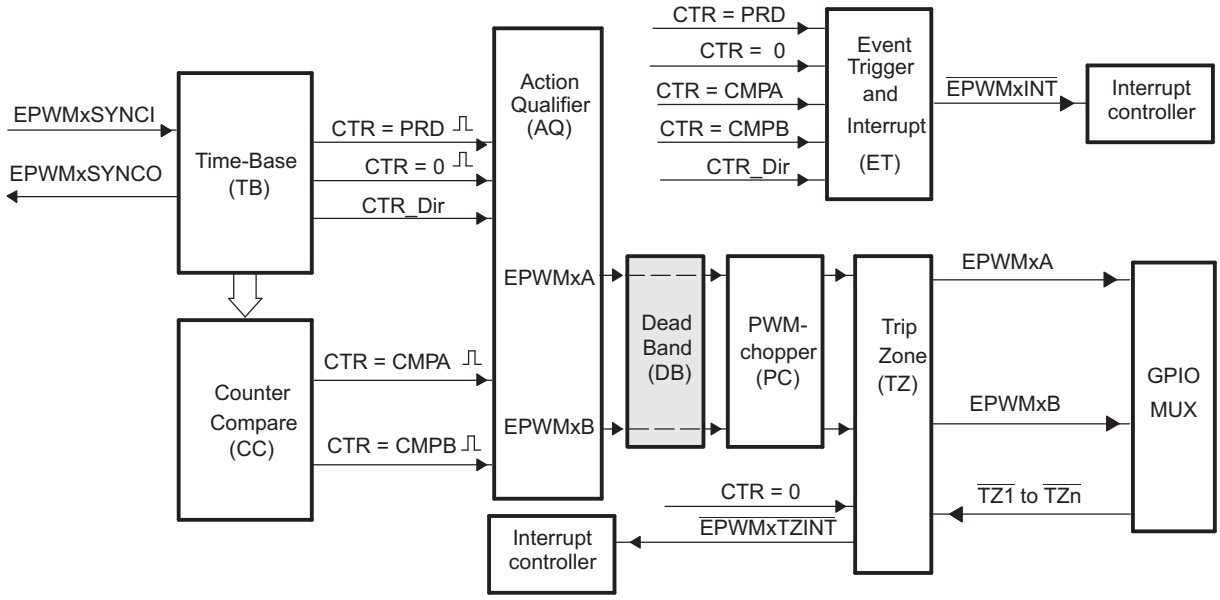
Register	Bit	Value	Comments
CMPA	CMPA	EdgePosA	Adjust duty for output EPWM1A
CMPB	CMPB	EdgePosB	

Copyright © 2013–2016, Texas Instruments Incorporated

16.2.6 Dead-Band Generator (DB) Submodule

Figure 16-28 illustrates the dead-band generator submodule within the ePWM module.

Figure 16-28. Dead-Band Generator Submodule



16.2.6.1 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 generator submodule should be used.

The key functions of the dead-band generator submodule 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)

16.2.6.2 Controlling and Monitoring the Dead-Band Submodule

The dead-band generator submodule operation is controlled and monitored via the following registers:

Table 16-25. Dead-Band Generator Submodule Registers

Acronym	Register Description	Address Offset	Shadowed
DBCTL	Dead-Band Control Register	1Eh	No
DBRED	Dead-Band Rising Edge Delay Count Register	20h	No
DBFED	Dead-Band Falling Edge Delay Count Register	22h	No

Copyright © 2013–2016, Texas Instruments Incorporated

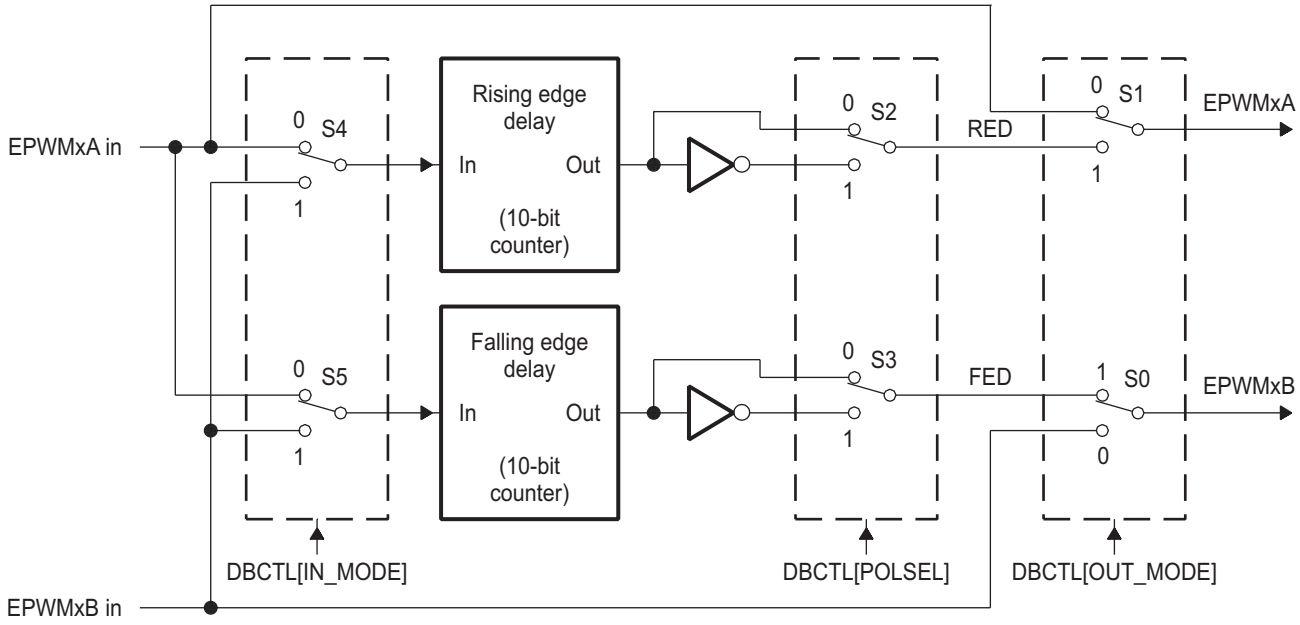
16.2.6.3 Operational Highlights for the Dead-Band Generator Submodule

The following sections provide the operational highlights.

The dead-band submodule has two groups of independent selection options as shown in Figure 16-29.

- **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.
- **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.

Figure 16-29. Configuration Options for the Dead-Band Generator Submodule



Copyright © 2013–2016, Texas Instruments Incorporated

Although all combinations are supported, not all are typical usage modes. Table 16-26 lists 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 16-26 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 16-30. Note that to generate equivalent waveforms to Figure 16-30, 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 16-26 show combinations where either the falling-edge-delay (FED) or rising-edge-delay (RED) blocks are bypassed.

Table 16-26. Classical Dead-Band Operating Modes

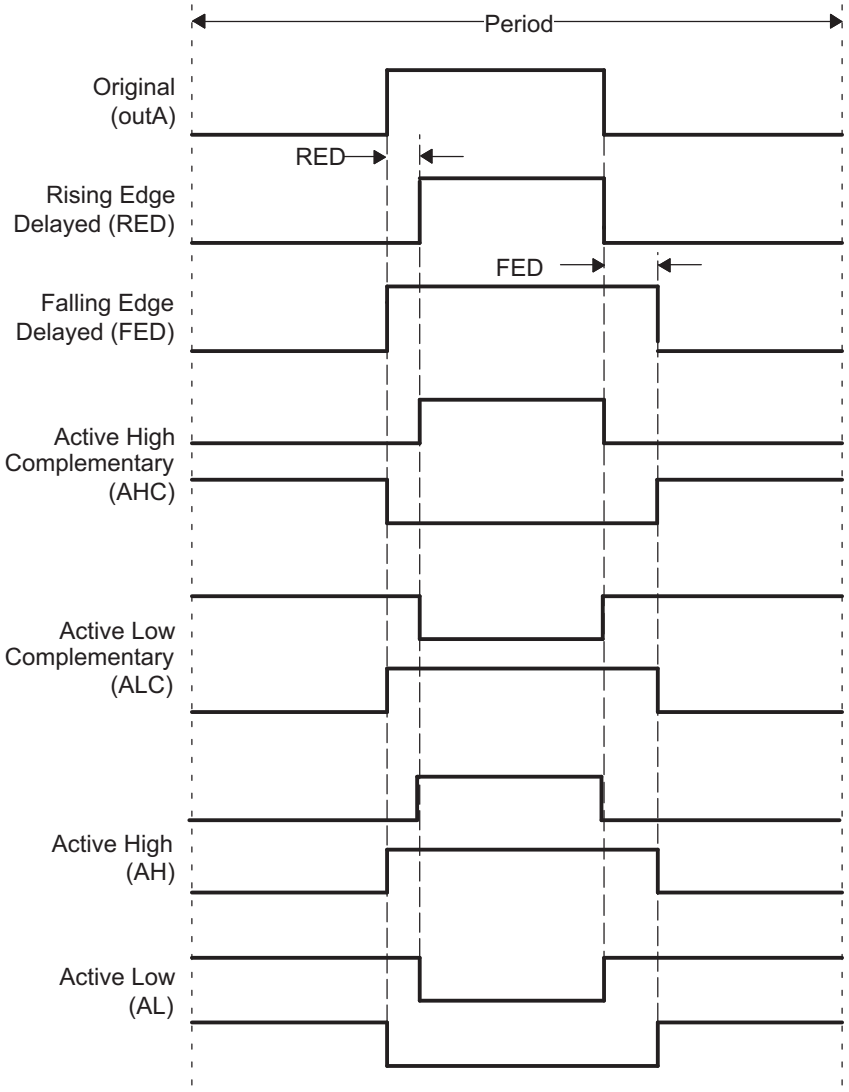
Mode	Mode Description [1]	DBCTL[POLSEL]		DBCTL[OUT_MODE]	
		S3	S2	S1	S0
1	EPWMxA and EPWMxB Passed Through (No Delay)	x	x	0	0
2	Active High Complementary (AHC)	1	0	1	1
3	Active Low Complementary (ALC)	0	1	1	1
4	Active High (AH)	0	0	1	1
5	Active Low (AL)	1	1	1	1
6	EPWMxA Out = EPWMxA In (No Delay) EPWMxB 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

(1) These are classical dead-band modes and assume that DBCTL[IN_MODE] = 0,0. That is, EPWMxA in is the source for both the falling-edge and rising-edge delays. Enhanced, non-traditional modes can be achieved by changing the IN_MODE configuration.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 16-30 shows waveforms for typical cases where 0% < duty < 100%.

Figure 16-30. Dead-Band Waveforms for Typical Cases (0% < Duty < 100%)



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:

$FED = DBFED \times T_{TBCLK}$

$RED = DBRED \times T_{TBCLK}$

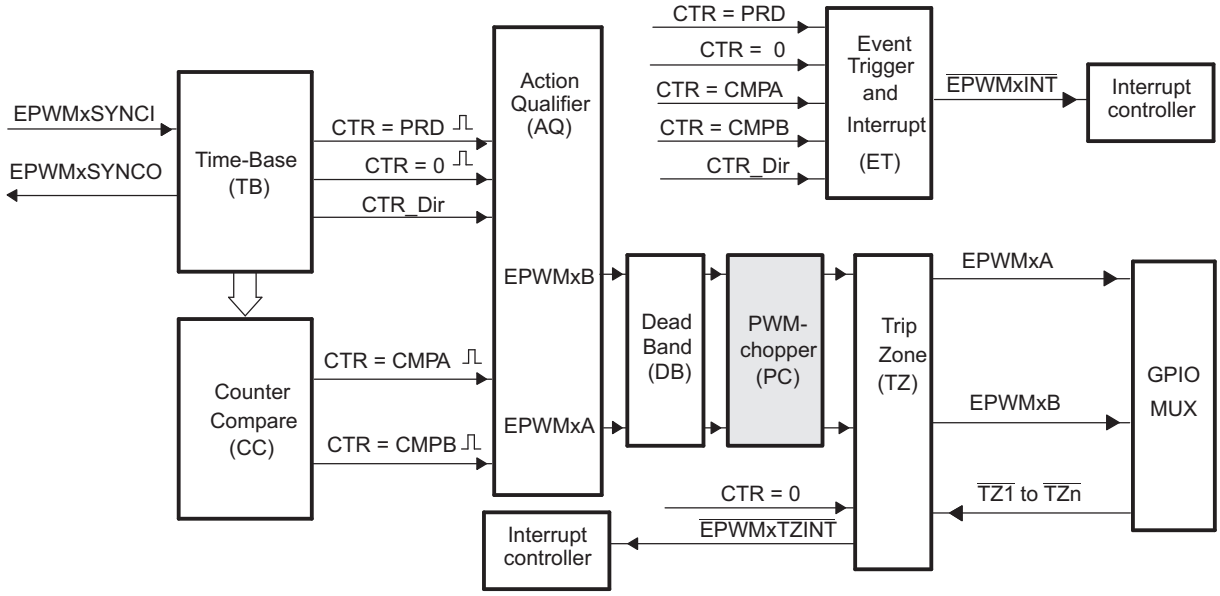
Where T_{TBCLK} is the period of TBCLK, the prescaled version of SYSCLKOUT.

Copyright © 2013–2016, Texas Instruments Incorporated

16.2.7 PWM-Chopper (PC) Submodule

Figure 16-31 illustrates the PWM-chopper (PC) submodule within the ePWM module. 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.

Figure 16-31. PWM-Chopper Submodule



16.2.7.1 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

16.2.7.2 Controlling the PWM-Chopper Submodule

The PWM-chopper submodule operation is controlled via the register in Table 16-27.

Table 16-27. PWM-Chopper Submodule Registers

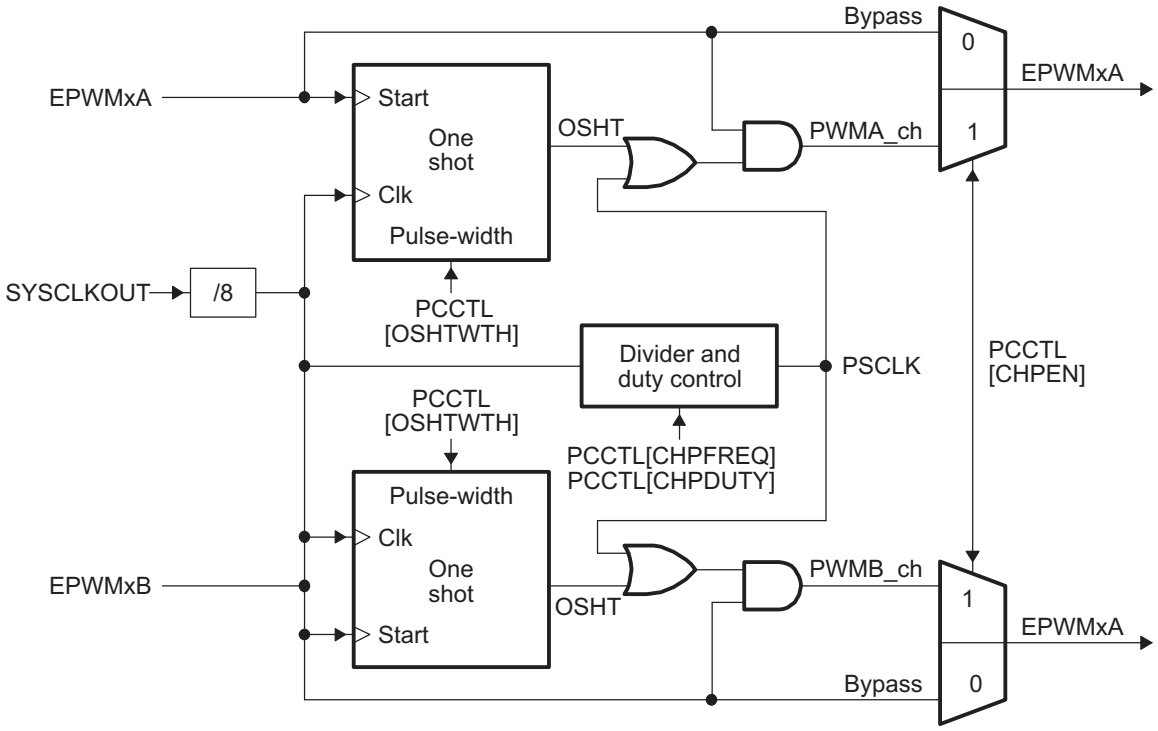
Acronym	Register Description	Address Offset	Shadowed
PCCTL	PWM-chopper Control Register	3Ch	No

Copyright © 2013–2016, Texas Instruments Incorporated

16.2.7.3 Operational Highlights for the PWM-Chopper Submodule

Figure 16-32 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.

Figure 16-32. PWM-Chopper Submodule Signals and Registers

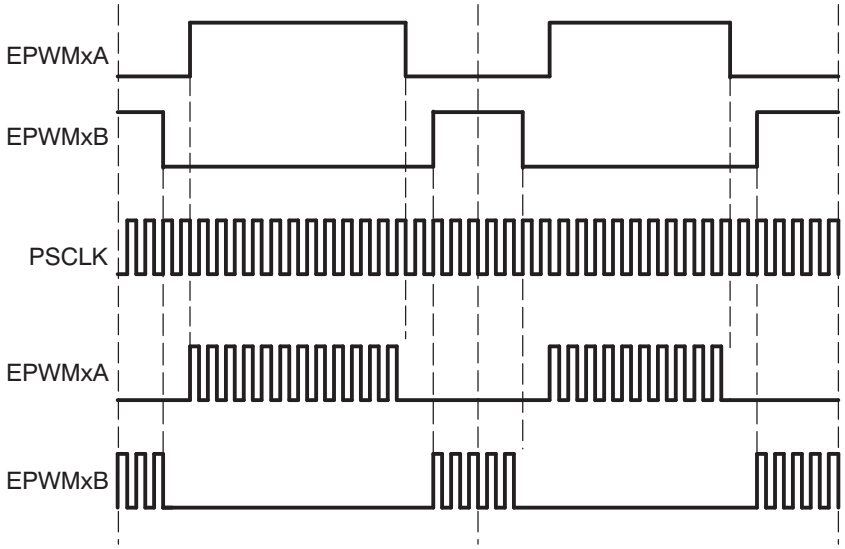


Copyright © 2013–2016, Texas Instruments Incorporated

16.2.7.4 Waveforms

Figure 16-33 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 16-33. Simple PWM-Chopper Submodule Waveforms Showing Chopping Action Only



16.2.7.4.1 One-Shot Pulse

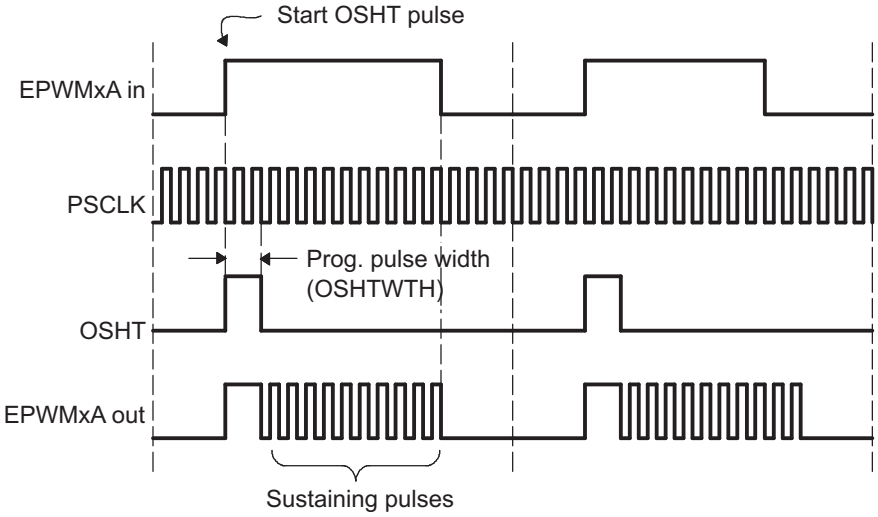
The width of the first pulse can be programmed to any of 16 possible pulse width values. The width or period of the first pulse is given by:

$T_{1stpulse} = T_{SYSCLKOUT} \times 8 \times OSHTWTH$

Where $T_{SYSCLKOUT}$ is the period of the system clock (SYSCLKOUT) and OSHTWTH is the four control bits (value from 1 to 16)

Figure 16-34 shows the first and subsequent sustaining pulses.

Figure 16-34. PWM-Chopper Submodule Waveforms Showing the First Pulse and Subsequent Sustaining Pulses



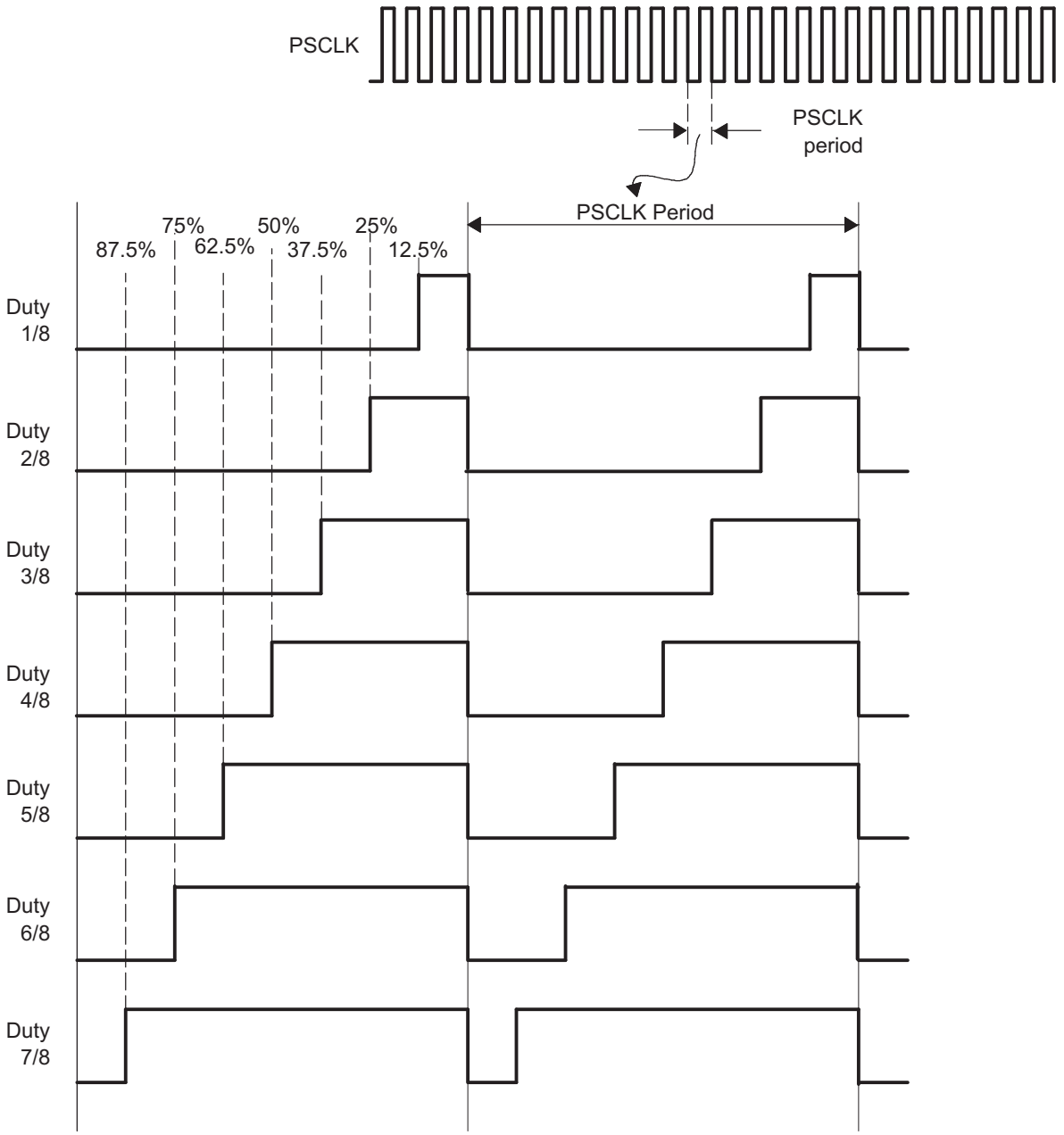
Copyright © 2013–2016, Texas Instruments Incorporated

16.2.7.4.2 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 16-35 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 16-35. PWM-Chopper Submodule Waveforms Showing the Pulse Width (Duty Cycle) Control of Sustaining Pulses

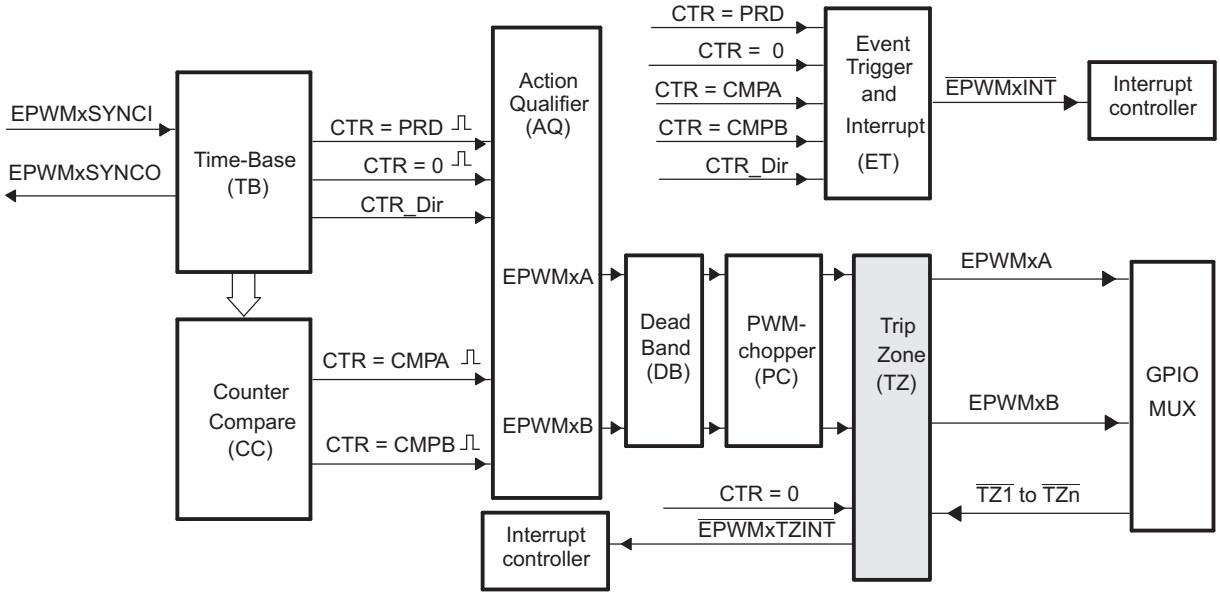


Copyright © 2013–2016, Texas Instruments Incorporated

16.2.8 Trip-Zone (TZ) Submodule

Figure 16-36 shows how the trip-zone (TZ) submodule fits within the ePWM module. Each ePWM module is connected to every $\overline{TZ}$ signal that are sourced from the GPIO MUX. These signals indicates external fault or trip conditions, and the ePWM outputs can be programmed to respond accordingly when faults occur. See your device-specific data manual to determine the number of trip-zone pins available for the device.

Figure 16-36. Trip-Zone Submodule



16.2.8.1 Purpose of the Trip-Zone Submodule

The key functions of the trip-zone submodule are:

- Trip inputs $\overline{TZ1}$ to $\overline{TZn}$ 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.
- Each trip-zone input pin can be allocated to either one-shot or cycle-by-cycle operation.
- Interrupt generation is possible on any trip-zone pin.
- Software-forced tripping is also supported.
- The trip-zone submodule can be fully bypassed if it is not required.

Copyright © 2013–2016, Texas Instruments Incorporated

16.2.8.2 Controlling and Monitoring the Trip-Zone Submodule

The trip-zone submodule operation is controlled and monitored through the following registers:

Table 16-28. Trip-Zone Submodule Registers

Acronym	Register Description	Address Offset	Shadowed
TZSEL	Trip-Zone Select Register	24h	No
TZCTL	Trip-Zone Control Register	28h	No
TZEINT	Trip-Zone Enable Interrupt Register	2Ah	No
TZFLG	Trip-Zone Flag Register	2Ch	No
TZCLR	Trip-Zone Clear Register	2Eh	No
TZFRC	Trip-Zone Force Register	30h	No

16.2.8.3 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 at pin $\overline{TZ1}$ to $\overline{TZn}$ is an active-low input signal. When the pin goes low, it indicates that a trip event has occurred. Each ePWM module can be individually configured to ignore or use each of the trip-zone pins. Which trip-zone pins are used by a particular ePWM module is determined by the TZSEL register for that specific ePWM module. The trip-zone signal may or may not be synchronized to the system clock (SYSCLKOUT). A minimum of 1 SYSCLKOUT low pulse on the $\overline{TZ}$ $\overline{n}$ inputs is sufficient to trigger a fault condition in the ePWM module. 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 the $\overline{TZn}$ inputs.

The $\overline{TZ}$ $\overline{n}$ input can be individually configured to provide either a cycle-by-cycle or one-shot trip event for a ePWM module. The configuration is determined by the TZSEL[CBCn] and TZSEL[OSHTn] bits (where n corresponds to the trip pin) respectively.

- **Cycle-by-Cycle (CBC):** When a cycle-by-cycle trip event occurs, the action specified in the TZCTL register is carried out immediately on the EPWMxA and/or EPWMxB output. Table 16-29 lists the possible actions. In addition, the cycle-by-cycle trip event flag (TZFLG[CBC]) is set and a EPWMxTZINT interrupt is generated if it is enabled in the TZEINT register.

  The specified condition on the pins is automatically cleared when the ePWM time-base counter reaches zero (TBCNT = 0000h) 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 register is carried out immediately on the EPWMxA and/or EPWMxB output. Table 16-29 lists the possible actions. In addition, the one-shot trip event flag (TZFLG[OST]) is set and a EPWMxTZINT interrupt is generated if it is enabled in the TZEINT register. The one-shot trip condition must be cleared manually by writing to the TZCLR[OST] bit.

The action taken when a trip event occurs can be configured individually for each of the ePWM output pins by way of the TZCTL[TZA] and TZCTL[TZB] register bits. One of four possible actions, shown in Table 16-29, can be taken on a trip event.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 16-29. Possible Actions On a Trip Event

TZCTL[TZA] and/or TZCTL[TZB]	EPWMxA and/or EPWMxB	Comment
0	High-Impedance	Tripped
1h	Force to High State	Tripped
2h	Force to Low State	Tripped
3h	No Change	Do Nothing. No change is made to the output.

Example 16-2. 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 $\overline{TZ}$ 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 $\overline{TZ}$ 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 ePWM1
  - TZCTL[TZA] = 0: EPWM1A will be put into a high-impedance state on a trip event.
  - TZCTL[TZB] = 3: EPWM1B will ignore the trip event.

16.2.8.4 Generating Trip Event Interrupts

Figure 16-37 and Figure 16-38 illustrate the trip-zone submodule control and interrupt logic, respectively.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 16-37. Trip-Zone Submodule Mode Control Logic

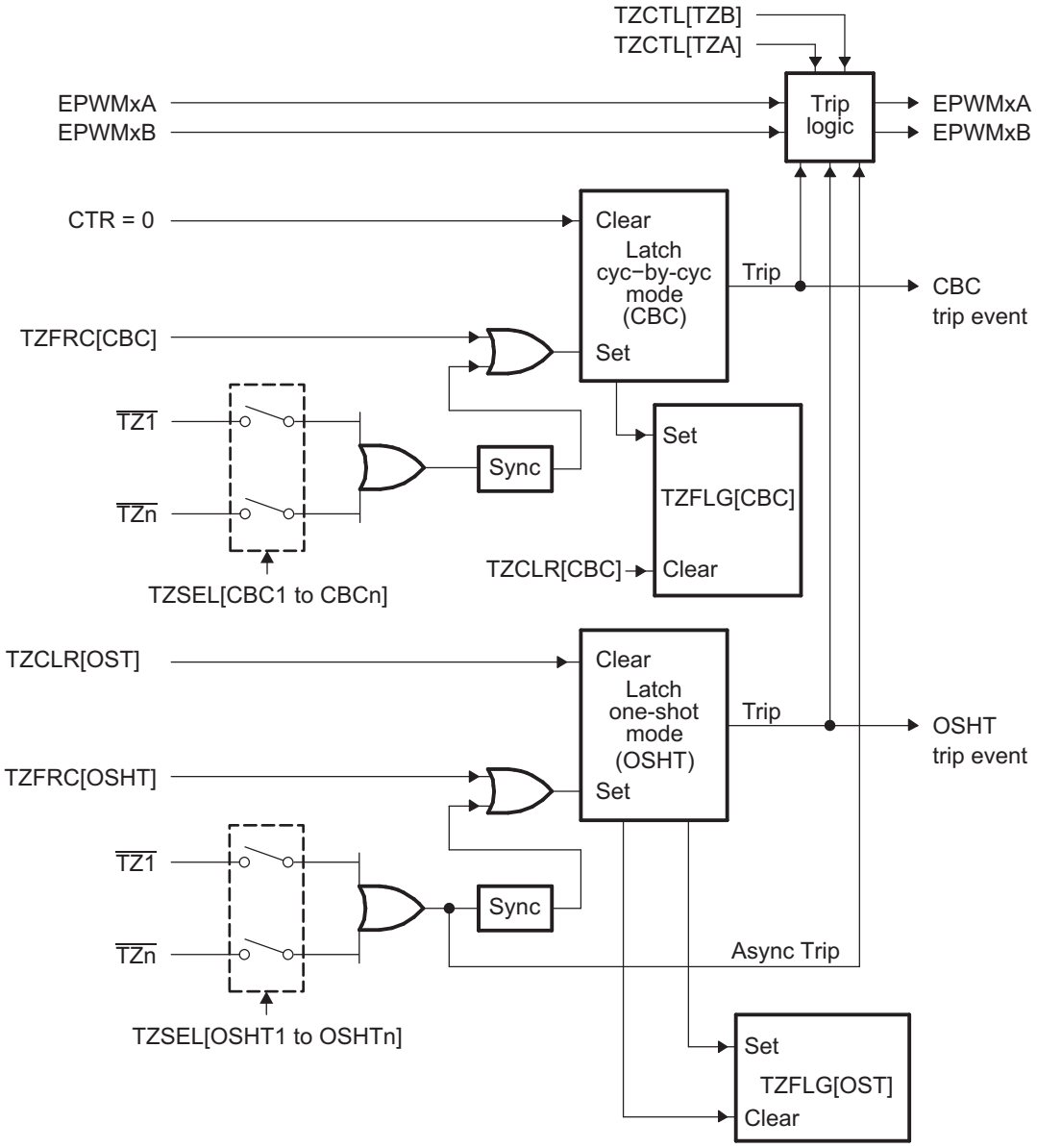
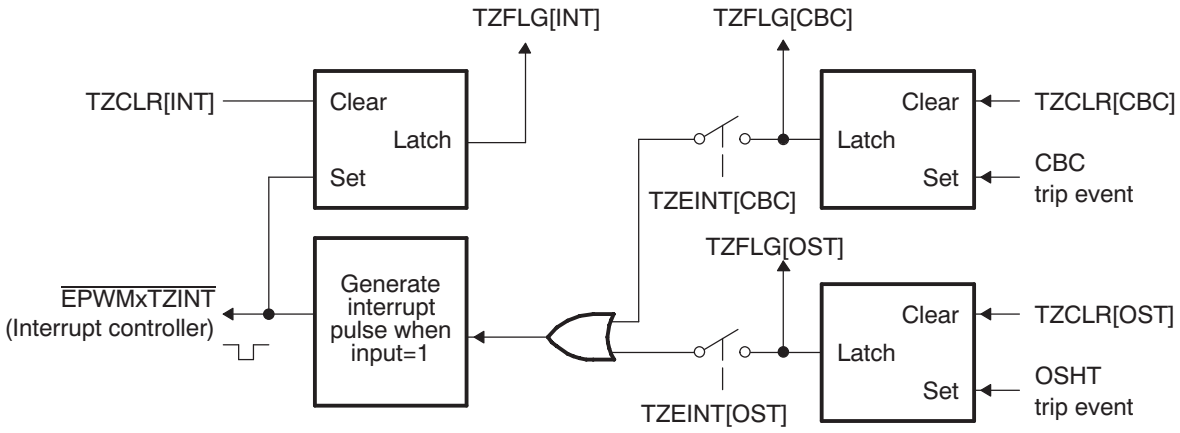


Figure 16-38. Trip-Zone Submodule Interrupt Logic

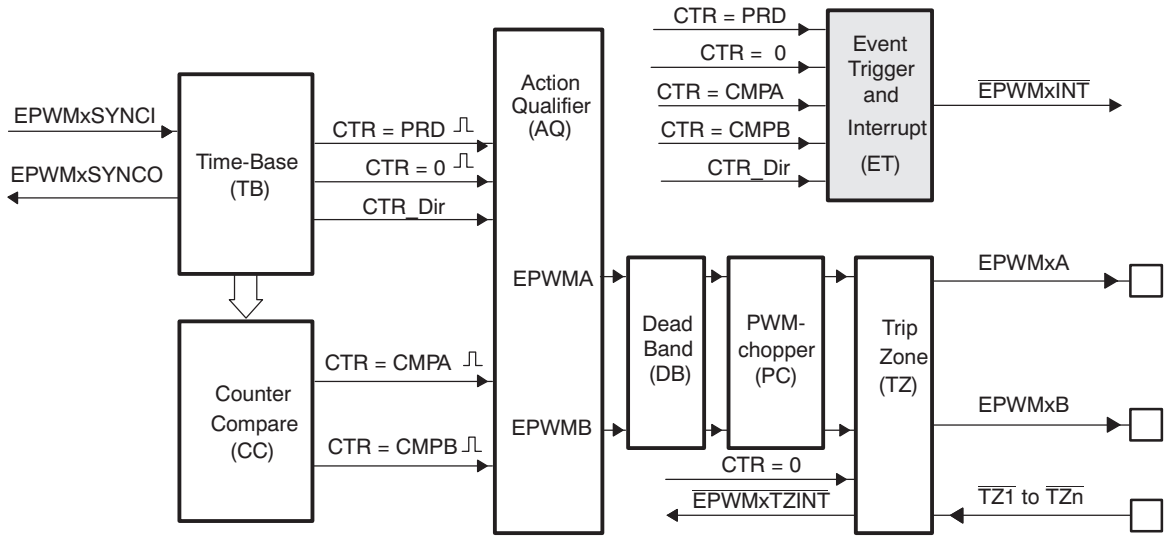


Copyright © 2013–2016, Texas Instruments Incorporated

16.2.9 Event-Trigger (ET) Submodule

Figure 16-39 shows the event-trigger (ET) submodule in the ePWM system. The event-trigger submodule manages the events generated by the time-base submodule and the counter-compare submodule to generate an interrupt to the CPU.

Figure 16-39. Event-Trigger Submodule



16.2.9.1 Purpose of the Event-Trigger Submodule

The key functions of the event-trigger submodule are:

- Receives event inputs generated by the time-base and counter-compare submodules
- Uses the time-base direction information for up/down event qualification
- Uses prescaling logic to issue interrupt requests at:
  - Every event
  - Every second event
  - Every third event
- Provides full visibility of event generation via event counters and flags

16.2.9.2 Controlling and Monitoring the Event-Trigger Submodule

The key registers used to configure the event-trigger submodule are shown in Table 16-30:

Table 16-30. Event-Trigger Submodule Registers

Acronym	Register Description	Address Offset	Shadowed
ETSEL	Event-Trigger Selection Register	32h	No
ETPS	Event-Trigger Prescale Register	34h	No
ETFLG	Event-Trigger Flag Register	36h	No
ETCLR	Event-Trigger Clear Register	38h	No
ETFRC	Event-Trigger Force Register	3Ah	No

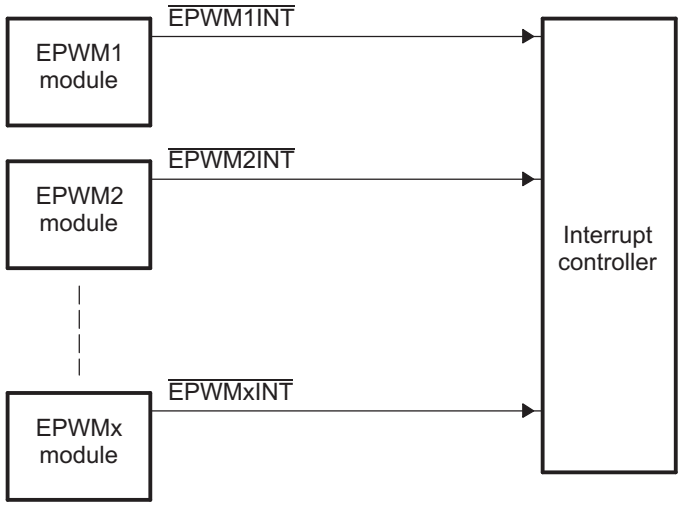
Copyright © 2013–2016, Texas Instruments Incorporated

16.2.9.3 Operational Overview of the 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 interrupt controller as shown in Figure 16-40.

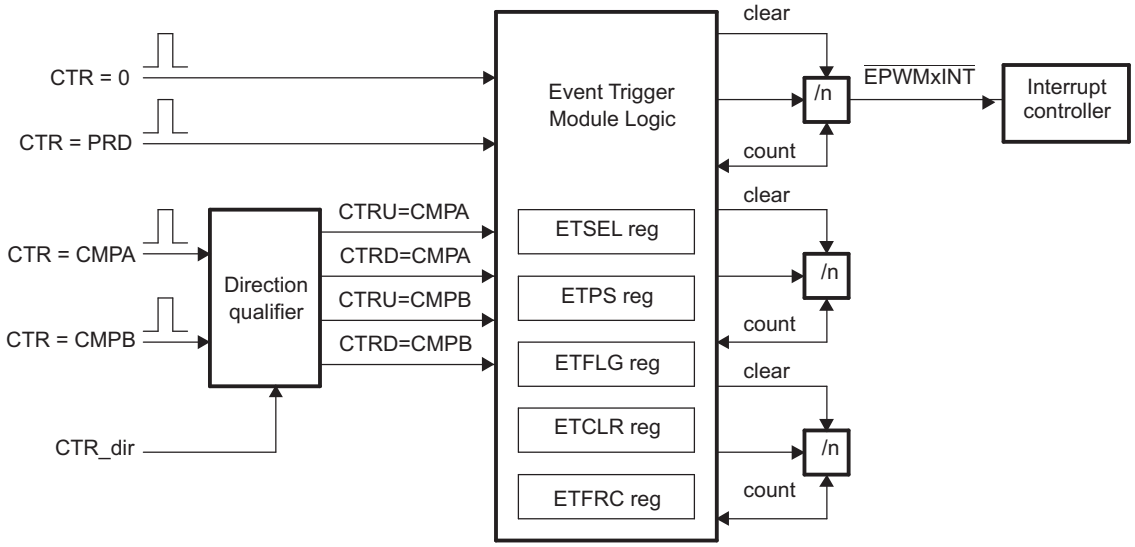
Figure 16-40. Event-Trigger Submodule Inter-Connectivity to Interrupt Controller



The event-trigger submodule monitors various event conditions (the left side inputs to event-trigger submodule shown in Figure 16-41) and can be configured to prescale these events before issuing an Interrupt request. The event-trigger prescaling logic can issue Interrupt requests at:

- Every event
- Every second event
- Every third event

Figure 16-41. Event-Trigger Submodule Showing Event Inputs and Prescaled Outputs



Copyright © 2013–2016, Texas Instruments Incorporated

- ETSEL—This selects which of the possible events will trigger an interrupt.
- ETPS—This programs the event prescaling options previously mentioned.
- 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.

A more detailed look at how the various register bits interact with the Interrupt is shown in Figure 16-42.

Figure 16-42 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

An interrupt cannot be generated on every fourth or more events.

Which event can cause an interrupt is configured by the interrupt selection (ETSEL[INTSEL]) bits. The event can be one of the following:

- Time-base counter equal to zero (TBCNT = 0000h).
- Time-base counter equal to period (TBCNT = 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.

The number of events that have occurred can be read from the interrupt event counter (ETPS[INTCNT]) register bits. That is, when the specified event occurs the ETPS[INTCNT] bits are incremented until they reach the value specified by ETPS[INTPRD]. 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 interrupt controller.

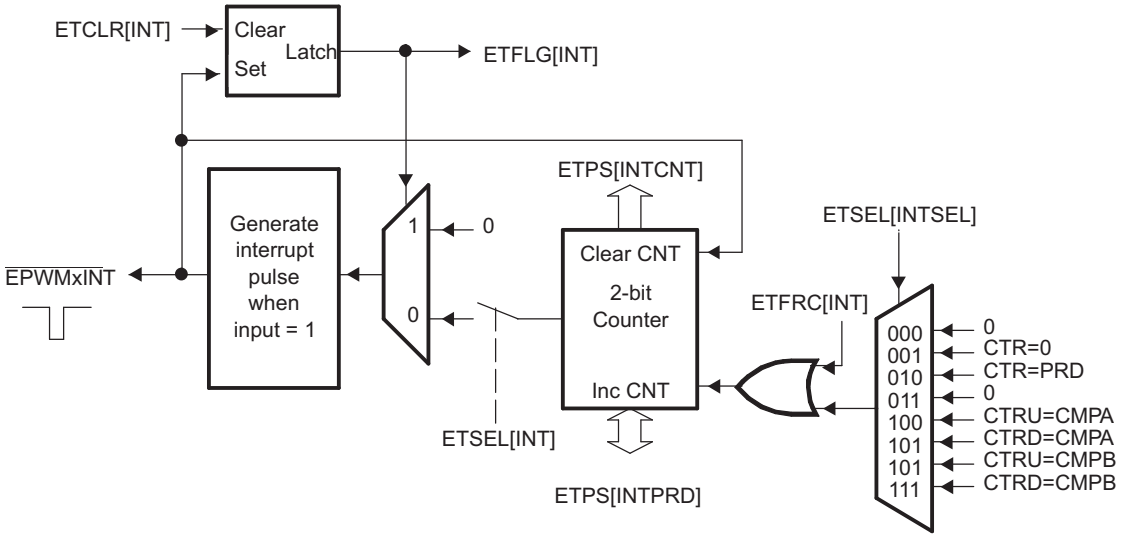
When ETPS[INTCNT] reaches ETPS[INTPRD], one of the following behaviors will occur:

- 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 will automatically clear the counter INTCNT = 0 and the counter output will be reset (so no interrupts are generated). 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.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 16-42. Event-Trigger Interrupt Generator

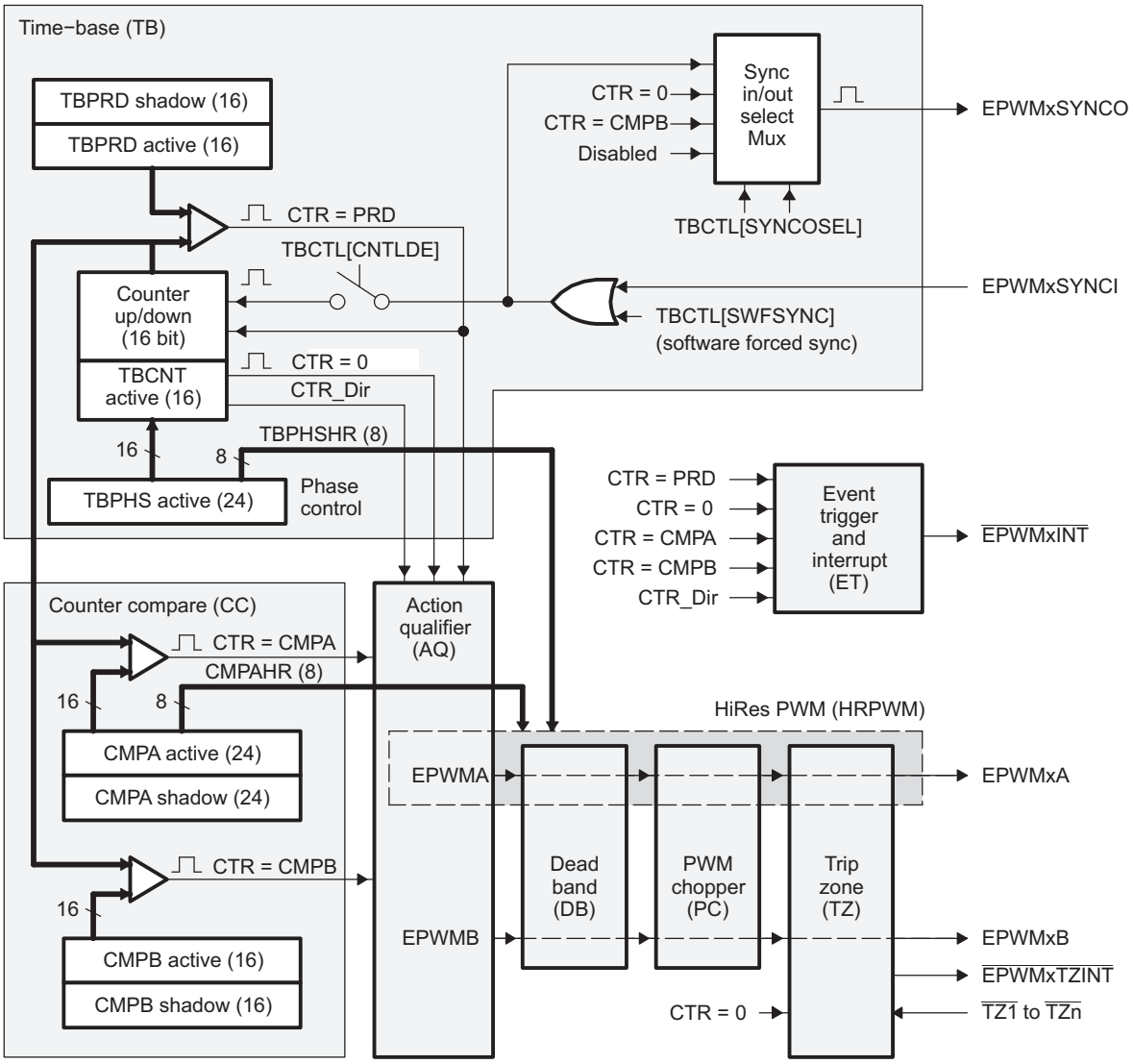


Copyright © 2013–2016, Texas Instruments Incorporated

16.2.10 High-Resolution PWM (HRPWM) Submodule

Figure 16-43 shows the high-resolution PWM (HRPWM) submodule in the ePWM system. Some devices include the high-resolution PWM submodule, see your device-specific data manual to determine which ePWM instances include this feature.

Figure 16-43. HRPWM System Interface



Copyright © 2013–2016, Texas Instruments Incorporated

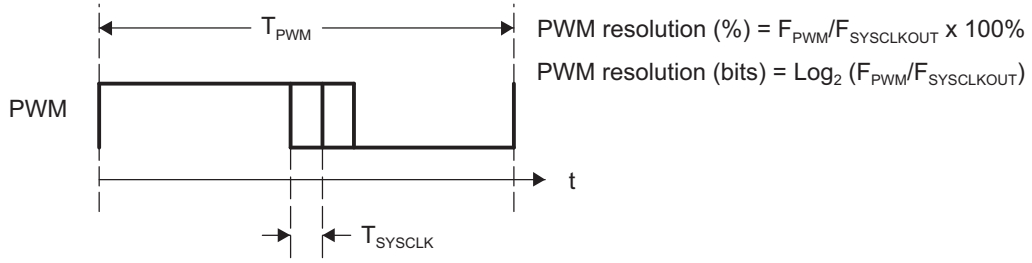
16.2.10.1 Purpose of the High-Resolution PWM Submodule

The enhanced high-resolution pulse-width modulator (eHRPWM) extends the time resolution capabilities of the conventionally derived digital pulse-width modulator (PWM). HRPWM is typically used when PWM resolution falls below ~9-10 bits. The key features of HRPWM are:

- Extended time resolution capability
- Used in both duty cycle and phase-shift control methods
- Finer time granularity control or edge positioning using extensions to the Compare A and Phase registers
- Implemented using the A signal path of PWM, that is, on the EPWMxA output. EPWMxB output has conventional PWM capabilities

The ePWM peripheral is used to perform a function that is mathematically equivalent to a digital-to-analog converter (DAC). As shown in Figure 16-44, the effective resolution for conventionally generated PWM is a function of PWM frequency (or period) and system clock frequency.

Figure 16-44. Resolution Calculations for Conventionally Generated PWM



If the required PWM operating frequency does not offer sufficient resolution in PWM mode, you may want to consider HRPWM. As an example of improved performance offered by HRPWM, Table 16-31 shows resolution in bits for various PWM frequencies. Table 16-31 values assume a MEP step size of 180 ps. See your device-specific data manual for typical and maximum performance specifications for the MEP.

Table 16-31. Resolution for PWM and HRPWM

	Regular Resolution (PWM)		High Resolution (HRPWM)	
PWM Frequency (kHz)	Bits	%	Bits	%
20	12.3	0.0	18.1	0.000
50	11.0	0.0	16.8	0.001
100	10.0	0.1	15.8	0.002
150	9.4	0.2	15.2	0.003
200	9.0	0.2	14.8	0.004
250	8.6	0.3	14.4	0.005
500	7.6	0.5	13.8	0.007
1000	6.6	1.0	12.4	0.018
1500	6.1	1.5	11.9	0.027
2000	5.6	2.0	11.4	0.036

Although each application may differ, typical low-frequency PWM operation (below 250 kHz) may not require HRPWM. HRPWM capability is most useful for high-frequency PWM requirements of power conversion topologies such as:

- Single-phase buck, boost, and flyback
- Multi-phase buck, boost, and flyback
- Phase-shifted full bridge
- Direct modulation of D-Class power amplifiers

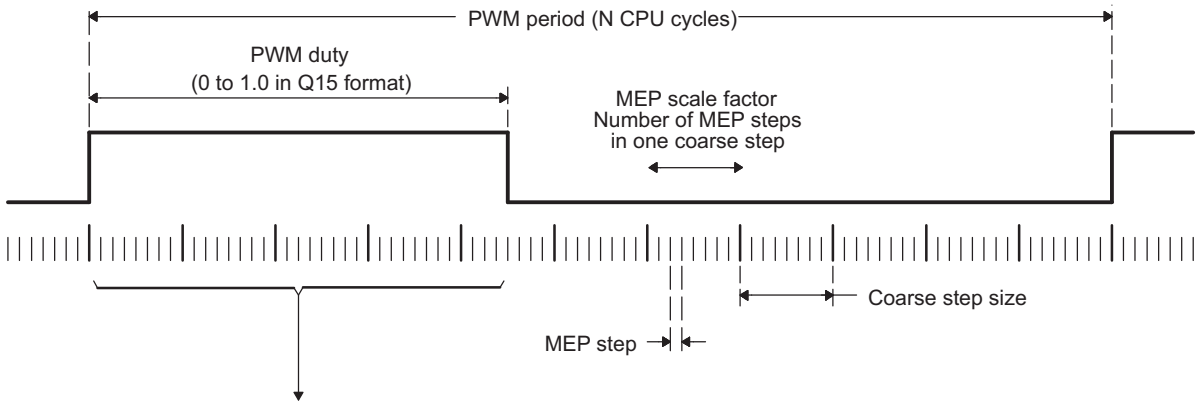
Copyright © 2013–2016, Texas Instruments Incorporated

16.2.10.2 Architecture of the High-Resolution PWM Submodule

The HRPWM is based on micro edge positioner (MEP) technology. MEP logic is capable of positioning an edge very finely by sub-dividing one coarse system clock of a conventional PWM generator. The time step accuracy is on the order of 150 ps. The HRPWM also has a self-check software diagnostics mode to check if the MEP logic is running optimally, under all operating conditions.

Figure 16-45 shows the relationship between one coarse system clock and edge position in terms of MEP steps, which are controlled via an 8-bit field in the Compare A extension register (CMPAHR).

Figure 16-45. Operating Logic Using MEP



Number of coarse steps = integer(PWMduty * PWMperiod)
Number of MEP steps = fraction(PWMduty * PWMperiod) * (MEPScaleFactor)

16−bit CMPA register value	= number of coarse steps
16−bit CMPAHR register value	= (number of MEP steps) << 8 + 0x180 (rounding) (A)

A For MEP range and rounding adjustment.

To generate an HRPWM waveform, configure the TBM, CCM, and AQM registers as you would to generate a conventional PWM of a given frequency and polarity. The HRPWM works together with the TBM, CCM, and AQM registers to extend edge resolution, and should be configured accordingly. Although many programming combinations are possible, only a few are needed and practical.

16.2.10.3 Controlling and Monitoring the High-Resolution PWM Submodule

The MEP of the HRPWM is controlled by two extension registers, each 8-bits wide. These two HRPWM registers are concatenated with the 16-bit TBPHS and CMPA registers used to control PWM operation.

- TBPHSHR - Time-Base Phase High-Resolution Register
- CMPAHR - Counter-Compare A High-Resolution Register

Table 16-32 lists the registers used to control and monitor the high-resolution PWM submodule.

Table 16-32. HRPWM Submodule Registers

Acronym	Register Description	Address Offset	Shadowed
TBPHSHR	Extension Register for HRPWM Phase	4h	No
CMPAHR	Extension Register for HRPWM Duty	10h	Yes
HRCNFG	HRPWM Configuration Register	1040h	No

Copyright © 2013–2016, Texas Instruments Incorporated

16.2.10.4 Configuring the High-Resolution PWM Submodule

Once the ePWM has been configured to provide conventional PWM of a given frequency and polarity, the HRPWM is configured by programming the HRCNFG register located at offset address 1040h. This register provides configuration options for the following key operating modes:

- **Edge Mode:** The MEP can be programmed to provide precise position control on the rising edge (RE), falling edge (FE), or both edges (BE) at the same time. FE and RE are used for power topologies requiring duty cycle control, while BE is used for topologies requiring phase shifting, for example, phase shifted full bridge.
- **Control Mode:** The MEP is programmed to be controlled either from the CMPAHR register (duty cycle control) or the TBPHSHR register (phase control). RE or FE control mode should be used with CMPAHR register. BE control mode should be used with TBPHSHR register.
- **Shadow Mode:** This mode provides the same shadowing (double buffering) option as in regular PWM mode. This option is valid only when operating from the CMPAHR register and should be chosen to be the same as the regular load option for the CMPA register. If TBPHSHR is used, then this option has no effect.

16.2.10.5 Operational Highlights for the High-Resolution PWM Submodule

The MEP logic is capable of placing an edge in one of 255 (8 bits) discrete time steps, each of which has a time resolution on the order of 150 ps. The MEP works with the TBM and CCM registers to be certain that time steps are optimally applied and that edge placement accuracy is maintained over a wide range of PWM frequencies, system clock frequencies and other operating conditions. Table 16-33 shows the typical range of operating frequencies supported by the HRPWM.

Table 16-33. Relationship Between MEP Steps, PWM Frequency and Resolution

System (MHz)	MEP Steps Per SYSCLKOUT(1) (2) (3)	PWM Minimum (Hz)(4)	PWM Maximum (MHz)	Resolution at Maximum (Bits)(5)
50.0	111	763	2.50	11.1
60.0	93	916	3.00	10.9
70.0	79	1068	3.50	10.6
80.0	69	1221	4.00	10.4
90.0	62	1373	4.50	10.3
100.0	56	1526	5.00	10.1

(1) System frequency = SYSCLKOUT, that is, CPU clock. TBCLK = SYSCLKOUT
(2) Table data based on a MEP time resolution of 180 ps (this is an example value)
(3) MEP steps applied = $T_{SYSCLKOUT}$/180 ps in this example.
(4) PWM minimum frequency is based on a maximum period value, TBPRD = 65 535. PWM mode is asymmetrical up-count.
(5) Resolution in bits is given for the maximum PWM frequency stated.

Copyright © 2013–2016, Texas Instruments Incorporated

16.2.10.5.1 Edge Positioning

In a typical power control loop (switch modes, digital motor control (DMC), uninterruptible power supply (UPS)), a digital controller (PID, 2pole/2zero, lag/lead, etc.) issues a duty command, usually expressed in a per unit or percentage terms.

In the following example, assume that for a particular operating point, the demanded duty cycle is 0.405 or 40.5% on-time and the required converter PWM frequency is 1.25 MHz. In conventional PWM generation with a system clock of 100 MHz, the duty cycle choices are in the vicinity of 40.5%. In Figure 16-46, a compare value of 32 counts (duty = 40%) is the closest to 40.5% that you can attain. This is equivalent to an edge position of 320 ns instead of the desired 324 ns. This data is shown in Table 16-34.

By utilizing the MEP, you can achieve an edge position much closer to the desired point of 324 ns. Table 16-34 shows that in addition to the CMPA value, 22 steps of the MEP (CMPAHR register) will position the edge at 323.96 ns, resulting in almost zero error. In this example, it is assumed that the MEP has a step resolution of 180 ns.

Figure 16-46. Required PWM Waveform for a Requested Duty = 40.5%

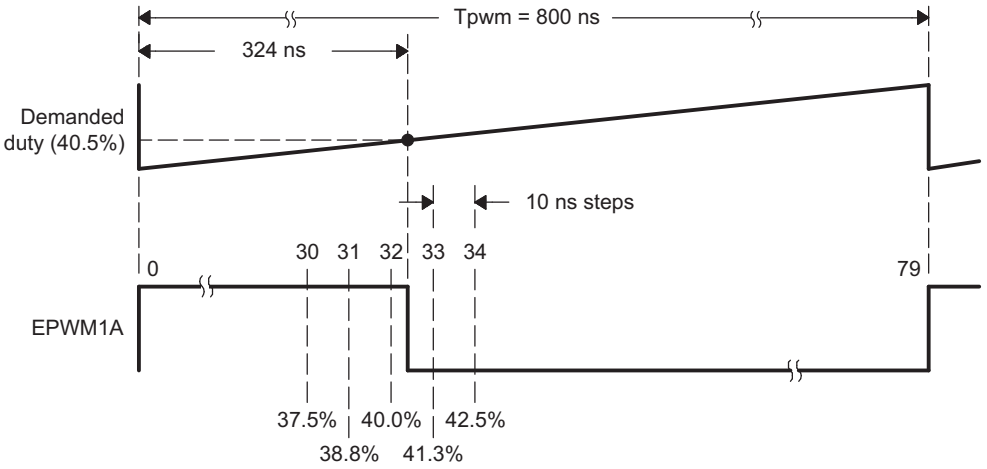


Table 16-34. CMPA vs Duty (left), and [CMPA:CMPAHR] vs Duty (right)

CMPA (count)[1] [2] [3]	DUTY (%)	High Time (ns)	CMPA (count)	CMPAHR (count)	Duty (%)	High Time (ns)
28	35.0	280	32	18	40.405	323.24
29	36.3	290	32	19	40.428	323.42
30	37.5	300	32	20	40.450	323.60
31	38.8	310	32	21	40.473	323.78
32	40.0	320	32	22	40.495	323.96
33	41.3	330	32	23	40.518	324.14
34	42.5	340	32	24	40.540	324.32
			32	25	40.563	324.50
Required			32	26	40.585	324.68
32.40	40.5	324	32	27	40.608	324.86

(1) System clock, SYSCLKOUT and TBCLK = 100 MHz, 10 ns
(2) For a PWM Period register value of 80 counts, PWM Period = 80 × 10 ns = 800 ns, PWM frequency = 1/800 ns = 1.25 MHz
(3) Assumed MEP step size for the above example = 180 ps

Copyright © 2013–2016, Texas Instruments Incorporated

16.2.10.5.2 Scaling Considerations

The mechanics of how to position an edge precisely in time has been demonstrated using the resources of the standard (CMPA) and MEP (CMPAHR) registers. In a practical application, however, it is necessary to seamlessly provide the CPU a mapping function from a per-unit (fractional) duty cycle to a final integer (non-fractional) representation that is written to the [CMPA:CMPAHR] register combination.

To do this, first examine the scaling or mapping steps involved. It is common in control software to express duty cycle in a per-unit or percentage basis. This has the advantage of performing all needed math calculations without concern for the final absolute duty cycle, expressed in clock counts or high time in ns. Furthermore, it makes the code more transportable across multiple converter types running different PWM frequencies.

To implement the mapping scheme, a two-step scaling procedure is required.

Assumptions for this example:

System clock, SYSCLKOUT	=	10 ns (100 MHz)
PWM frequency	=	1.25 MHz (1/800 ns)
Required PWM duty cycle, **PWMDuty**	=	0.405 (40.5%)
PWM period in terms of coarse steps, **PWMperiod** (800 ns/10 ns)	=	80
Number of MEP steps per coarse step at 180 ps (10 ns/180 ps), **MEP_SF**	=	55
Value to keep CMPAHR within the range of 1-255 and fractional rounding constant (default value)	=	180h

Step 1: Percentage Integer Duty value conversion for CMPA register

CMPA register value	=	int(**PWMDuty** × **PWMperiod**); int means integer part
	=	int(0.405 × 80)
	=	int(32.4)
CMPA register value	=	32 (20h)

Step 2: Fractional value conversion for CMPAHR register

CMPAHR register value	=	(frac(**PWMDuty** × **PWMperiod**) × **MEP_SF**) << 8) + 180h; frac means fractional part
	=	(frac(32.4) × 55 <<8) + 180h; Shift is to move the value as CMPAHR high byte
	=	((0.4 × 55) <<8) + 180h
	=	(22 <<8) + 180h
	=	22 × 256 + 180h ; Shifting left by 8 is the same multiplying by 256.
	=	5632 + 180h
	=	1600h + 180h
CMPAHR value	=	1780h; CMPAHR value = 1700h, lower 8 bits will be ignored by hardware.

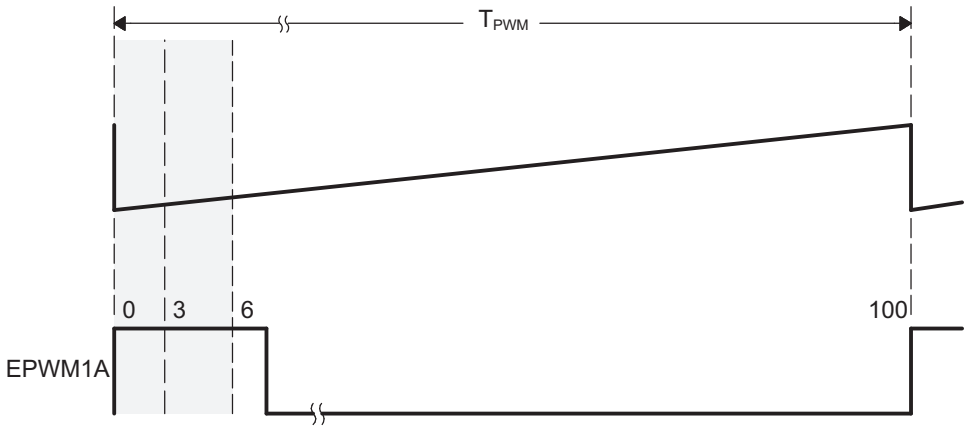
Copyright © 2013–2016, Texas Instruments Incorporated

16.2.10.5.3 Duty Cycle Range Limitation

In high resolution mode, the MEP is not active for 100% of the PWM period. It becomes operational 3 SYSCLK cycles after the period starts.

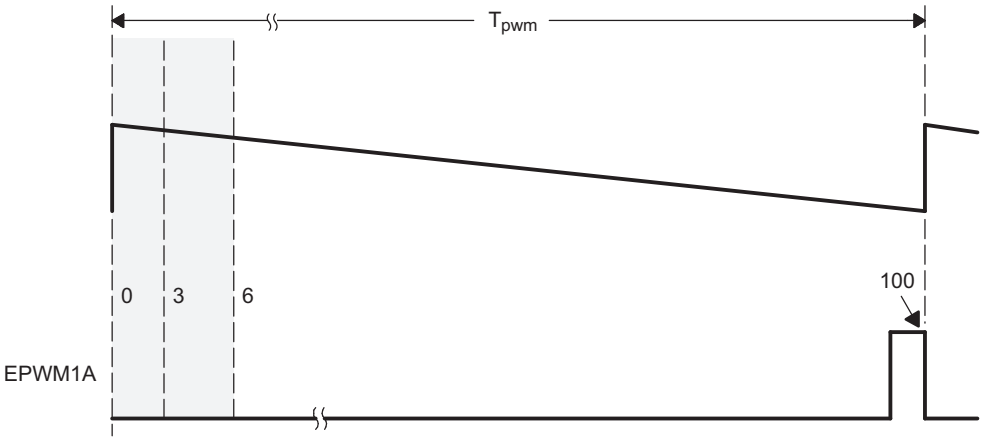
Duty cycle range limitations are illustrated in Figure 16-47. This limitation imposes a lower duty cycle limit on the MEP. For example, precision edge control is not available all the way down to 0% duty cycle. Although for the first 3 or 6 cycles, the HRPWM capabilities are not available, regular PWM duty control is still fully operational down to 0% duty. In most applications this should not be an issue as the controller regulation point is usually not designed to be close to 0% duty cycle.

Figure 16-47. Low % Duty Cycle Range Limitation Example When PWM Frequency = 1 MHz



If the application demands HRPWM operation in the low percent duty cycle region, then the HRPWM can be configured to operate in count-down mode with the rising edge position (REP) controlled by the MEP. This is illustrated in Figure 16-48. In this case low percent duty limitation is no longer an issue.

Figure 16-48. High % Duty Cycle Range Limitation Example when PWM Frequency = 1 MHz



Copyright © 2013–2016, Texas Instruments Incorporated

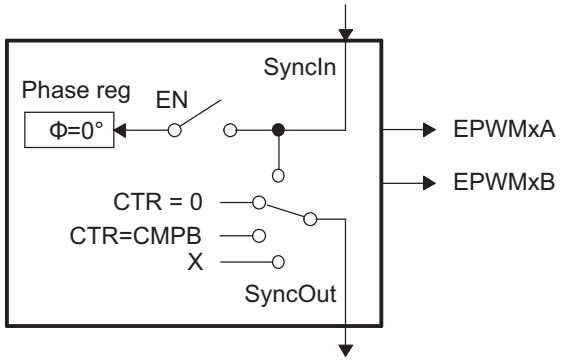
16.3 Applications to Power Topologies

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.

16.3.1 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 16-49. 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 16-49. Simplified ePWM Module



Copyright © 2013–2016, Texas Instruments Incorporated

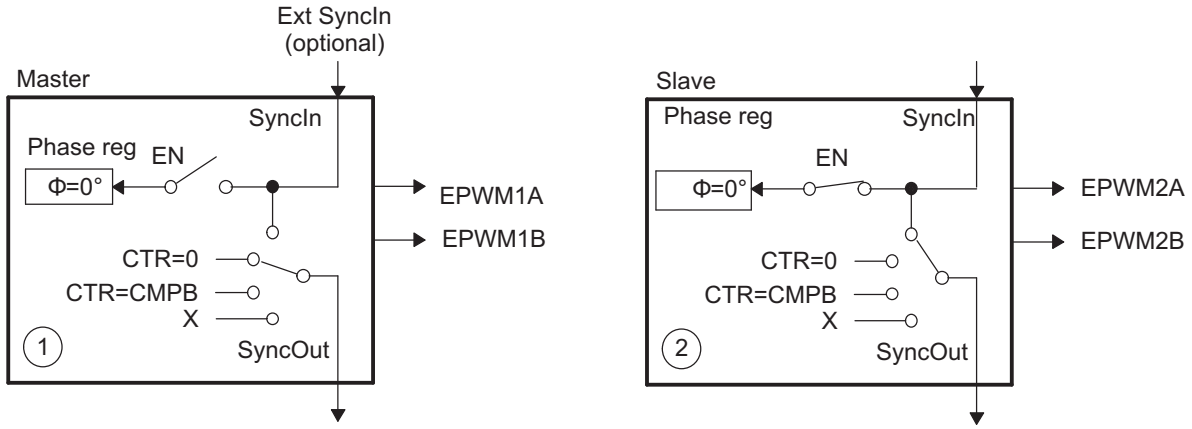
16.3.2 Key Configuration Capabilities

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 16-50.

Figure 16-50. EPWM1 Configured as a Typical Master, EPWM2 Configured as a Slave

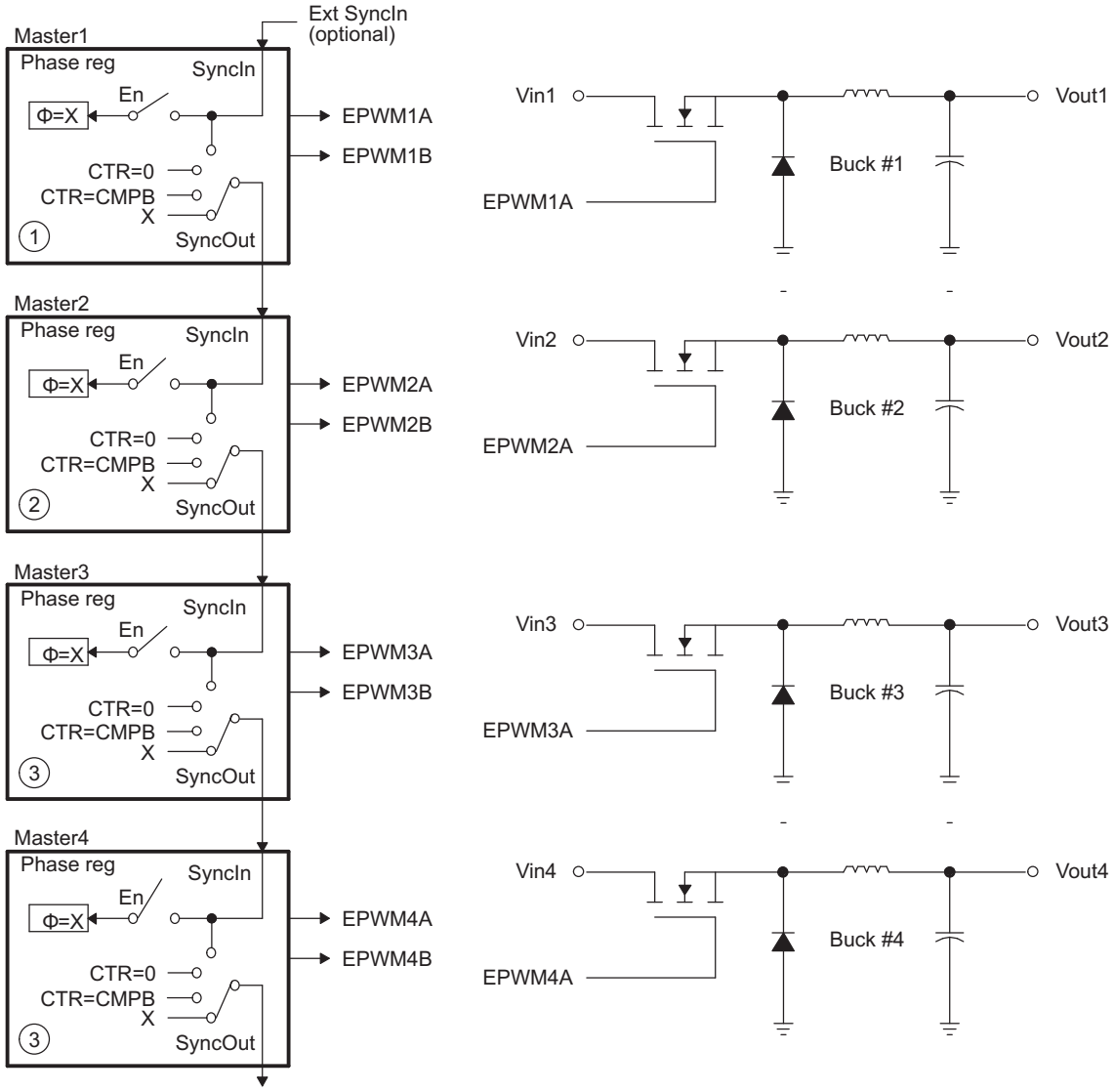


Copyright © 2013–2016, Texas Instruments Incorporated

16.3.3 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 16-51 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 16-52 shows the waveforms generated by the setup shown in Figure 16-51; note that only three waveforms are shown, although there are four stages.

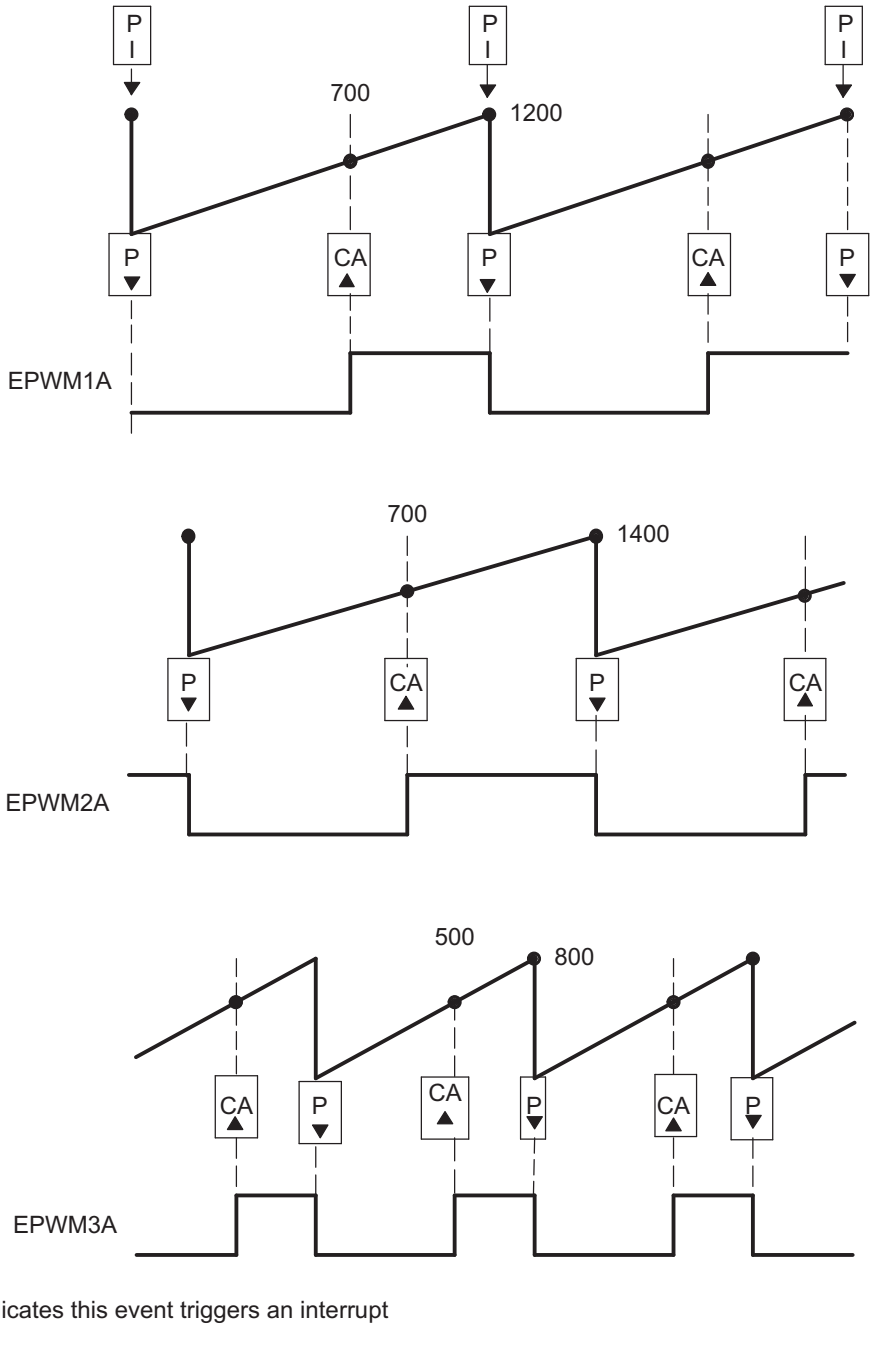
Figure 16-51. Control of Four Buck Stages. (Note: $F_{PWM1} \neq F_{PWM2} \neq F_{PWM3} \neq F_{PWM4}$)



NOTE: Θ = X indicates value in phase register is a "don't care"

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 16-52. Buck Waveforms for Figure 16-51 (Note: Only three bucks shown here)



Copyright © 2013–2016, Texas Instruments Incorporated

Table 16-35. EPWM1 Initialization for Figure 16-52

Register	Bit	Value	Comments
TBPRD	TBPRD	1200 (4B0h)	Period = 1201 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCTL	CTRMODE	TB_UP	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	PRD	AQ_CLEAR	
	CAU	AQ_SET	

Table 16-36. EPWM2 Initialization for Figure 16-52

Register	Bit	Value	Comments
TBPRD	TBPRD	1400 (578h)	Period = 1401 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCTL	CTRMODE	TB_UP	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	PRD	AQ_CLEAR	
	CAU	AQ_SET	

Table 16-37. EPWM3 Initialization for Figure 16-52

Register	Bit	Value	Comments
TBPRD	TBPRD	800 (320h)	Period = 801 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCTL	CTRMODE	TB_UP	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_DISABLE	
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	PRD	AQ_CLEAR	
	CAU	AQ_SET	

Copyright © 2013–2016, Texas Instruments Incorporated

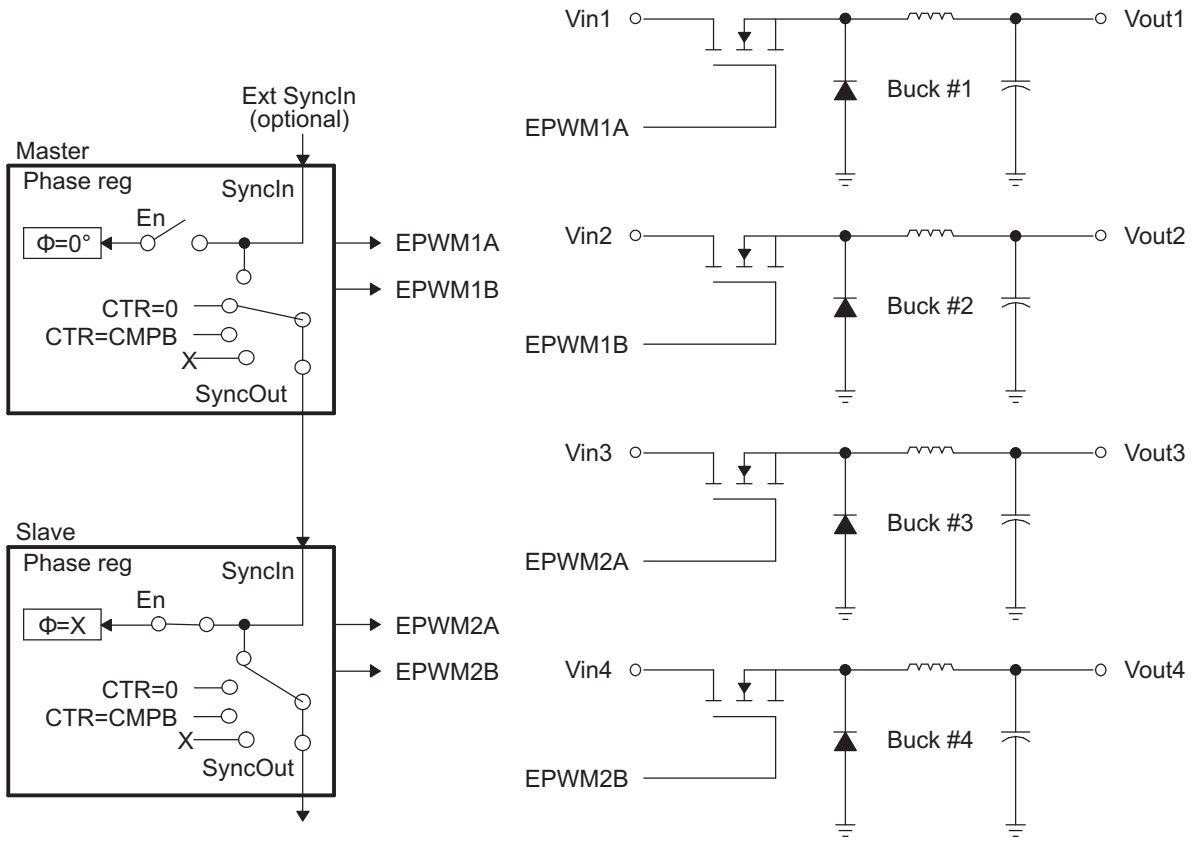
Example 16-3. Configuration for Example in Figure 16-52

```
// Run Time (Note: Example execution of one run-time instance)
//=====================================================
    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
```

16.3.4 Controlling Multiple Buck Converters With Same Frequencies

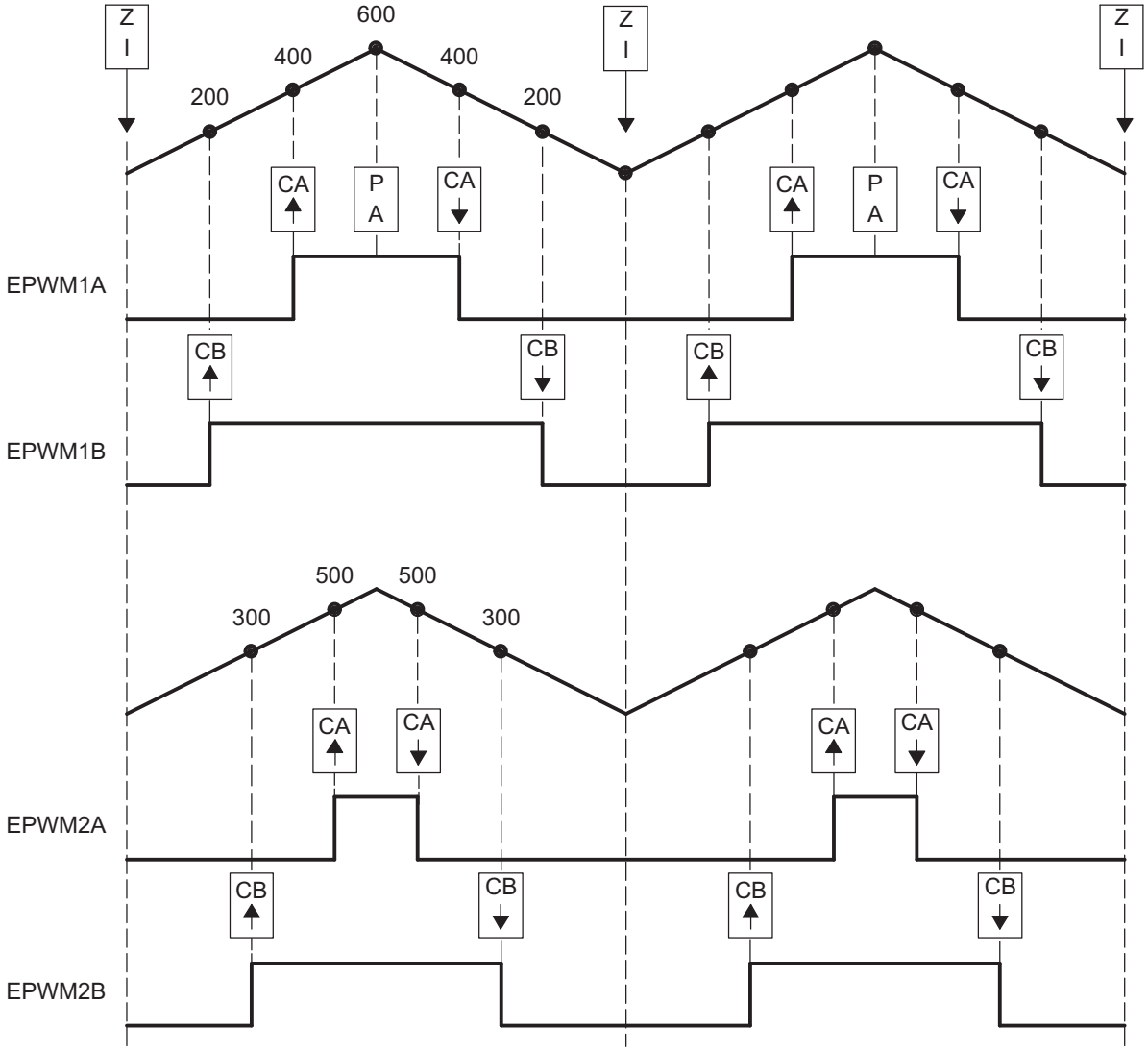
If synchronization is a requirement, ePWM module 2 can be configured as a slave and can operate at integer multiple (N) frequencies of module 1. The sync signal from master to slave ensures these modules remain locked. Figure 16-53 shows such a configuration; Figure 16-54 shows the waveforms generated by the configuration.

Figure 16-53. Control of Four Buck Stages. (Note: $F_{PWM2} = N \times F_{PWM1}$)



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 16-54. Buck Waveforms for Figure 16-53 (Note: $F_{PWM2} = F_{PWM1}$)



Copyright © 2013–2016, Texas Instruments Incorporated

Table 16-38. EPWM1 Initialization for Figure 16-53

Register	Bit	Value	Comments
TBPRD	TBPRD	600 (258h)	Period = 1200 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_CTR_ZERO	Sync down-stream module
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	CAU	AQ_SET	Set actions for EPWM1A
	CAD	AQ_CLEAR	
AQCTLB	CBU	AQ_SET	Set actions for EPWM1B
	CBD	AQ_CLEAR	

Table 16-39. EPWM2 Initialization for Figure 16-53

Register	Bit	Value	Comments
TBPRD	TBPRD	600 (258h)	Period = 1200 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_ENABLE	Phase loading enabled
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_IN	Sync flow-through
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	CAU	AQ_SET	Set actions for EPWM2A
	CAD	AQ_CLEAR	
AQCTLB	CBU	AQ_SET	Set actions for EPWM2B
	CBD	AQ_CLEAR	

Example 16-4. Code Snippet for Configuration in Figure 16-53

```
// 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
```

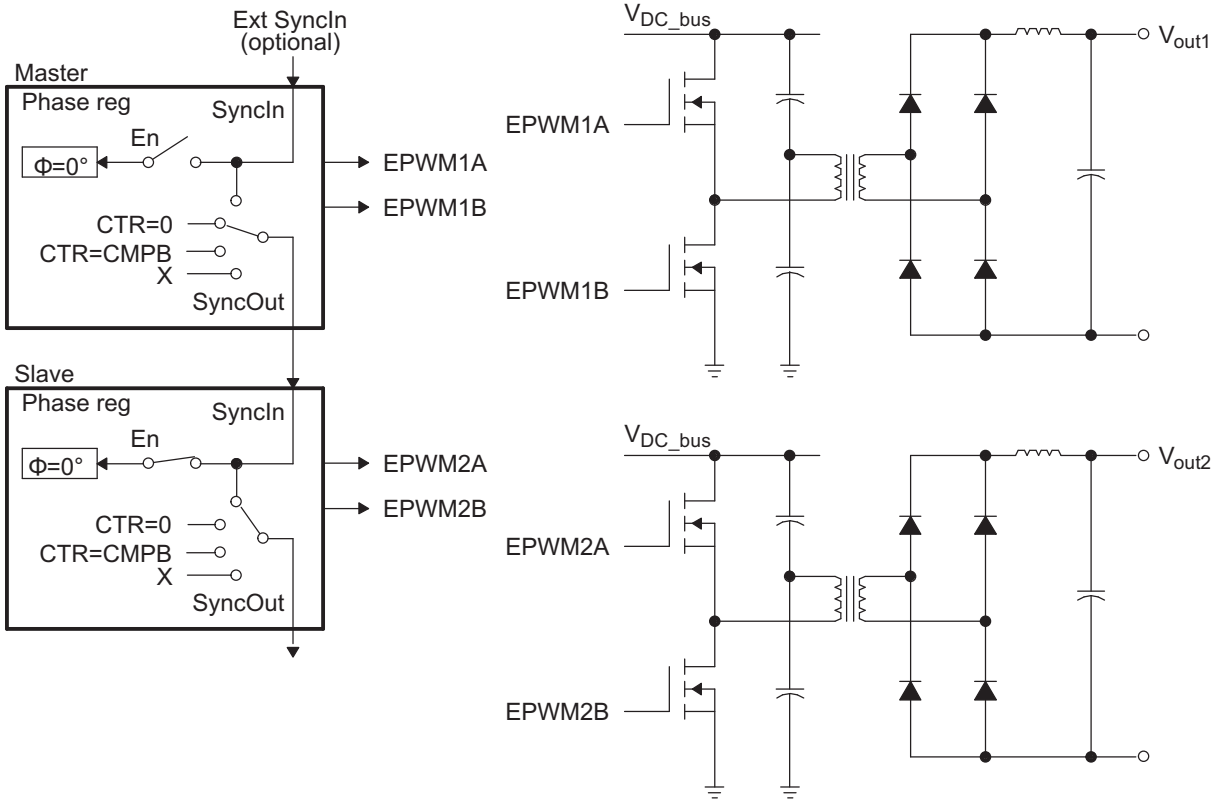

Copyright © 2013–2016, Texas Instruments Incorporated

16.3.5 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 16-55 shows control of two synchronized Half-H bridge stages where stage 2 can operate at integer multiple (N) frequencies of stage 1. Figure 16-56 shows the waveforms generated by the configuration shown in Figure 16-55.

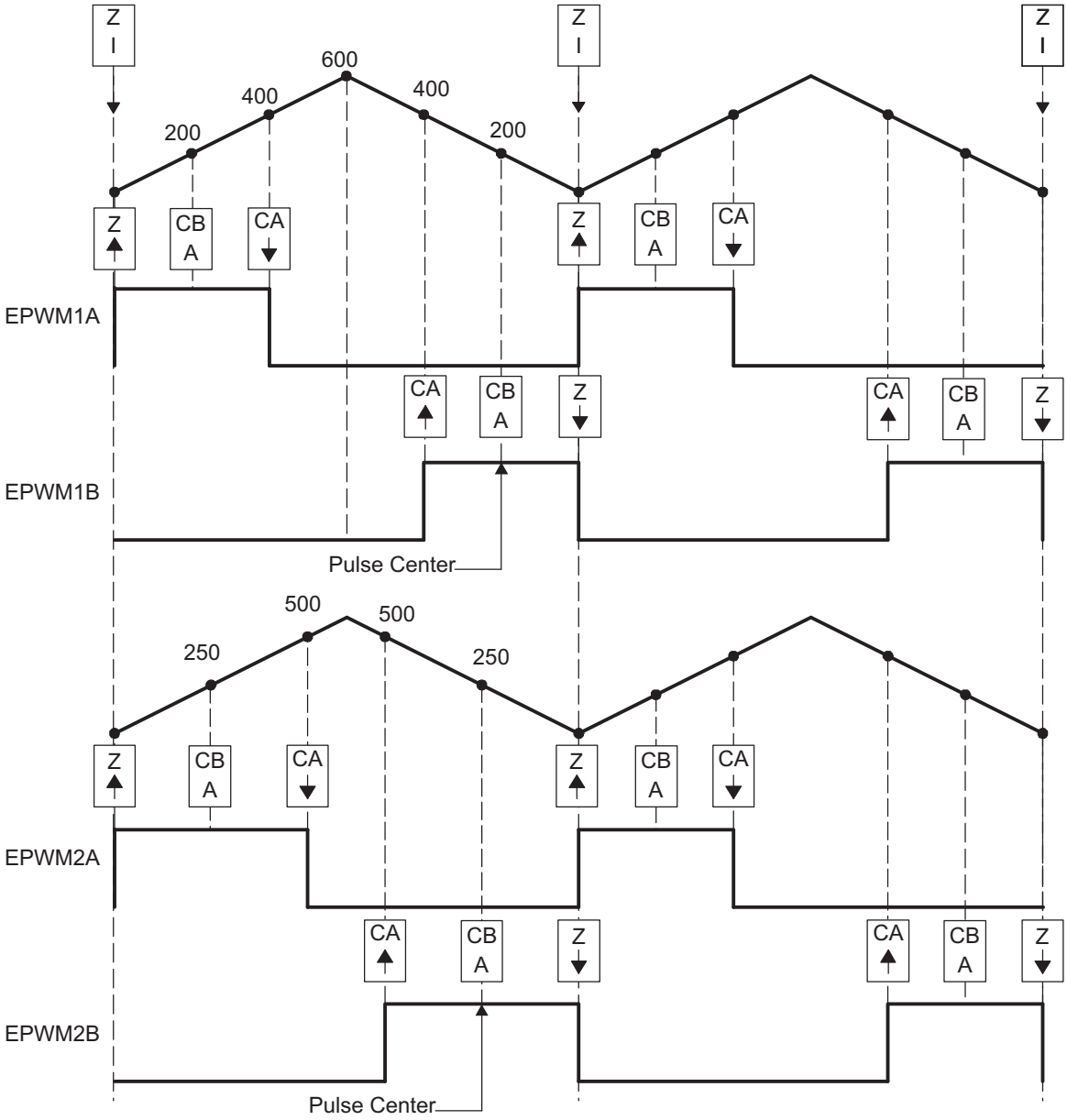
Module 2 (slave) is configured for Sync flow-through; if required, this configuration allows for a third Half-H bridge to be controlled by PWM module 3 and also, most importantly, to remain in synchronization with master module 1.

Figure 16-55. Control of Two Half-H Bridge Stages ($F_{PWM2} = N \times F_{PWM1}$)



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 16-56. Half-H Bridge Waveforms for Figure 16-55 (Note: $F_{PWM2} = F_{PWM1}$)



Copyright © 2013–2016, Texas Instruments Incorporated

Table 16-40. EPWM1 Initialization for Figure 16-55

Register	Bit	Value	Comments
TBPRD	TBPRD	600 (258h)	Period = 1200 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_CTR_ZERO	Sync down-stream module
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	ZRO	AQ_SET	Set actions for EPWM1A
	CAU	AQ_CLEAR	
AQCTLB	ZRO	AQ_CLEAR	Set actions for EPWM1B
	CAD	AQ_SET	

Table 16-41. EPWM2 Initialization for Figure 16-55

Register	Bit	Value	Comments
TBPRD	TBPRD	600 (258h)	Period = 1200 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_ENABLE	Phase loading enabled
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_IN	Sync flow-through
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	ZRO	AQ_SET	Set actions for EPWM2A
	CAU	AQ_CLEAR	
AQCTLB	ZRO	AQ_CLEAR	Set actions for EPWM2B
	CAD	AQ_SET	

Example 16-5. Code Snippet for Configuration in Figure 16-55

```
// 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 = 250;           // adjust duty for output EPWM2B
```

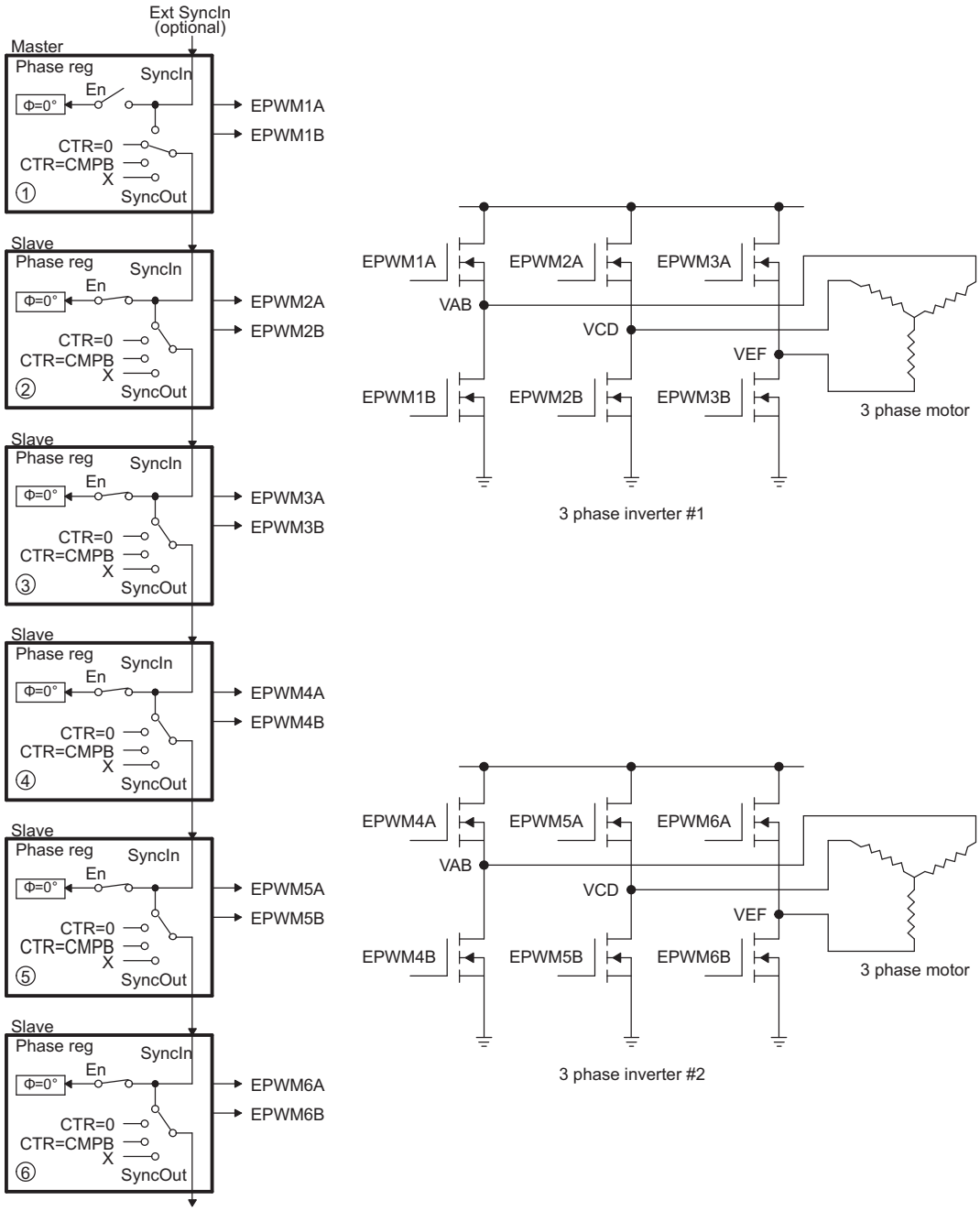

Copyright © 2013–2016, Texas Instruments Incorporated

16.3.6 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 16-57 shows how six PWM modules can control two independent 3-phase Inverters; each running a motor.

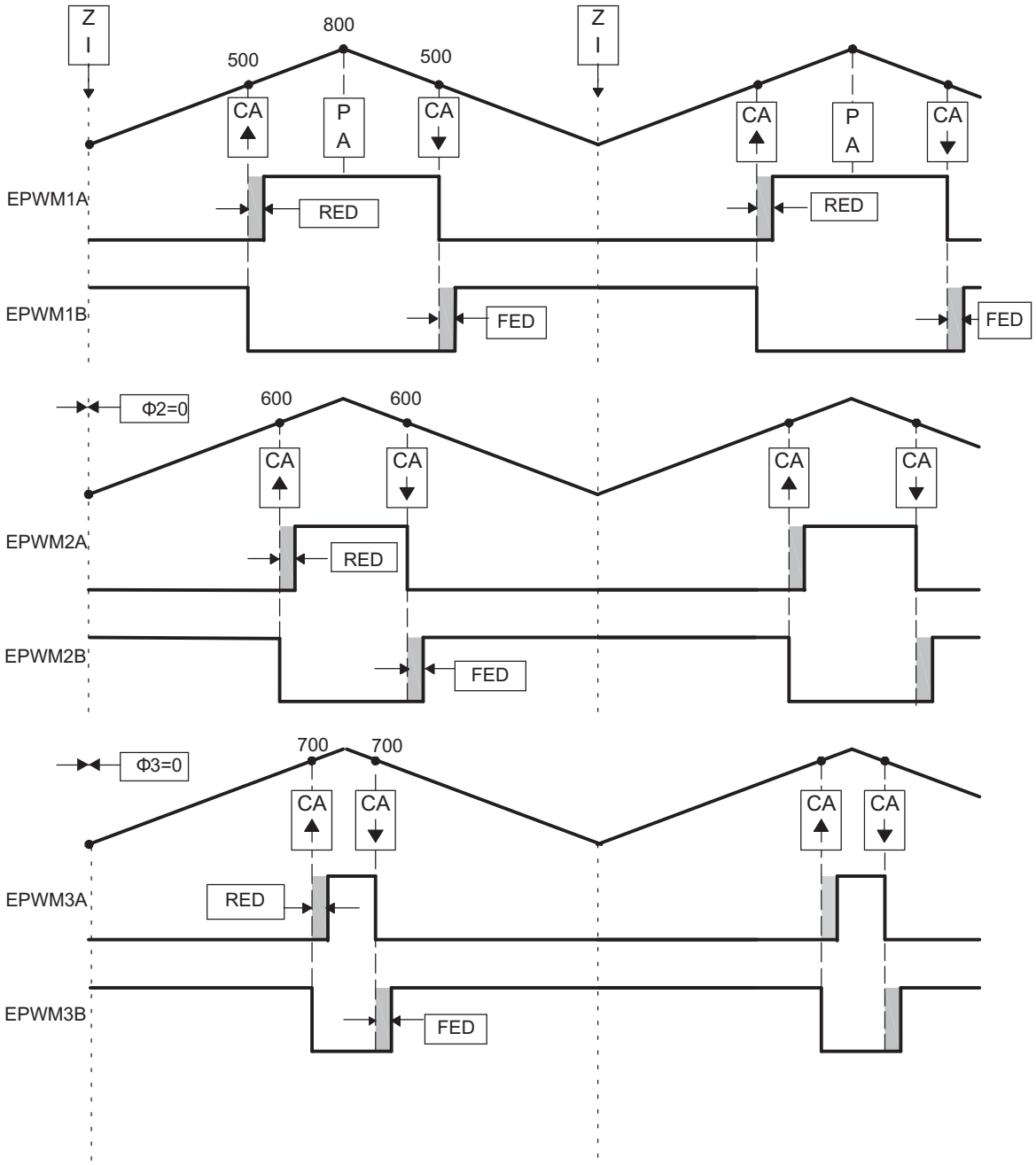
As in the cases shown in the previous sections, we have a choice of running each inverter at a different frequency (module 1 and module 4 are masters as in Figure 16-57), or both inverters can be synchronized by using one master (module 1) and five slaves. In this case, the frequency of modules 4, 5, and 6 (all equal) can be integer multiples of the frequency for modules 1, 2, 3 (also all equal).

Figure 16-57. Control of Dual 3-Phase Inverter Stages as Is Commonly Used in Motor Control



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 16-58. 3-Phase Inverter Waveforms for Figure 16-57 (Only One Inverter Shown)



Copyright © 2013–2016, Texas Instruments Incorporated

Table 16-42. EPWM1 Initialization for Figure 16-57

Register	Bit	Value	Comments
TBPRD	TBPRD	800 (320h)	Period = 1600 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_CTR_ZERO	Sync down-stream module
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	CAU	AQ_SET	Set actions for EPWM1A
	CAD	AQ_CLEAR	
DBCTL	MODE	DB_FULL_ENABLE	Enable Dead-band module
	POLSEL	DB_ACTV_HIC	Active Hi complementary
DBFED	DBFED	50	FED = 50 TBCLKs
	DBRED	50	RED = 50 TBCLKs

Table 16-43. EPWM2 Initialization for Figure 16-57

Register	Bit	Value	Comments
TBPRD	TBPRD	800 (320h)	Period = 1600 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_ENABLE	Slave module
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_IN	Sync flow-through
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	CAU	AQ_SET	Set actions for EPWM2A
	CAD	AQ_CLEAR	
DBCTL	MODE	DB_FULL_ENABLE	Enable Dead-band module
	POLSEL	DB_ACTV_HIC	Active Hi complementary
DBFED	DBFED	50	FED = 50 TBCLKs
	DBRED	50	RED = 50 TBCLKs

Copyright © 2013–2016, Texas Instruments Incorporated

Table 16-44. EPWM3 Initialization for Figure 16-57

Register	Bit	Value	Comments
TBPRD	TBPRD	800 (320h)	Period = 1600 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_ENABLE	Slave module
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_IN	Sync flow-through
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	CAU	AQ_SET	Set actions for EPWM3A
	CAD	AQ_CLEAR	
DBCTL	MODE	DB_FULL_ENABLE	Enable Dead-band module
	POLSEL	DB_ACTV_HIC	Active Hi complementary
DBFED	DBFED	50	FED = 50 TBCLKs
	DBRED	50	RED = 50 TBCLKs

Example 16-6. Code Snippet for Configuration in Figure 16-57

```
// Run Time (Note: Example execution of one run-time instance)
//=====================================================
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 © 2013–2016, Texas Instruments Incorporated

16.3.7 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 TBCNT register. To illustrate this concept, Figure 16-59 shows a master and slave module with a phase relationship of 120°, that is, the slave leads the master.

Figure 16-59. Configuring Two PWM Modules for Phase Control

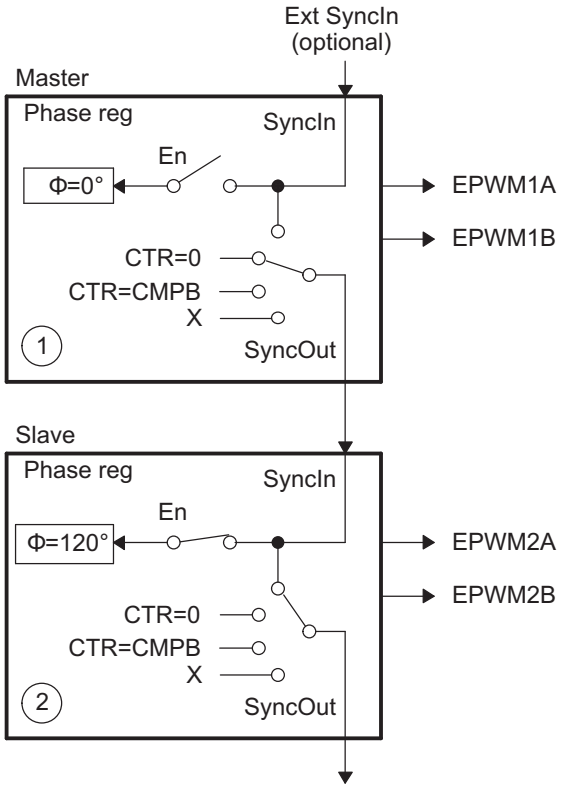
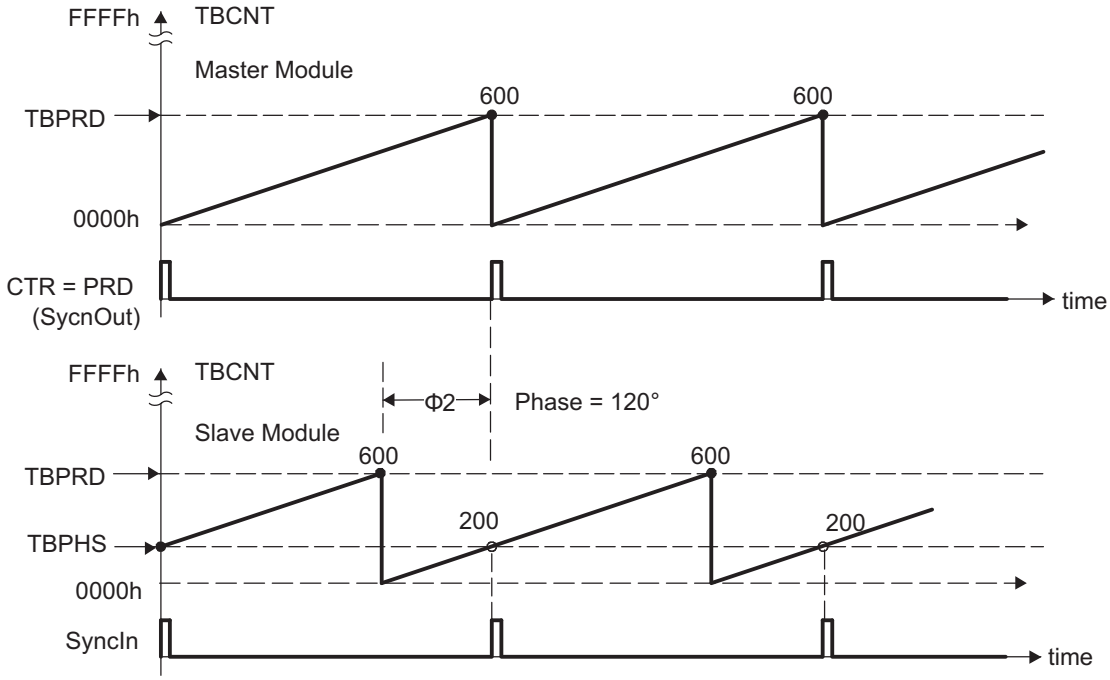


Figure 16-60 shows the associated timing waveforms for this configuration. Here, TBPRD = 600 for both master and slave. For the slave, TBPHS = 200 (200/600 × 360° = 120°). Whenever the master generates a SyncIn pulse (CTR = PRD), the value of TBPHS = 200 is loaded into the slave TBCNT register so the slave time-base is always leading the master's time-base by 120°.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 16-60. Timing Waveforms Associated With Phase Control Between 2 Modules



16.3.8 Controlling a 3-Phase Interleaved DC/DC Converter

A popular power topology that makes use of phase-offset between modules is shown in Figure 16-61. 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) × (M - 1)

Where:

N = number of phases

M = PWM module number

For example, for the 3-phase case (N = 3), TBPRD = 600,

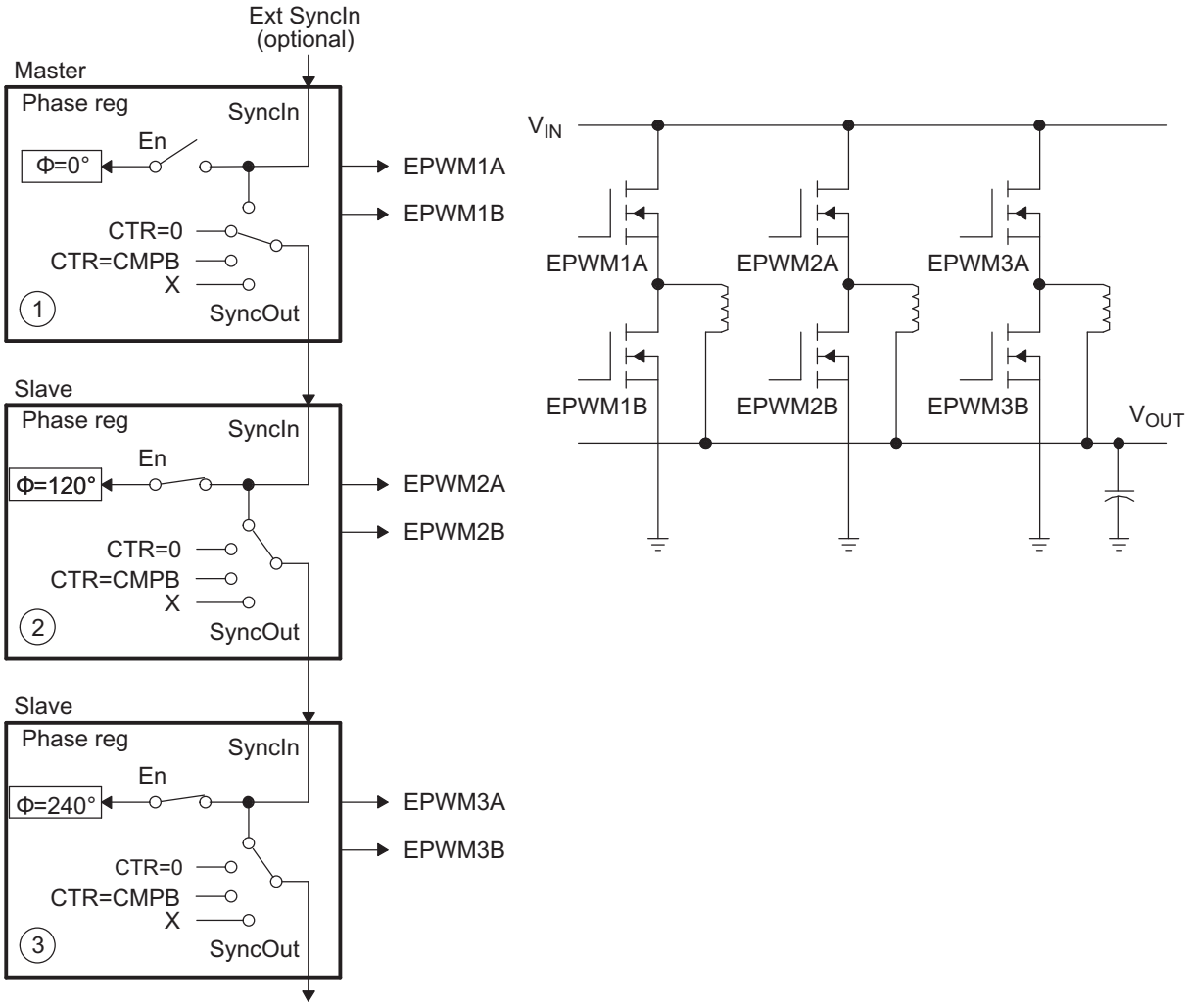
TBPHS(3,2) = (600/3) × (2 - 1) = 200 × 1 = 200 (Phase value for Slave module 2)

TBPHS(3,3) = (600/3) × (3 - 1) = 200 × 2 = 400 (Phase value for Slave module 3)

Figure 16-62 shows the waveforms for the configuration in Figure 16-61.

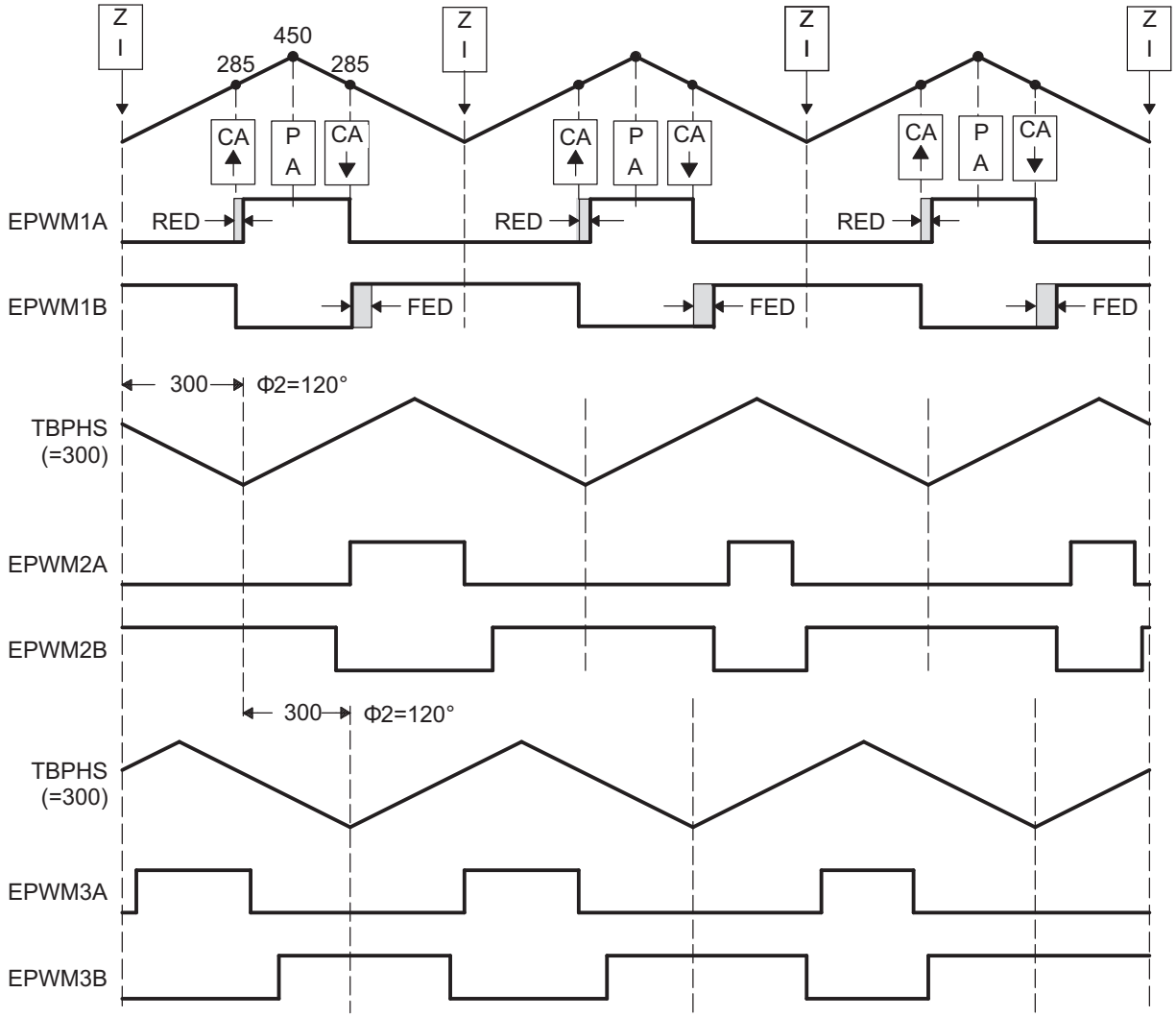
Copyright © 2013–2016, Texas Instruments Incorporated

Figure 16-61. Control of a 3-Phase Interleaved DC/DC Converter



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 16-62. 3-Phase Interleaved DC/DC Converter Waveforms for Figure 16-61



Copyright © 2013–2016, Texas Instruments Incorporated

Table 16-45. EPWM1 Initialization for Figure 16-61

Register	Bit	Value	Comments
TBPRD	TBPRD	450 (1C2h)	Period = 900 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_CTR_ZERO	Sync down-stream module
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	CAU	AQ_SET	Set actions for EPWM1A
	CAD	AQ_CLEAR	
DBCTL	MODE	DB_FULL_ENABLE	Enable Dead-band module
	POLSEL	DB_ACTV_HIC	Active Hi complementary
DBFED	DBFED	20	FED = 20 TBCLKs
	DBRED	20	RED = 20 TBCLKs

Table 16-46. EPWM2 Initialization for Figure 16-61

Register	Bit	Value	Comments
TBPRD	TBPRD	450 (1C2h)	Period = 900 TBCLK counts
TBPHS	TBPHS	300	Phase = (300/900) × 360 = 120°
TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_ENABLE	Slave module
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_IN	Sync flow-through
	PHSDIR	TB_DOWN	Count DOWN on sync
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	CAU	AQ_SET	Set actions for EPWM2A
	CAD	AQ_CLEAR	
DBCTL	MODE	DB_FULL_ENABLE	Enable Dead-band module
	POLSEL	DB_ACTV_HIC	Active Hi complementary
DBFED	DBFED	20	FED = 20 TBCLKs
	DBRED	20	RED = 20 TBCLKs

Copyright © 2013–2016, Texas Instruments Incorporated

Table 16-47. EPWM3 Initialization for Figure 16-61

Register	Bit	Value	Comments
TBPRD	TBPRD	450 (1C2h)	Period = 900 TBCLK counts
TBPHS	TBPHS	300	Phase = (300/900) × 360 = 120°
TBCTL	CTRMODE	TB_UPDOWN	
	PHSEN	TB_ENABLE	Slave module
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_IN	Sync flow-through
	PHSDIR	TB_UP	Count UP on sync
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	CAU	AQ_SET	Set actions for EPWM3A
	CAD	AQ_CLEAR	
DBCTL	MODE	DB_FULL_ENABLE	Enable Dead-band module
	POLSEL	DB_ACTV_HIC	Active Hi complementary
DBFED	DBFED	20	FED = 20 TBCLKs
	DBRED	20	RED = 20 TBCLKs

Example 16-7. Code Snippet for Configuration in Figure 16-61

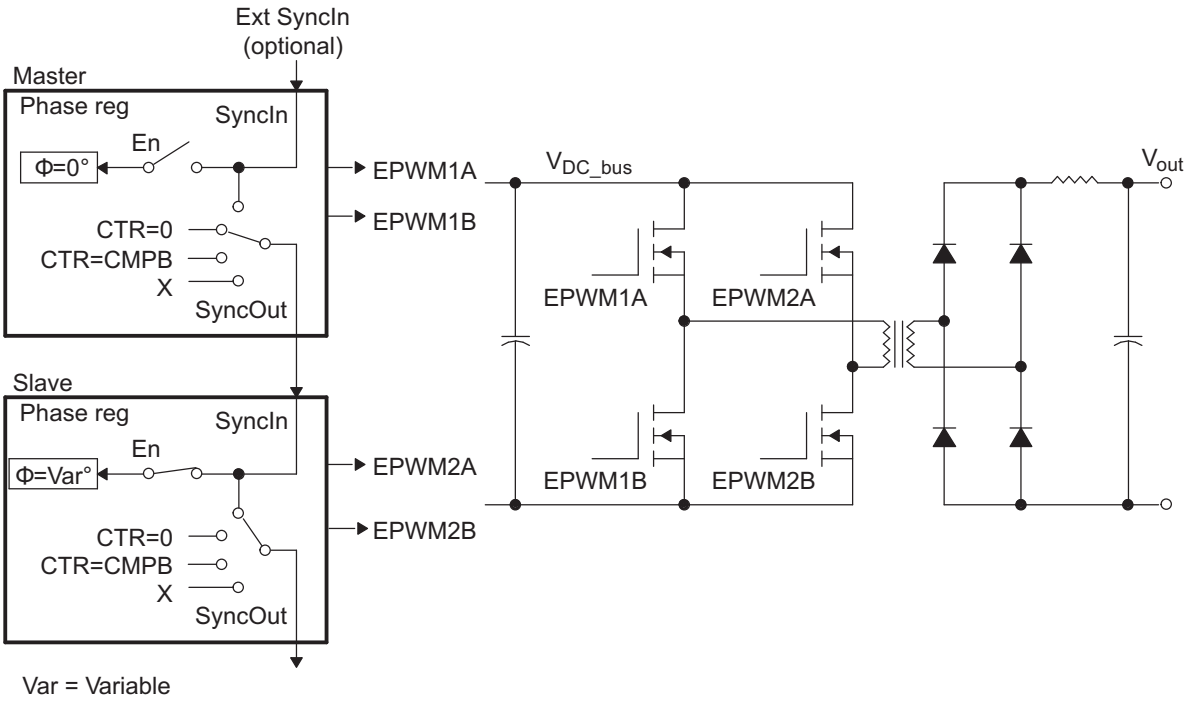
```
// Run Time (Note: Example execution of one run-time instance)
//=====================================================
EPwm1Regs.CMPA.half.CMPA = 285;         // adjust duty for output EPWM1A
EPwm2Regs.CMPA.half.CMPA = 285;         // adjust duty for output EPWM2A
EPwm3Regs.CMPA.half.CMPA = 285;         // adjust duty for output EPWM3A
```


Copyright © 2013–2016, Texas Instruments Incorporated

16.3.9 Controlling Zero Voltage Switched Full Bridge (ZVSFB) Converter

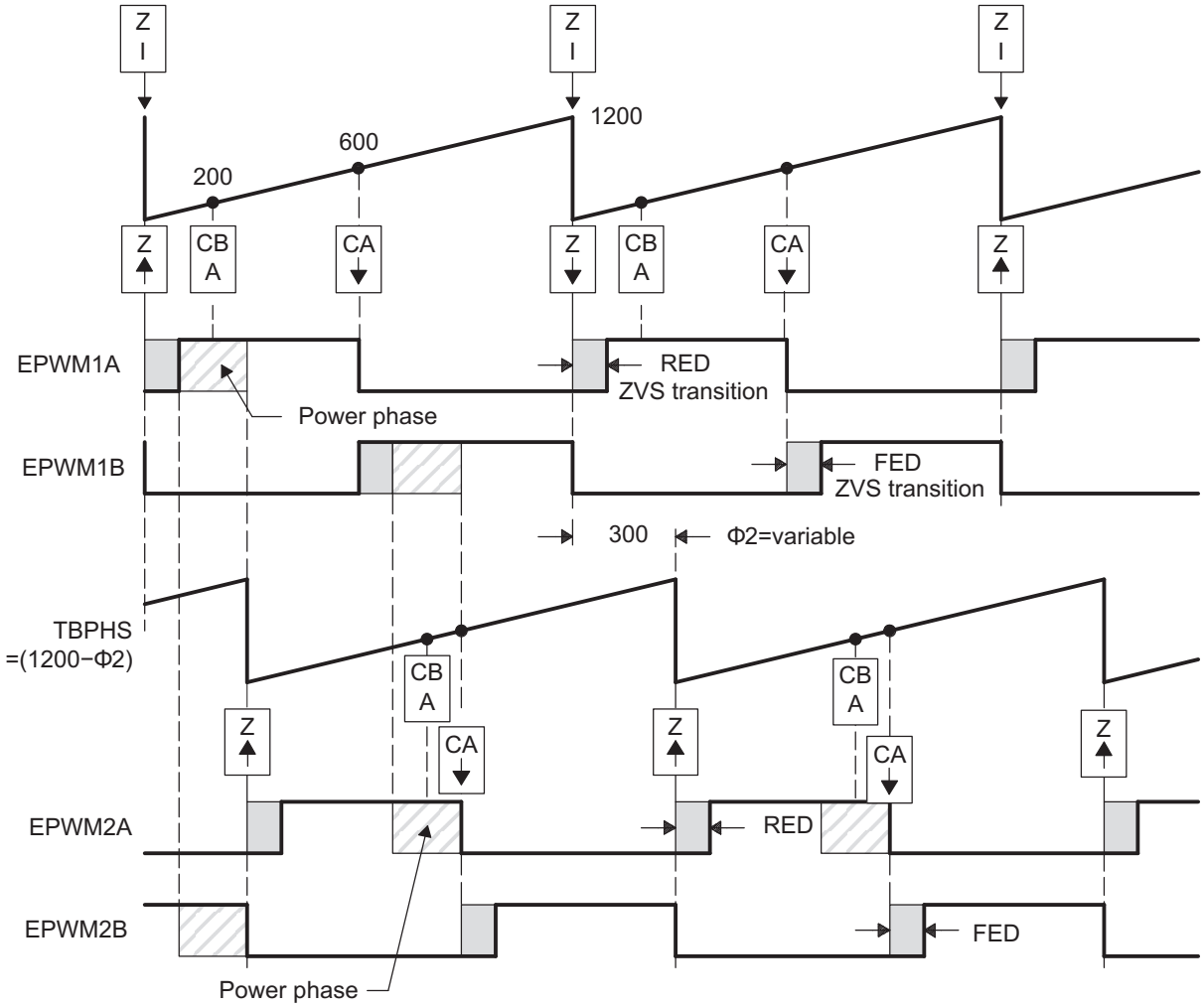
The example given in Figure 16-63 assumes a static or constant phase relationship between legs (modules). In such a case, control is achieved by modulating the duty cycle. It is also possible to dynamically change the phase value on a cycle-by-cycle basis. This feature lends itself to controlling a class 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 16-64 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 16-63. Controlling a Full-H Bridge Stage (F_{PWM2} = F_{PWM1})



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 16-64. ZVS Full-H Bridge Waveforms



Copyright © 2013–2016, Texas Instruments Incorporated

Table 16-48. EPWM1 Initialization for Figure 16-63

Register	Bit	Value	Comments
TBPRD	TBPRD	1200 (4B0h)	Period = 1201 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCTL	CTRMODE	TB_UP	
	PHSEN	TB_DISABLE	Phase loading disabled
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_CTR_ZERO	Sync down-stream module
CMPA	CMPA	600 (258h)	Set 50% duty for EPWM1A
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	ZRO	AQ_SET	Set actions for EPWM1A
	CAU	AQ_CLEAR	
DBCTL	MODE	DB_FULL_ENABLE	Enable Dead-band module
	POLSEL	DB_ACTV_HIC	Active Hi complementary
DBFED	DBFED	50	FED = 50 TBCLKs
	DBRED	70	RED = 70 TBCLKs

Table 16-49. EPWM2 Initialization for Figure 16-63

Register	Bit	Value	Comments
TBPRD	TBPRD	1200 (4B0h)	Period = 1201 TBCLK counts
TBPHS	TBPHS	0	Clear Phase Register to 0
TBCTL	CTRMODE	TB_UP	
	PHSEN	TB_ENABLE	Slave module
	PRDLD	TB_SHADOW	
	SYNCOSEL	TB_SYNC_IN	Sync flow-through
CMPA	CMPA	600 (258h)	Set 50% duty for EPWM2A
CMPCTL	SHDWAMODE	CC_SHADOW	
	SHDWBMODE	CC_SHADOW	
	LOADAMODE	CC_CTR_ZERO	Load on CTR = 0
	LOADBMODE	CC_CTR_ZERO	Load on CTR = 0
AQCTLA	ZRO	AQ_SET	Set actions for EPWM2A
	CAU	AQ_CLEAR	
DBCTL	MODE	DB_FULL_ENABLE	Enable Dead-band module
	POLSEL	DB_ACTV_HIC	Active Hi complementary
DBFED	DBFED	30	FED = 30 TBCLKs
	DBRED	40	RED = 40 TBCLKs

Example 16-8. Code Snippet for Configuration in Figure 16-63

```
// Run Time (Note: Example execution of one run-time instance)
//=========================================================
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
```


Copyright © 2013–2016, Texas Instruments Incorporated

16.4 Registers

This section includes the registers for the submodules.

Table 16-50. Submodule Registers

Submodule	Section
Time-Base Submodule Registers	Section 16.4.1
Counter-Compare Submodule Registers	Section 16.4.2
Action-Qualifier Submodule Registers	Section 16.4.3
Dead-Band Generator Submodule Registers	Section 16.4.4
PWM-Chopper Submodule Registers	Section 16.4.5
Trip-Zone Submodule Registers	Section 16.4.6
Event-Trigger Submodule Registers	Section 16.4.7
High-Resolution PWM Registers	Section 16.4.8

16.4.1 Time-Base Submodule Registers

Table 16-51 lists the memory-mapped registers for the time-base submodule. See your device-specific data manual for the memory address of these registers. All other register offset addresses not listed in Table 16-51 should be considered as reserved locations and the register contents should not be modified.

Table 16-51. Time-Base Submodule Registers

Offset	Acronym	Register Description	Section
0h	TBCTL	Time-Base Control Register	Section 16.4.1.1
2h	TBSTS	Time-Base Status Register	Section 16.4.1.2
4h	TBPHSHR	Time-Base Phase High-Resolution Register[(1)]	Section 16.4.8.1
6h	TBPHS	Time-Base Phase Register	Section 16.4.1.3
8h	TBCNT	Time-Base Counter Register	Section 16.4.1.4
Ah	TBPRD	Time-Base Period Register	Section 16.4.1.5

(1) This register is only available on ePWM instances that include the high-resolution PWM (HRPWM) extension; otherwise, this location is reserved. See your device-specific data manual to determine which instances include the HRPWM.

16.4.1.1 Time-Base Control Register (TBCTL)

The time-base control register (TBCTL) is shown in Figure 16-65 and described in Table 16-52.

Figure 16-65. 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	

7	6	5	4	3	2	1	0
HSPCLKDIV	SWFSYNC	SYNCOSEL		PRDLD	PHSEN	CTRMODE	
R/W-1	R/W-0	R/W-0		R/W-0	R/W-0	R/W-3h	

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

Copyright © 2013–2016, Texas Instruments Incorporated

Table 16-52. Time-Base Control Register (TBCTL) Field Descriptions

Bit	Field	Value	Description
15-14	FREE, SOFT	0-3h	Emulation Mode Bits. These bits select the behavior of the ePWM time-base counter during emulation events:
		0	Stop after the next time-base counter increment or decrement
		1h	Stop when counter completes a whole cycle: • Up-count mode: stop when the time-base counter = period (TBCNT = TBPRD) • Down-count mode: stop when the time-base counter = 0000 (TBCNT = 0000h) • Up-down-count mode: stop when the time-base counter = 0000 (TBCNT = 0000h)
		2h-3h	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 (TBCNT) 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	0-7h	Time-base Clock Prescale Bits. These bits determine part of the time-base clock prescale value. TBCLK = SYSCLKOUT/(HSPCLKDIV × CLKDIV)
		0	/1 (default on reset)
		1h	/2
		2h	/4
		3h	/8
		4h	/16
		5h	/32
		6h	/64
		7h	/128
9-7	HSPCLKDIV	0-7h	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.
		0	/1
		1h	/2 (default on reset)
		2h	/4
		3h	/6
		4h	/8
		5h	/10
		6h	/12
		7h	/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	0-3h	Synchronization Output Select. These bits select the source of the EPWMxSYNCO signal.
		0	EPWMxSYNC:
		1h	CTR = 0: Time-base counter equal to zero (TBCNT = 0000h)
		2h	CTR = CMPB : Time-base counter equal to counter-compare B (TBCNT = CMPB)
		3h	Disable EPWMxSYNCO signal

Copyright © 2013–2016, Texas Instruments Incorporated

Table 16-52. 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, TBCNT, is equal to zero.
			A write or read to the TBPRD register accesses the shadow register.
		1	Load the TBPRD register immediately without using a shadow register.
			A write or read to the TBPRD register directly accesses the active register.
2	PHSEN		Counter Register Load From Phase Register Enable
		0	Do not load the time-base counter (TBCNT) from the time-base phase register (TBPHS)
		1	Load the time-base counter with the phase register when an EPWMxSYNCI input signal occurs or when a software synchronization is forced by the SWFSYNC bit.
1-0	CTRMODE	0-3h	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:
		0	Up-count mode
		1h	Down-count mode
		2h	Up-down-count mode
		3h	Stop-freeze counter operation (default on reset)

16.4.1.2 Time-Base Status Register (TBSTS)

The time-base status register (TBSTS) is shown in Figure 16-66 and described in Table 16-53.

Figure 16-66. Time-Base Status Register (TBSTS)

15	3	2	1	0
Reserved		CTRMAX	SYNCI	CTRDIR
R-0		R/W1C-0	R/W1C-0	R-1

LEGEND: R/W = Read/Write; R/W1C = Read/Write 1 to clear; -*n* = value after reset

Table 16-53. Time-Base Status Register (TBSTS) Field Descriptions

Bit	Field	Value	Description
15-3	Reserved	0	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.

16.4.1.3 Time-Base Phase Register (TBPHS)

The time-base phase register (TBPHS) is shown in Figure 16-67 and described in Table 16-54.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 16-67. Time-Base Phase Register (TBPHS)

15	0
TBPHS	
R/W-0	

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

Table 16-54. Time-Base Phase Register (TBPHS) Field Descriptions

Bits	Name	Value	Description
15-0	TBPHS	0-FFFFh	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 (TBCNT) 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.

16.4.1.4 Time-Base Counter Register (TBCNT)

The time-base counter register (TBCNT) is shown in Figure 16-68 and described in Table 16-55.

Figure 16-68. Time-Base Counter Register (TBCNT)

15	0
TBCNT	
R/W-0	

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

Table 16-55. Time-Base Counter Register (TBCNT) Field Descriptions

Bits	Name	Value	Description
15-0	TBCNT	0-FFFFh	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.

Copyright © 2013–2016, Texas Instruments Incorporated

16.4.1.5 Time-Base Period Register (TBPRD)

The time-base period register (TBPRD) is shown in Figure 16-69 and described in Table 16-56.

Figure 16-69. Time-Base Period Register (TBPRD)

15	0
TBPRD	
R/W-0	

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

Table 16-56. Time-Base Period Register (TBPRD) Field Descriptions

Bits	Name	Value	Description
15-0	TBPRD	0-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.

16.4.2 Counter-Compare Submodule Registers

Table 16-57 lists the memory-mapped registers for the counter-compare submodule. See your device-specific data manual for the memory address of these registers. All other register offset addresses not listed in Table 16-57 should be considered as reserved locations and the register contents should not be modified.

Table 16-57. Counter-Compare Submodule Registers

Offset	Acronym	Register Description	Section
Eh	CMPCTL	Counter-Compare Control Register	Section 16.4.2.1
10h	CMPAHR	Counter-Compare A High-Resolution Register(1)	Section 16.4.8.2
12h	CMPA	Counter-Compare A Register	Section 16.4.2.2
14h	CMPB	Counter-Compare B Register	Section 16.4.2.3

(1) This register is only available on ePWM instances that include the high-resolution PWM (HRPWM) extension; otherwise, this location is reserved. See your device-specific data manual to determine which instances include the HRPWM.

Copyright © 2013–2016, Texas Instruments Incorporated

16.4.2.1 Counter-Compare Control Register (CMPCTL)

The counter-compare control register (CMPCTL) is shown in Figure 16-70 and described in Table 16-58.

Figure 16-70. Counter-Compare Control Register (CMPCTL)

15					10	9	8
Reserved						SHDWBFULL	SHDWAFULL
R-0						R-0	R-0
7	6	5	4	3	2	1	0
Reserved	SHDWBMODE	Reserved	SHDWAMODE	LOADBMODE		LOADAMODE	
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 16-58. Counter-Compare Control Register (CMPCTL) Field Descriptions

Bits	Name	Value	Description
15-10	Reserved	0	Reserved
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
3-2	LOADBMODE	0-3h	Active Counter-Compare B (CMPB) Load From Shadow Select Mode. This bit has no effect in immediate mode (CMPCTL[SHDWBMODE] = 1).
		0	Load on CTR = 0: Time-base counter equal to zero (TBCNT = 0000h)
		1h	Load on CTR = PRD: Time-base counter equal to period (TBCNT = TBPRD)
		2h	Load on either CTR = 0 or CTR = PRD
		3h	Freeze (no loads possible)
1-0	LOADAMODE	0-3h	Active Counter-Compare A (CMPA) Load From Shadow Select Mode. This bit has no effect in immediate mode (CMPCTL[SHDWAMODE] = 1).
		0	Load on CTR = 0: Time-base counter equal to zero (TBCNT = 0000h)
		1h	Load on CTR = PRD: Time-base counter equal to period (TBCNT = TBPRD)
		2h	Load on either CTR = 0 or CTR = PRD
		3h	Freeze (no loads possible)

Copyright © 2013–2016, Texas Instruments Incorporated

16.4.2.2 Counter-Compare A Register (CMPA)

The counter-compare A register (CMPA) is shown in Figure 16-71 and described in Table 16-59.

Figure 16-71. Counter-Compare A Register (CMPA)

15	0
CMPA	
R/W-0	

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

Table 16-59. Counter-Compare A Register (CMPA) Field Descriptions

Bits	Name	Value	Description
15-0	CMPA	0-FFFFh	The value in the active CMPA register is continuously compared to the time-base counter (TBCNT). 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.

Copyright © 2013–2016, Texas Instruments Incorporated

16.4.2.3 Counter-Compare B Register (CMPB)

The counter-compare B register (CMPB) is shown in Figure 16-72 and described in Table 16-60.

Figure 16-72. Counter-Compare B Register (CMPB)

15	0
CMPB	
R/W-0	

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

Table 16-60. Counter-Compare B Register (CMPB) Field Descriptions

Bits	Name	Value	Description
15-0	CMPB	0-FFFFh	The value in the active CMPB register is continuously compared to the time-base counter (TBCNT). 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.

16.4.3 Action-Qualifier Submodule Registers

Table 16-61 lists the memory-mapped registers for the action-qualifier submodule. See your device-specific data manual for the memory address of these registers. All other register offset addresses not listed in Table 16-61 should be considered as reserved locations and the register contents should not be modified.

Table 16-61. Action-Qualifier Submodule Registers

Offset	Acronym	Register Description	Section
16h	AQCTLA	Action-Qualifier Output A Control Register	Section 16.4.3.1
18h	AQCTLB	Action-Qualifier Output B Control Register	Section 16.4.3.2
1Ah	AQSFRC	Action-Qualifier Software Force Register	Section 16.4.3.3
1Ch	AQCSFRC	Action-Qualifier Continuous Software Force Register	Section 16.4.3.4

Copyright © 2013–2016, Texas Instruments Incorporated

16.4.3.1 Action-Qualifier Output A Control Register (AQCTLA)

The action-qualifier output A control register (AQCTLA) is shown in Figure 16-73 and described in Table 16-62.

Figure 16-73. Action-Qualifier Output A Control Register (AQCTLA)

15 12	11 10	9 8	7 6	5 4	3 2	1 0
Reserved	CBD	CBU	CAD	CAU	PRD	ZRO
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 16-62. Action-Qualifier Output A Control Register (AQCTLA) Field Descriptions

Bits	Name	Value	Description
15-12	Reserved	0	Reserved
11-10	CBD	0-3h	Action when the time-base counter equals the active CMPB register and the counter is decrementing.
		0	Do nothing (action disabled)
		1h	Clear: force EPWMxA output low.
		2h	Set: force EPWMxA output high.
		3h	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.
9-8	CBU	0-3h	Action when the counter equals the active CMPB register and the counter is incrementing.
		0	Do nothing (action disabled)
		1h	Clear: force EPWMxA output low.
		2h	Set: force EPWMxA output high.
		3h	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.
7-6	CAD	0-3h	Action when the counter equals the active CMPA register and the counter is decrementing.
		0	Do nothing (action disabled)
		1h	Clear: force EPWMxA output low.
		2h	Set: force EPWMxA output high.
		3h	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.
5-4	CAU	0-3h	Action when the counter equals the active CMPA register and the counter is incrementing.
		0	Do nothing (action disabled)
		1h	Clear: force EPWMxA output low.
		2h	Set: force EPWMxA output high.
		3h	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.
3-2	PRD	0-3h	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.
		0	Do nothing (action disabled)
		1h	Clear: force EPWMxA output low.
		2h	Set: force EPWMxA output high.
		3h	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.
1-0	ZRO	0-3h	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.
		0	Do nothing (action disabled)
		1h	Clear: force EPWMxA output low.
		2h	Set: force EPWMxA output high.
		3h	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.

Copyright © 2013–2016, Texas Instruments Incorporated

16.4.3.2 Action-Qualifier Output B Control Register (AQCTLB)

The action-qualifier output B control register (AQCTLB) is shown in Figure 16-74 and described in Table 16-63.

Figure 16-74. Action-Qualifier Output B Control Register (AQCTLB)

15 12	11 10	9 8	7 6	5 4	3 2	1 0
Reserved	CBD	CBU	CAD	CAU	PRD	ZRO
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 16-63. Action-Qualifier Output B Control Register (AQCTLB) Field Descriptions

Bits	Name	Value	Description
15-12	Reserved	0	Reserved
11-10	CBD	0-3h	Action when the counter equals the active CMPB register and the counter is decrementing.
		0	Do nothing (action disabled)
		1h	Clear: force EPWMxB output low.
		2h	Set: force EPWMxB output high.
		3h	Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.
9-8	CBU	0-3h	Action when the counter equals the active CMPB register and the counter is incrementing.
		0	Do nothing (action disabled)
		1h	Clear: force EPWMxB output low.
		2h	Set: force EPWMxB output high.
		3h	Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.
7-6	CAD	0-3h	Action when the counter equals the active CMPA register and the counter is decrementing.
		0	Do nothing (action disabled)
		1h	Clear: force EPWMxB output low.
		2h	Set: force EPWMxB output high.
		3h	Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.
5-4	CAU	0-3h	Action when the counter equals the active CMPA register and the counter is incrementing.
		0	Do nothing (action disabled)
		1h	Clear: force EPWMxB output low.
		2h	Set: force EPWMxB output high.
		3h	Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.
3-2	PRD	0-3h	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.
		0	Do nothing (action disabled)
		1h	Clear: force EPWMxB output low.
		2h	Set: force EPWMxB output high.
		3h	Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.
1-0	ZRO	0-3h	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.
		0	Do nothing (action disabled)
		1h	Clear: force EPWMxB output low.
		2h	Set: force EPWMxB output high.
		3h	Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.

Copyright © 2013–2016, Texas Instruments Incorporated

16.4.3.3 Action-Qualifier Software Force Register (AQSFRC)

The action-qualifier software force register (AQSFRC) is shown in Figure 16-75 and described in Table 16-64.

Figure 16-75. Action-Qualifier Software Force Register (AQSFRC)

15 8	7 6	5	4 3	2	1 0
Reserved	RLDCSF	OTSFB	ACTSFB	OTSFA	ACTSFA
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 16-64. Action-Qualifier Software Force Register (AQSFRC) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved
7-6	RLDCSF	0-3h	AQCSFRC Active Register Reload From Shadow Options
		0	Load on event counter equals zero
		1h	Load on event counter equals period
		2h	Load on event counter equals zero or counter equals period
		3h	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, that is, 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	0-3h	Action when One-Time Software Force B Is invoked
		0	Does nothing (action disabled)
		1h	Clear (low)
		2h	Set (high)
		3h	Toggle (Low -> High, High -> Low)
			Note: This action is not qualified by counter direction (CNT_dir)
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 (that is, a forced event is initiated).
		1	Initiates a single software forced event
1-0	ACTSFA	0-3h	Action When One-Time Software Force A Is Invoked
		0	Does nothing (action disabled)
		1h	Clear (low)
		2h	Set (high)
		3h	Toggle (Low → High, High → Low)
			Note: This action is not qualified by counter direction (CNT_dir)

Copyright © 2013–2016, Texas Instruments Incorporated

16.4.3.4 Action-Qualifier Continuous Software Force Register (AQCSFRC)

The action-qualifier continuous software force register (AQCSFRC) is shown in Figure 16-76 and described in Table 16-65.

Figure 16-76. Action-Qualifier Continuous Software Force Register (AQCSFRC)

15	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 16-65. Action-Qualifier Continuous Software Force Register (AQCSFRC) Field Descriptions

Bits	Name	Value	Description
15-4	Reserved	0	Reserved
3-2	CSFB	0-3h	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].
		0	Forcing disabled, that is, has no effect
		1h	Forces a continuous low on output B
		2h	Forces a continuous high on output B
		3h	Software forcing is disabled and has no effect
1-0	CSFA	0-3h	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.
		0	Forcing disabled, that is, has no effect
		1h	Forces a continuous low on output A
		2h	Forces a continuous high on output A
		3h	Software forcing is disabled and has no effect

16.4.4 Dead-Band Generator Submodule Registers

Table 16-66 lists the memory-mapped registers for the dead-band generator submodule. See your device-specific data manual for the memory address of these registers. All other register offset addresses not listed in Table 16-66 should be considered as reserved locations and the register contents should not be modified.

Table 16-66. Dead-Band Generator Submodule Registers

Offset	Acronym	Register Description	Section
1Eh	DBCTL	Dead-Band Generator Control Register	Section 16.4.4.1
20h	DBRED	Dead-Band Generator Rising Edge Delay Register	Section 16.4.4.2
22h	DBFED	Dead-Band Generator Falling Edge Delay Register	Section 16.4.4.3

Copyright © 2013–2016, Texas Instruments Incorporated

16.4.4.1 Dead-Band Generator Control Register (DBCTL)

The dead-band generator control register (DBCTL) is shown in Figure 16-77 and described in Table 16-67.

Figure 16-77. Dead-Band Generator Control Register (DBCTL)

15 6	5 4	3 2	1 0
Reserved	IN_MODE	POLSEL	OUT_MODE
R-0	R/W-0	R/W-0	R/W-0

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

Table 16-67. Dead-Band Generator Control Register (DBCTL) Field Descriptions

Bits	Name	Value	Description
15-6	Reserved	0	Reserved
5-4	IN_MODE	0-3h	Dead Band Input Mode Control. Bit 5 controls the S5 switch and bit 4 controls the S4 switch shown in Figure 16-29. 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.
		0	EPWMxA In (from the action-qualifier) is the source for both falling-edge and rising-edge delay.
		1h	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.
		2h	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.
		3h	EPWMxB In (from the action-qualifier) is the source for both rising-edge delay and falling-edge delayed signal.
3-2	POLSEL	0-3h	Polarity Select Control. Bit 3 controls the S3 switch and bit 2 controls the S2 switch shown in Figure 16-29. 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.
		0	Active high (AH) mode. Neither EPWMxA nor EPWMxB is inverted (default).
		1h	Active low complementary (ALC) mode. EPWMxA is inverted.
		2h	Active high complementary (AHC). EPWMxB is inverted.
		3h	Active low (AL) mode. Both EPWMxA and EPWMxB are inverted.
1-0	OUT_MODE	0-3h	Dead-band Output Mode Control. Bit 1 controls the S1 switch and bit 0 controls the S0 switch shown in Figure 16-29. This allows you to selectively enable or bypass the dead-band generation for the falling-edge and rising-edge delay.
		0	Dead-band generation is bypassed for both output signals. In this mode, both the EPWMxA and EPWMxB output signals from the action-qualifier are passed directly to the PWM-chopper submodule. In this mode, the POLSEL and IN_MODE bits have no effect.
		1h	Disable rising-edge delay. The EPWMxA signal from the action-qualifier is passed straight through to the EPWMxA input of the PWM-chopper submodule. The falling-edge delayed signal is seen on output EPWMxB. The input signal for the delay is determined by DBCTL[IN_MODE].
		2h	Disable falling-edge delay. The EPWMxB signal from the action-qualifier is passed straight through to the EPWMxB input of the PWM-chopper submodule. The rising-edge delayed signal is seen on output EPWMxA. The input signal for the delay is determined by DBCTL[IN_MODE].
		3h	Dead-band is fully enabled for both rising-edge delay on output EPWMxA and falling-edge delay on output EPWMxB. The input signal for the delay is determined by DBCTL[IN_MODE].

Copyright © 2013–2016, Texas Instruments Incorporated

16.4.4.2 Dead-Band Generator Rising Edge Delay Register (DBRED)

The dead-band generator rising edge delay register (DBRED) is shown in Figure 16-78 and described in Table 16-68.

Figure 16-78. Dead-Band Generator Rising Edge Delay Register (DBRED)

15	10	9	0
Reserved		DEL	
R-0		R/W-0	

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

Table 16-68. Dead-Band Generator Rising Edge Delay Register (DBRED) Field Descriptions

Bits	Name	Value	Description
15-10	Reserved	0	Reserved
9-0	DEL	0-3FFh	Rising Edge Delay Count. 10-bit counter.

16.4.4.3 Dead-Band Generator Falling Edge Delay Register (DBFED)

The dead-band generator falling edge delay register (DBFED) is shown in Figure 16-79 and described in Table 16-69.

Figure 16-79. Dead-Band Generator Falling Edge Delay Register (DBFED)

15	10	9	0
Reserved		DEL	
R-0		R/W-0	

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

Table 16-69. Dead-Band Generator Falling Edge Delay Register (DBFED) Field Descriptions

Bits	Name	Value	Description
15-10	Reserved	0	Reserved
9-0	DEL	0-3FFh	Falling Edge Delay Count. 10-bit counter

Copyright © 2013–2016, Texas Instruments Incorporated

16.4.5 PWM-Chopper Submodule Register

The PWM-chopper control register (PCCTL) is shown in Figure 16-80 and described in Table 16-70.

Figure 16-80. PWM-Chopper Control Register (PCCTL)

15 11	10 8	7 5	4 1	0
Reserved	CHPDUTY	CHPFREQ	OSHTWTH	CHPEN
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 16-70. PWM-Chopper Control Register (PCCTL) Bit Descriptions

Bits	Name	Value	Description
15-11	Reserved	0	Reserved
10-8	CHPDUTY	0-7h	Chopping Clock Duty Cycle
		0	Duty = 1/8 (12.5%)
		1h	Duty = 2/8 (25.0%)
		2h	Duty = 3/8 (37.5%)
		3h	Duty = 4/8 (50.0%)
		4h	Duty = 5/8 (62.5%)
		5h	Duty = 6/8 (75.0%)
		6h	Duty = 7/8 (87.5%)
		7h	Reserved
7-5	CHPFREQ	0-7h	Chopping Clock Frequency
		0	Divide by 1 (no prescale)
		1h	Divide by 2
		2h	Divide by 3
		3h-7h	Divide by 4 to divide by 8
4-1	OSHTWTH	0-Fh	One-Shot Pulse Width
		0	1 × SYSCLKOUT/8 wide
		1h	2 × SYSCLKOUT/8 wide
		2h	3 × SYSCLKOUT/8 wide
		3h-Fh	4 × SYSCLKOUT/8 wide to 16 × SYSCLKOUT/8 wide
0	CHPEN		PWM-chopping Enable
		0	Disable (bypass) PWM chopping function
		1	Enable chopping function

Copyright © 2013–2016, Texas Instruments Incorporated

16.4.6 Trip-Zone Submodule Registers

Table 16-71 lists the memory-mapped registers for the trip-zone submodule. See your device-specific data manual for the memory address of these registers. All other register offset addresses not listed in Table 16-71 should be considered as reserved locations and the register contents should not be modified.

Table 16-71. Trip-Zone Submodule Registers

Offset	Acronym	Register Description	Section
24h	TZSEL	Trip-Zone Select Register	Section 16.4.6.1
28h	TZCTL	Trip-Zone Control Register	Section 16.4.6.2
2Ah	TZEINT	Trip-Zone Enable Interrupt Register	Section 16.4.6.3
2Ch	TZFLG	Trip-Zone Flag Register	Section 16.4.6.4
2Eh	TZCLR	Trip-Zone Clear Register	Section 16.4.6.5
30h	TZFRC	Trip-Zone Force Register	Section 16.4.6.6

16.4.6.1 Trip-Zone Select Register (TZSEL)

The trip-zone select register (TZSEL) is shown in Figure 16-81 and described in Table 16-72.

Figure 16-81. Trip-Zone Select Register (TZSEL)

15 9	8	7 1	0
Reserved/OSHT*n* (1)	OSHT1	Reserved/CBC*n* (1)	CBC1
R/W-0	R/W-0	R/W-0	R/W-0

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

(1) Number of register bits depends on how many trip-zone pins are available in the device. See your device-specific data manual.

Table 16-72. Trip-Zone Submodule Select Register (TZSEL) Field Descriptions

Bits	Name	Value	Description
15-8	OSHT*n*		Trip-zone *n* ($\overline{TZn}$) select. 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 (Section 16.4.6.2) is taken on the EPWMxA and EPWMxB outputs. The one-shot trip condition remains latched until you clear the condition via the TZCLR register (Section 16.4.6.5).
		0	Disable $\overline{TZn}$ as a one-shot trip source for this ePWM module.
		1	Enable $\overline{TZn}$ as a one-shot trip source for this ePWM module.
7-0	CBC*n*		Trip-zone *n* ($\overline{TZn}$) select. 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 (Section 16.4.6.2) is taken on the EPWMxA and EPWMxB outputs. A cycle-by-cycle trip condition is automatically cleared when the time-base counter reaches zero.
		0	Disable $\overline{TZn}$ as a CBC trip source for this ePWM module.
		1	Enable $\overline{TZn}$ as a CBC trip source for this ePWM module.

Copyright © 2013–2016, Texas Instruments Incorporated

16.4.6.2 Trip-Zone Control Register (TZCTL)

The trip-zone control register (TZCTL) is shown in Figure 16-82 and described in Table 16-73.

Figure 16-82. Trip-Zone Control Register (TZCTL)

15	4	3	2	1	0
Reserved		TZB		TZA	
R-0		R/W-0		R/W-0	

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

Table 16-73. Trip-Zone Control Register (TZCTL) Field Descriptions

Bits	Name	Value	Description
15–4	Reserved	0	Reserved
3–2	TZB	0-3h	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 (Section 16.4.6.1).
		0	High impedance (EPWMxB = High-impedance state)
		1h	Force EPWMxB to a high state
		2h	Force EPWMxB to a low state
		3h	Do nothing, no action is taken on EPWMxB.
1–0	TZA	0-3h	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 (Section 16.4.6.1).
		0	High impedance (EPWMxA = High-impedance state)
		1h	Force EPWMxA to a high state
		2h	Force EPWMxA to a low state
		3h	Do nothing, no action is taken on EPWMxA.

16.4.6.3 Trip-Zone Enable Interrupt Register (TZEINT)

The trip-zone enable interrupt register (TZEINT) is shown in Figure 16-83 and described in Table 16-74.

Figure 16-83. Trip-Zone Enable Interrupt Register (TZEINT)

15	3	2	1	0
Reserved		OST	CBC	Rsvd
R-0		R/W-0	R/W-0	R-0

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

Table 16-74. Trip-Zone Enable Interrupt Register (TZEINT) Field Descriptions

Bits	Name	Value	Description
15-3	Reserved	0	Reserved
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 EPWMxTZINT 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 EPWMxTZINT interrupt.
0	Reserved	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

16.4.6.4 Trip-Zone Flag Register (TZFLG)

The trip-zone flag register (TZFLG) is shown in Figure 16-84 and described in Table 16-75.

Figure 16-84. Trip-Zone Flag Register (TZFLG)

15	3	2	1	0
Reserved		OST	CBC	INT
R-0		R-0	R-0	R-0

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

Table 16-75. Trip-Zone Flag Register (TZFLG) Field Descriptions

Bits	Name	Value	Description
15-3	Reserved	0	Reserved
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 (Section 16.4.6.5).
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 pin 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 pins is automatically cleared when the ePWM time-base counter reaches zero (TBCNT = 0000h) if the trip condition is no longer present. The condition on the pins is only cleared when the TBCNT = 0000h no matter where in the cycle the CBC flag is cleared. This bit is cleared by writing the appropriate value to the TZCLR register (Section 16.4.6.5).
0	INT		Latched Trip Interrupt Status Flag
		0	Indicates no interrupt has been generated.
		1	Indicates an EPWMxTZINT interrupt was generated because of a trip condition. No further EPWMxTZINT 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 (Section 16.4.6.5).

Copyright © 2013–2016, Texas Instruments Incorporated

16.4.6.5 Trip-Zone Clear Register (TZCLR)

The trip-zone clear register (TZCLR) is shown in Figure 16-85 and described in Table 16-76.

Figure 16-85. Trip-Zone Clear Register (TZCLR)

15	3	2	1	0
Reserved		OST	CBC	INT
R-0		R/W-0	R/W-0	R/W-0

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

Table 16-76. Trip-Zone Clear Register (TZCLR) Field Descriptions

Bits	Name	Value	Description
15-3	Reserved	0	Reserved
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 EPWMxTZINT 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.

16.4.6.6 Trip-Zone Force Register (TZFRC)

The trip-zone force register (TZFRC) is shown in Figure 16-86 and described in Table 16-77.

Figure 16-86. Trip-Zone Force Register (TZFRC)

15	3	2	1	0
Reserved		OST	CBC	Rsvd
R-0		R/W-0	R/W-0	R-0

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

Table 16-77. Trip-Zone Force Register (TZFRC) Field Descriptions

Bits	Name	Value	Description
15-3	Reserved	0	Reserved
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	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

16.4.7 Event-Trigger Submodule Registers

Table 16-78 lists the memory-mapped registers for the event-trigger submodule. See your device-specific data manual for the memory address of these registers. All other register offset addresses not listed in Table 16-78 should be considered as reserved locations and the register contents should not be modified.

Table 16-78. Event-Trigger Submodule Registers

Offset	Acronym	Register Description	Section
32h	ETSEL	Event-Trigger Selection Register	Section 16.4.7.1
34h	ETPS	Event-Trigger Prescale Register	Section 16.4.7.2
36h	ETFLG	Event-Trigger Flag Register	Section 16.4.7.3
38h	ETCLR	Event-Trigger Clear Register	Section 16.4.7.4
3Ah	ETFRC	Event-Trigger Force Register	Section 16.4.7.5

16.4.7.1 Event-Trigger Selection Register (ETSEL)

The event-trigger selection register (ETSEL) is shown in Figure 16-87 and described in Table 16-79.

Figure 16-87. Event-Trigger Selection Register (ETSEL)

15	4	3	2	0
Reserved		INTEN	INTSEL	
R-0		R/W-0	R/W-0	

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

Table 16-79. Event-Trigger Selection Register (ETSEL) Field Descriptions

Bits	Name	Value	Description
15-4	Reserved	0	Reserved
3	INTEN		Enable ePWM Interrupt (EPWMx_INT) Generation
		0	Disable EPWMx_INT generation
		1	Enable EPWMx_INT generation
2-0	INTSEL	0-7h	ePWM Interrupt (EPWMx_INT) Selection Options
		0	Reserved
		1h	Enable event time-base counter equal to zero. (TBCNT = 0000h)
		2h	Enable event time-base counter equal to period (TBCNT = TBPRD)
		3h	Reserved
		4h	Enable event time-base counter equal to CMPA when the timer is incrementing.
		5h	Enable event time-base counter equal to CMPA when the timer is decrementing.
		6h	Enable event: time-base counter equal to CMPB when the timer is incrementing.
		7h	Enable event: time-base counter equal to CMPB when the timer is decrementing.

Copyright © 2013–2016, Texas Instruments Incorporated

16.4.7.2 Event-Trigger Prescale Register (ETPS)

The event-trigger prescale register (ETPS) is shown in Figure 16-88 and described in Table 16-80.

Figure 16-88. Event-Trigger Prescale Register (ETPS)

15	4	3	2	1	0
Reserved		INTCNT		INTPRD	
R-0		R-0		R/W-0	

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

Table 16-80. Event-Trigger Prescale Register (ETPS) Field Descriptions

Bits	Name	Value	Description
15-4	Reserved	0	Reserved
3-2	INTCNT	0-3h	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].
		0	No events have occurred.
		1h	1 event has occurred.
		2h	2 events have occurred.
		3h	3 events have occurred.
1-0	INTPRD	0-3h	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 a INTPRD value that is the same as the current counter value will trigger an interrupt if it is enabled and the status flag is clear. Writing a INTPRD value that is less than the current counter value will result in an undefined state. If a counter event occurs at the same instant as a new zero or non-zero INTPRD value is written, the counter is incremented.
		0	Disable the interrupt event counter. No interrupt will be generated and ETFRC[INT] is ignored.
		1h	Generate an interrupt on the first event INTCNT = 01 (first event)
		2h	Generate interrupt on ETPS[INTCNT] = 1,0 (second event)
		3h	Generate interrupt on ETPS[INTCNT] = 1,1 (third event)

Copyright © 2013–2016, Texas Instruments Incorporated

16.4.7.3 Event-Trigger Flag Register (ETFLG)

The event-trigger flag register (ETFLG) is shown in Figure 16-89 and described in Table 16-81.

Figure 16-89. Event-Trigger Flag Register (ETFLG)

15	1	0
Reserved		INT
R-0		R-0

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

Table 16-81. Event-Trigger Flag Register (ETFLG) Field Descriptions

Bits	Name	Value	Description
15-1	Reserved	0	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 16-42.

16.4.7.4 Event-Trigger Clear Register (ETCLR)

The event-trigger clear register (ETCLR) is shown in Figure 16-90 and described in Table 16-82.

Figure 16-90. Event-Trigger Clear Register (ETCLR)

15	1	0
Reserved		INT
R-0		R-0

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

Table 16-82. Event-Trigger Clear Register (ETCLR) Field Descriptions

Bits	Name	Value	Description
15-1	Reserved	0	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.

Copyright © 2013–2016, Texas Instruments Incorporated

16.4.7.5 Event-Trigger Force Register (ETFRC)

The event-trigger force register (ETFRC) is shown in Figure 16-91 and described in Table 16-83.

Figure 16-91. Event-Trigger Force Register (ETFRC)

15	1	0
Reserved		INT
R-0		R-0

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

Table 16-83. Event-Trigger Force Register (ETFRC) Field Descriptions

Bits	Name	Value	Description
15-1	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.

16.4.8 High-Resolution PWM Submodule Registers

Table 16-84 lists the memory-mapped registers for the high-resolution PWM submodule. See your device-specific data manual for the memory address of these registers. All other register offset addresses not listed in Table 16-84 should be considered as reserved locations and the register contents should not be modified.

Table 16-84. High-Resolution PWM Submodule Registers

Offset	Acronym	Register Description	Section
4h	TBPHSHR	Time-Base Phase High-Resolution Register	Section 16.4.8.1
10h	CMPAHR	Counter-Compare A High-Resolution Register	Section 16.4.8.2
1040h	HRCNFG	HRPWM Configuration Register	Section 16.4.8.3

Copyright © 2013–2016, Texas Instruments Incorporated

16.4.8.1 Time-Base Phase High-Resolution Register (TBPHSHR)

The time-base phase high-resolution register (TBPHSHR) is shown in Figure 16-92 and described in Table 16-85.

Figure 16-92. Time-Base Phase High-Resolution Register (TBPHSHR)

15	8	7	0
TBPHSH		Reserved	
R/W-0		R/W-0	

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

Table 16-85. Time-Base Phase High-Resolution Register (TBPHSHR) Field Descriptions

Bit	Field	Value	Description
15-8	TBPHSH	0-FFh	Time-base phase high-resolution bits
7-0	Reserved	0	Reserved

16.4.8.2 Counter-Compare A High-Resolution Register (CMPAHR)

The counter-compare A high-resolution register (CMPAHR) is shown in Figure 16-93 and described in Table 16-86.

Figure 16-93. Counter-Compare A High-Resolution Register (CMPAHR)

15	8	7	0
CMPAHR		Reserved	
R/W-0		R/W-0	

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

Table 16-86. Counter-Compare A High-Resolution Register (CMPAHR) Field Descriptions

Bit	Field	Value	Description
15-8	CMPAHR	1-FFh	Compare A High-Resolution register bits for MEP step control. A minimum value of 1h is needed to enable HRPWM capabilities. Valid MEP range of operation 1-255h.
7-0	Reserved	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

16.4.8.3 HRPWM Configuration Register (HRCNFG)

The HRPWM configuration register (HRCNFG) is shown in Figure 16-94 and described in Table 16-87.

Figure 16-94. HRPWM Configuration Register (HRCNFG)

15 4	3	2	1 0
Reserved	HRLOAD	CTLMODE	EDGMODE
R-0	R/W-0	R/W-0	R/W-0

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

Table 16-87. HRPWM Configuration Register (HRCNFG) Field Descriptions

Bit	Field	Value	Description
15-4	Reserved	0	ReserveD
3	HRLOAD		Shadow mode bit: Selects the time event that loads the CMPAHR shadow value into the active register:
		0	CTR = 0 (counter equals zero)
		1	CTR = PRD (counter equal period)
			Note: Load mode selection is valid only if CTLMODE = 0 has been selected. You should select this event to match the selection of the CMPA load mode (CMPCTL[LOADMODE] bits) in the EPWM module as follows:
			0 Load on CTR = 0: Time-base counter equal to zero (TBCNT = 0000h)
			1h Load on CTR = PRD: Time-base counter equal to period (TBCNT = TBPRD)
			2h Load on either CTR = 0 or CTR = PRD (should not be used with HRPWM)
			3h Freeze (no loads possible – should not be used with HRPWM)
2	CTLMODE		Control Mode Bits: Selects the register (CMP or TBPHS) that controls the MEP:
		0	CMPAHR(8) Register controls the edge position (this is duty control mode). (default on reset)
		1	TBPHSHR(8) Register controls the edge position (this is phase control mode).
1-0	EDGMODE	0-3h	Edge Mode Bits: Selects the edge of the PWM that is controlled by the micro-edge position (MEP) logic:
		0	HRPWM capability is disabled (default on reset)
		1h	MEP control of rising edge
		2h	MEP control of falling edge
		3h	MEP control of both edges

Copyright © 2013–2016, Texas Instruments Incorporated

Enhanced Direct Memory Access (EDMA3) Controller

The enhanced direct memory access (EDMA3) controller is a high-performance, multichannel, multithreaded DMA controller that allows you to program a wide variety of transfer geometries and transfer sequences. This chapter describes the features and operations of the EDMA3 controller.

Section 17.1 provides a brief overview, features, and terminology. Section 17.2 provides the architecture details and common operations of the EDMA3 channel controllers (EDMA3_*m*_CC0) and the EDMA3 transfer controllers (EDMA3_*m*_TC*n*). Section 17.3 contains examples and common usage scenarios. Section 17.4 describes the memory-mapped registers associated with the EDMA3 controller.

Topic **Page**

17.1 Introduction **562**
17.2 Architecture **567**
17.3 Transfer Examples **609**
17.4 Registers **626**
17.5 Tips **693**
17.6 Setting Up a Transfer **695**

Copyright © 2013–2016, Texas Instruments Incorporated

17.1 Introduction

17.1.1 Overview

The enhanced direct memory access (EDMA3) controller's primary purpose is to service user-programmed data transfers between two memory-mapped slave endpoints on the device. Typical usage includes, but is not limited to:

- Servicing software driven paging transfers (for example, from external memory to internal device memory
- Servicing event driven peripherals, such as a serial port
- Performing sorting or subframe extraction of various data structures
- Offloading data transfers from the main device CPU(s) (See your device-specific data manual for specific peripherals that are accessible via EDMA3. See the section on SCR connectivity in your device-specific data manual for EDMA3 connectivity.)

The EDMA3 controller consists of two principal blocks:

- EDMA3 channel controller: (EDMA3_*m*_CC0)
- EDMA3 transfer controller: (EDMA3_*m*_TC*n*)

The EDMA3 channel controller serves as the user interface for the EDMA3 controller. The EDMA3CC includes parameter RAM (PaRAM), channel control registers, and interrupt control registers. The EDMA3CC serves to prioritize incoming software requests or events from peripherals, and submits transfer requests (TR) to the EDMA3 transfer controller.

The EDMA3 transfer controllers are responsible for data movement. The transfer request packets (TRP) submitted by the EDMA3CC contains the transfer context, based on which the transfer controller issues read/write commands to the source and destination addresses programmed for a given transfer.

17.1.2 Features

The EDMA3 channel controller (EDMA3CC) has the following features:

- Fully orthogonal transfer description
  - 3 transfer dimensions
  - A-synchronized transfers: 1 dimension serviced per event
  - AB-synchronized transfers: 2 dimensions serviced per event
  - Independent indexes on source and destination
  - Chaining feature allows 3-D transfer based on single event
- Flexible transfer definition
  - Increment or constant addressing modes
  - Linking mechanism allows automatic PaRAM set update. Useful for ping-pong type transfers, auto-reload transfers.
  - Chaining allows multiple transfers to execute with a single event
- Interrupt generation for:
  - Transfer completion
  - Error conditions (illegal addresses, illegal modes, exceeding queue threshold)
- Debug visibility
  - Queue watermarking
  - Error and status recording to facilitate debug
  - Missed event detection

Copyright © 2013–2016, Texas Instruments Incorporated

- EDMA3_0_CC0:
  - 32 DMA channels
  - 8 QDMA channels
  - 128 parameter RAM (PaRAM) entries
  - 2 event queues
  - 4 shadow regions
  - 2 transfer controllers (EDMA3_0_TC0 and EDMA3_0_TC1)
  - 5 interrupts:
    - EDMA3_0_CC0_INT0
    - EDMA3_0_CC0_INT1
    - EDMA3_0_CC0_INT2
    - EDMA3_0_CC0_INT3
    - EDMA3_0_CC0_ERRINT
- EDMA3_1_CC0:
  - 32 DMA channels
  - 8 QDMA channels
  - 128 parameter RAM (PaRAM) entries
  - 1 event queue
  - 4 shadow regions
  - 1 transfer controller (EDMA3_1_TC0)
  - 5 interrupts:
    - EDMA3_1_CC0_INT0
    - EDMA3_1_CC0_INT1
    - EDMA3_1_CC0_INT2
    - EDMA3_1_CC0_INT3
    - EDMA3_1_CC0_ERRINT

Copyright © 2013–2016, Texas Instruments Incorporated

The EDMA3 transfer controller (EDMA3TC) has the following features:

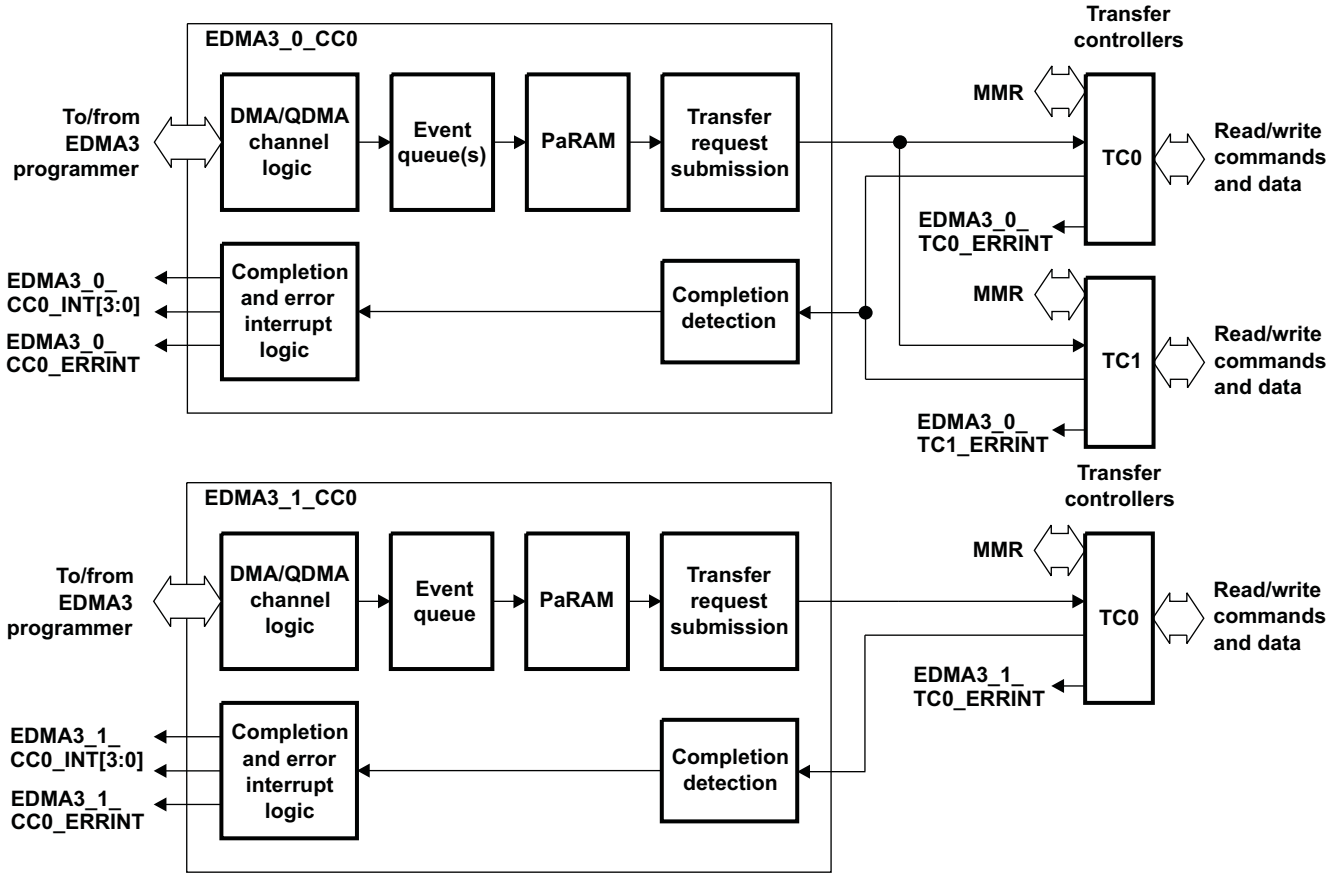
- Supports 2-dimensional transfers with independent indexes on source and destination (EDMA3CC manages the 3rd dimension)
- More then one transfer controller allows concurrent transfers
- Programmable priority level for each transfer controller relative to each other and other masters in the system.
- Support for increment or constant addressing mode transfers
- Error conditions with interrupt support
- Supports more then one in-flight transfer requests
- Debug/status visibility
- 64-bit wide read and write ports
- Little-endian mode
- EDMA3_0_TC0:
  - FIFIOSIZE = 128 bytes
  - BUSWIDTH (Read/Write Controllers) = 8 byte (64 bits)
  - DSTREGDEPTH = 4
  - DBS (default) = 16 bytes. The default burst size (DBS) is programmable, and can be configured for 16-, 32-, or 64-bytes burst size. See the Chip Configuration 0 Register (CFGCHIP0) in the *System Configuration (SYSCFG) Module* chapter for details to change the default burst size value.
  - Error interrupt: EDMA3_0_TC0_ERRINT
  - EDMA3 channel controller used: EDMA3_0_CC0
- EDMA3_0_TC1:
  - FIFIOSIZE = 128 bytes
  - BUSWIDTH (Read/Write Controllers) = 8 byte (64 bits)
  - DSTREGDEPTH = 4
  - DBS (default) = 16 bytes. The default burst size (DBS) is programmable, and can be configured for 16-, 32-, or 64-bytes burst size. See the Chip Configuration 0 Register (CFGCHIP0) in the *System Configuration (SYSCFG) Module* chapter for details to change the default burst size value.
  - Error interrupt: EDMA3_0_TC1_ERRINT
  - EDMA3 channel controller used: EDMA3_0_CC0
- EDMA3_1_TC0:
  - FIFIOSIZE = 256 bytes
  - BUSWIDTH (Read/Write Controllers) = 8 byte (64 bits)
  - DSTREGDEPTH = 4
  - DBS (default) = 16 bytes. The default burst size (DBS) is programmable, and can be configured for 16-, 32-, or 64-bytes burst size. See the Chip Configuration 1 Register (CFGCHIP1) in the *System Configuration (SYSCFG) Module* chapter for details to change the default burst size value.
  - Error interrupt: EDMA3_1_TC0_ERRINT
  - EDMA3 channel controller used: EDMA3_1_CC0

Copyright © 2013–2016, Texas Instruments Incorporated

17.1.3 Functional Block Diagram

Figure 17-1 shows a block diagram of the EDMA3 controller.

Figure 17-1. EDMA3 Controller Block Diagram



17.1.4 Terminology Used in This Document

The following are some terms used in this chapter.

Term	Meaning
A-synchronized transfer	A transfer type where 1 dimension is serviced per synchronization event.
AB-synchronized transfer	A transfer type where 2 dimensions are serviced per synchronization event.
Chaining	A trigger mechanism in which a transfer can be initiated at the completion of another transfer or subtransfer.
CPU(s)	The main processing engine or engines on a device.
DMA channel	A channel that can be triggered by external, manual, and chained events. All DMA channels exist in the EDMA3CC.
Dummy set or Dummy PaRAM set	A PaRAM set for which at least one of the count fields is equal to 0 and at least one of the count fields is nonzero. A null PaRAM set has all the count set fields cleared.
Dummy transfer	A dummy set results in the EDMA3CC performing a dummy transfer. This is not an error condition. A null set results in an error condition.

Copyright © 2013–2016, Texas Instruments Incorporated

Term	Meaning
EDMA3 channel controller (EDMA3CC)	The user-programmable portion of the EDMA3. The EDMA3CC contains the parameter RAM (PaRAM) , event processing logic, DMA/QDMA channels, event queues, etc. The EDMA3CC services events (external, manual, chained, QDMA) and is responsible for submitting transfer requests to the transfer controllers (EDMA3TC), which perform the actual transfer.
EDMA3 programmer	Any entity on the chip that has read/write access to the EDMA3 registers and can program an EDMA3 transfer.
EDMA3 transfer controller(s) (EDMA3TC)	Transfer controllers are the transfer engine for the EDMA3. Performs the read/writes as dictated by the transfer requests submitted by the EDMA3CC.
Enhanced direct memory access (EDMA3) controller	Consists of the EDMA3 channel controller (EDMA3CC) and EDMA3 transfer controller(s) (EDMA3TC). Is referred to as EDMA3 in this document.
Link parameter set	A PaRAM set that is used for linking.
Linking	The mechanism of reloading a PaRAM set with new transfer characteristics on completion of the current transfer.
Memory-mapped slave	All on-chip memories, off-chip memories, and slave peripherals. These typically rely on the EDMA3 (or other master peripheral) to perform transfers to and from them.
Master peripherals	All peripherals that are capable of initiating read and write transfers to the peripherals system and may not solely rely on the EDMA3 for their data transfers.
Null set or Null PaRAM set	A PaRAM set that has all count fields cleared (except for the link field). A dummy PaRAM set has at least one of the count fields nonzero.
Null transfer	A trigger event for a null PaRAM set results in the EDMA3CC performing a null transfer. This is an error condition. A dummy transfer is not an error condition.
QDMA channel	One of the 8 channels that can be triggered when writing to the trigger word (TRWORD) of a PaRAM set. All QDMA channels exist in the EDMA3CC.
Parameter RAM (PaRAM)	Programmable RAM that stores PaRAM sets used by DMA channels, QDMA channels, and linking.
Parameter RAM (PaRAM) set	A 32-byte EDMA3 channel transfer definition. Each parameter set consists of 8 words (4-bytes each), which store the context for a DMA/QDMA/link transfer. A PaRAM set includes source address, destination address, counts, indexes, options, etc.
Parameter RAM (PaRAM) set entry	One of the 4-byte components of the parameter set.
Slave end points	All on-chip memories, off-chip memories, and slave peripherals. These rely on the EDMA3 to perform transfers to and from them.
Transfer request (TR)	A command for data movement that is issued from the EDMA3CC to the EDMA3TC. A TR includes source and destination addresses, counts, indexes, options, etc.
Trigger event	Action that causes the EDMA3CC to service the PaRAM set and submit a transfer request to the EDMA3TC. Trigger events for DMA channels include manual triggered (CPU triggered), external event triggered, and chain triggered. Trigger events for QDMA channels include autotriggered and link triggered.
Trigger word	For QDMA channels, the trigger word specifies the PaRAM set entry that when written results in a QDMA trigger event. The trigger word is programmed via the QDMA channel *n* mapping register (QCHMAP*n*) and can point to any PaRAM set entry.
TR synchronization (sync) event	See Trigger event.

Copyright © 2013–2016, Texas Instruments Incorporated

17.2 Architecture

This section discusses the architecture of the EDMA3 controller.

17.2.1 Functional Overview

This section provides an overview of the EDMA3 channel controller (EDMA3CC) and EDMA3 transfer controller (EDMA3TC).

17.2.1.1 EDMA3 Channel Controller (EDMA3CC)

Figure 17-2 shows a functional block diagram of the EDMA3 channel controller (EDMA3CC).

The main blocks of the EDMA3CC are:

- DMA/QDMA Channel Logic: This block consists of logic that captures external system or peripheral events that can be used to initiate event triggered transfers, it also includes registers that allow configuring the DMA/QDMA channels (queue mapping, PaRAM entry mapping). It includes all the registers for different trigger type (manual, external events, chained and auto triggered) for enabling/disabling events, and monitor event status.
- Parameter RAM (PaRAM): Maintains parameter set entries for channel and reload parameter sets. The PaRAM needs to be written with the transfer context for the desired channels and link parameter sets.
- Event queues: These form the interface between the event detection logic and the transfer request submission logic.
- Transfer Request Submission Logic: This logic processes PaRAM sets based on a trigger event submitted to the event queue and submits a transfer request (TR) to the transfer controller associated with the event queue.
- Completion detection: The completion detect block detects completion of transfers by the EDMA3 transfer controller (EDMA3TC) and/or slave peripherals. Completion of transfers can optionally be used to chain trigger new transfers or to assert interrupts. The logic includes the interrupt processing registers for enabling/disabling interrupt (to be sent to the CPU), interrupt status/clearing registers.

Additionally there are:

- Region registers: Region registers allow DMA resources (DMA channels and interrupts) to be assigned to unique regions, which can be owned by unique EDMA programmers (a use model for hetero/multi core devices) or by unique tasks/threads (a use model for single core devices).
- Debug registers: Debug registers allow debug visibility by providing registers to read the queue status, channel controller status (what logic within the CC is active), and missed event status.

The EDMA3CC includes two channel types: DMA channels and QDMA channels.

Each channel is associated with a given event queue/transfer controller and with a given PaRAM set. The main difference between a DMA channel and QDMA channel is how the transfers are triggered by the system. See Section 17.2.4.

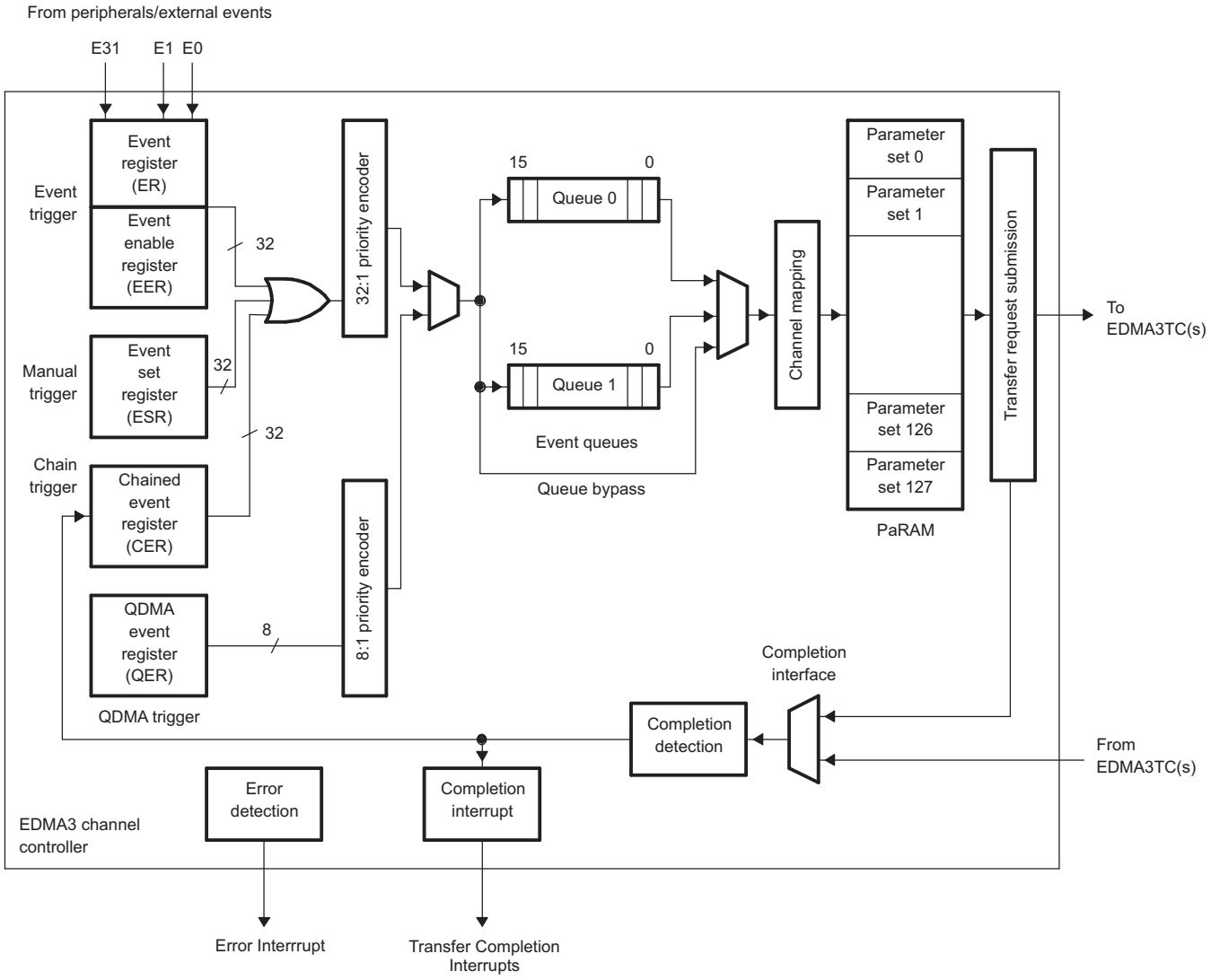
A trigger event is needed to initiate a transfer. For DMA channels, a trigger event may be due to an external event, manual write to the event set register, or chained event. QDMA channels are autotriggered when a write is performed to the user-programmed trigger word. All such trigger events are logged into appropriate registers upon recognition. See DMA channel registers (Section 17.4.2.5) and QDMA channel registers (Section 17.4.2.7).

Once a trigger event is recognized, the event type/channel is queued in the appropriate EDMA3CC event queue. The assignment of each DMA/QDMA channel to event queue is programmable. Each queue is 16 deep, so up to 16 events may be queued (on a single queue) in the EDMA3CC at an instant in time. Additional pending events mapped to a full queue are queued when event queue space becomes available. See Section 17.2.10.

If events on different channels are detected simultaneously, the events are queued based on fixed priority arbitration scheme with the DMA channels being higher priority than the QDMA channels. Among the two groups of channels, the lowest-numbered channel is the highest priority.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 17-2. EDMA3 Channel Controller (EDMA3CC) Block Diagram



Each event in the event queue is processed in the order it was queued. On reaching the head of the queue, the PaRAM associated with that channel is read to determine the transfer details. The TR submission logic evaluates the validity of the TR and is responsible for submitting a valid transfer request (TR) to the appropriate EDMA3TC (based on the event queue to EDMA3TC association, Q0 goes to TC0, and Q1 goes to TC1, etc.). For more details, see Section 17.2.3.

The EDMA3TC receives the request and is responsible for data movement as specified in the transfer request packet (TRP) and other necessary tasks like buffering, ensuring transfers are carried out in an optimal fashion wherever possible. For more details on EDMA3TC, see Section 17.2.1.2.

You may have chosen to receive an interrupt or chain to another channel on completion of the current transfer in which case the EDMA3TC signals completion to the EDMA3CC completion detection logic when the transfer is done. You can alternately choose to trigger completion when a TR leaves the EDMA3CC boundary rather than wait for all the data transfers to complete. Based on the setting of the EDMA3CC interrupt registers, the completion interrupt generation logic is responsible for generating EDMA3CC completion interrupts to the CPU. For more details, see Section 17.2.5.

Additionally, the EDMA3CC also has an error detection logic, which causes error interrupt generation on various error conditions (like missed events, exceeding event queue thresholds, etc.). For more details on error interrupts, see Section 17.2.9.4.

Copyright © 2013–2016, Texas Instruments Incorporated

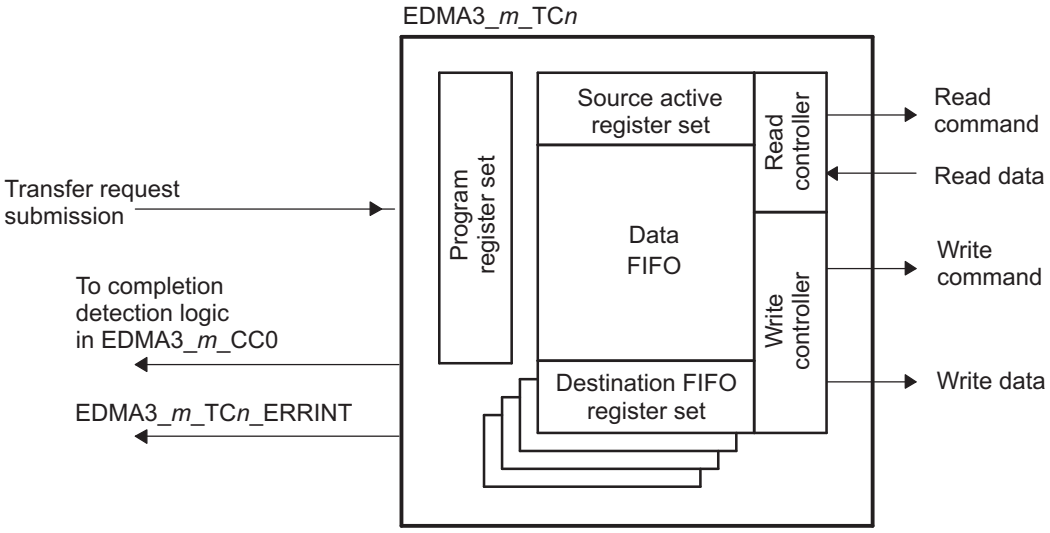
17.2.1.2 EDMA3 Transfer Controller (EDMA3TC)

Figure 17-3 shows a functional block diagram of the EDMA3 transfer controller (EDMA3TC).

The main blocks of the EDMA3TC are:

- DMA program register set: The DMA program register set stores the transfer requests received from the EDMA3 channel controller (EDMA3CC).
- DMA source active register set: The DMA source active register set stores the context for the DMA transfer request currently in progress in the read controller.
- Read controller: The read controller issues read commands to the source address.
- Destination FIFO register set: The destination (Dst) FIFO register set stores the context for the DMA transfer request(s) currently in progress or pending in the write controller.
- Write controller: The write controller issues write commands/write data to the destination address.
- Data FIFO: The data FIFO holds temporary in-flight data. The source peripheral's read data is stored in the data FIFO and subsequently written to the destination peripheral/end point by the write controller.
- Completion interface: The completion interface sends completion codes to the EDMA3CC when a transfer completes, and is used for generating interrupts and chained events (see Section 17.2.5 for details on transfer completion reporting).

Figure 17-3. EDMA3 Transfer Controller (EDMA3TC) Block Diagram



When the EDMA3TC is idle and receives its first TR, the TR is received in the DMA program register set, where it transitions to the DMA source active set and the destination FIFO register set immediately. The source active register set tracks the commands for the source side of the transfers, and the destination FIFO register set tracks commands for the destination side of the transfer. The second TR (if pending from EDMA3CC) is loaded into the DMA program set, ensuring it can start as soon as possible when the active transfer (the transfer in the source active set) is completed. As soon as the current active set is exhausted, the TR is loaded from the DMA program register set into the DMA source active register set as well as to the appropriate entry in the destination FIFO register set.

The read controller issues read commands governed by the rules of command fragmentation and optimization. These are issued only when the data FIFO has space available for the read data. The number of read commands issued depends on the TR transfer size. The TC write controller starts issuing write commands as soon as sufficient data is read in the data FIFO for the write controller to issue optimally sized write commands following the rules for command fragmentation and optimization. For details on command fragmentation and optimization, see Section 17.2.11.1.2.

Copyright © 2013–2016, Texas Instruments Incorporated

The DSTREGDEPTH parameter (fixed for a given transfer controller) determines the number of entries in the Dst FIFO register set. The number of entries determines the amount of TR pipelining possible for a given TC. The write controller can manage the write context for the number of entries in the Dst FIFO register set. This allows the read controller to go ahead and issue read commands for the subsequent TRs while the Dst FIFO register set manages the write commands and data for the previous TR. In summary, if the DSTREGDEPTH is *n*, the read controller is able to process up to *n*TRs ahead of the write controller. However, the overall TR pipelining is also subject to the amount of free space in the data FIFO.

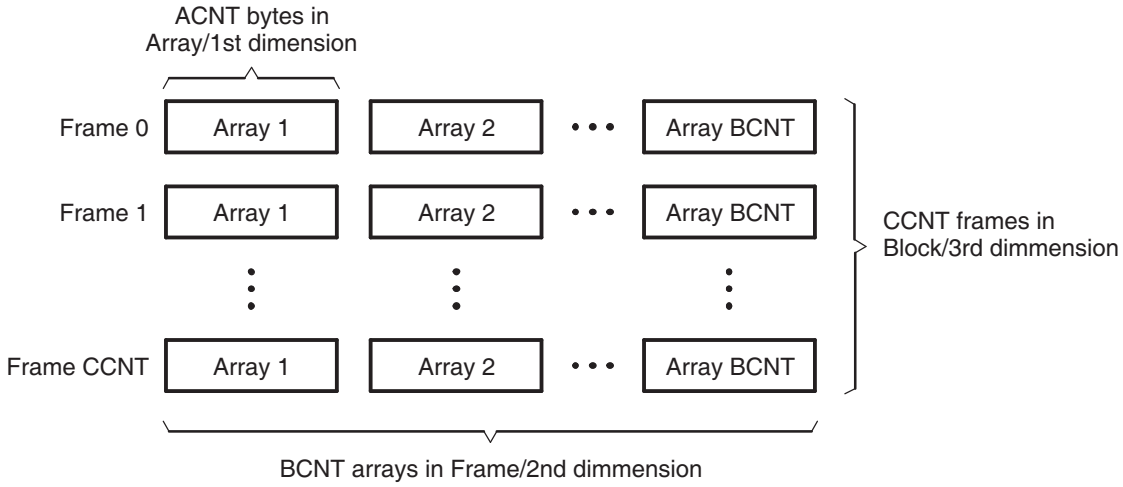
17.2.2 Types of EDMA3 Transfers

An EDMA3 transfer is always defined in terms of three dimensions. Figure 17-4 shows the three dimensions used by EDMA3 transfers. These three dimensions are defined as:

- 1st Dimension or Array (A): The 1st dimension in a transfer consists of ACNT contiguous bytes.
- 2nd Dimension or Frame (B): The 2nd dimension in a transfer consists of BCNT arrays of ACNT bytes. Each array transfer in the 2nd dimension is separated from each other by an index programmed using SRCBIDX or DSTBIDX.
- 3rd Dimension or Block (C): The 3rd dimension in a transfer consists of CCNT frames of BCNT arrays of ACNT bytes. Each transfer in the 3rd dimension is separated from the previous by an index programmed using SRCCIDX or DSTCIDX.

Note that the reference point for the index depends on the synchronization type. The amount of data transferred upon receipt of a trigger/synchronization event is controlled by the synchronization types (SYNCDIM bit in OPT). Of the three dimensions, only two synchronization types are supported: A-synchronized transfers and AB-synchronized transfers.

Figure 17-4. Definition of ACNT, BCNT, and CCNT



Copyright © 2013–2016, Texas Instruments Incorporated

17.2.2.1 A-Synchronized Transfers

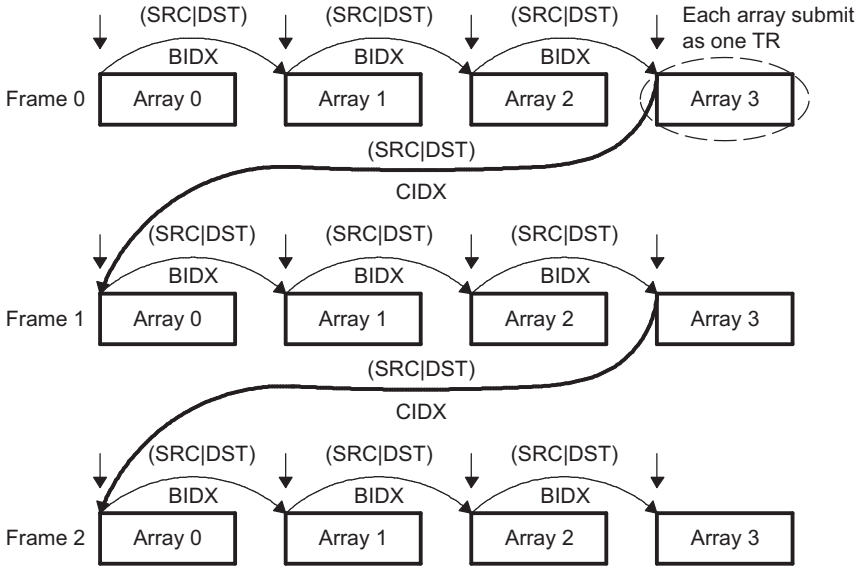
In an A-synchronized transfer, each EDMA3 sync event initiates the transfer of the 1st dimension of ACNT bytes, or one array of ACNT bytes. In other words, each event/TR packet conveys the transfer information for one array only. Thus, BCNT × CCNT events are needed to completely service a PaRAM set.

Arrays are always separated by SRCBIDX and DSTBIDX, as shown in Figure 17-5, where the start address of Array N is equal to the start address of Array N – 1 plus source (SRCBIDX) or destination (DSTBIDX).

Frames are always separated by SRCCIDX and DSTCIDX. For A-synchronized transfers, after the frame is exhausted, the address is updated by adding SRCCIDX/DSTCIDX to the beginning address of the last array in the frame. As in Figure 17-5, SRCCIDX/DSTCIDX is the difference between the start of Frame 0 Array 3 to the start of Frame 1 Array 0.

Figure 17-5 shows an A-synchronized transfer of 3 (CCNT) frames of 4 (BCNT) arrays of n (ACNT) bytes. In this example, a total of 12 sync events (BCNT × CCNT) exhaust a PaRAM set. See Section 17.2.3.6 for details on parameter set updates.

Figure 17-5. A-Synchronized Transfers (ACNT = n, BCNT = 4, CCNT = 3)



Copyright © 2013–2016, Texas Instruments Incorporated

17.2.2.2 AB-Synchronized Transfers

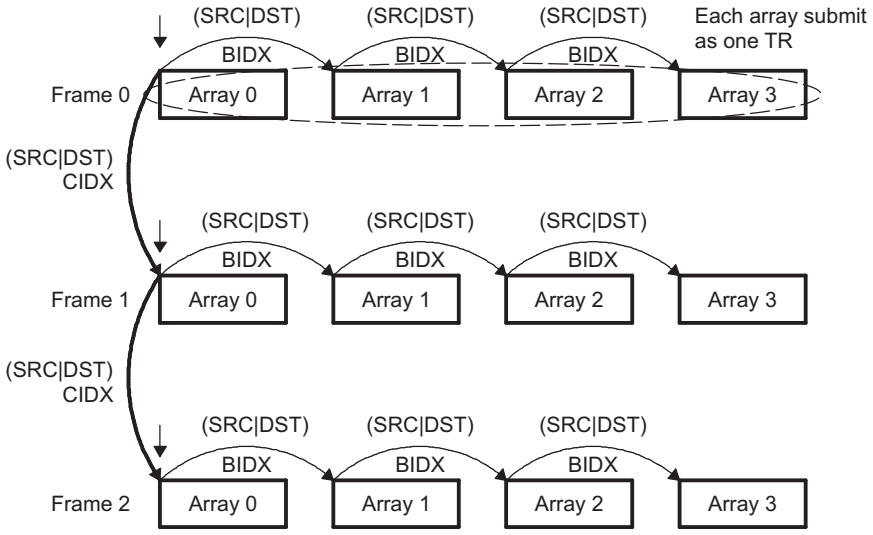
In a AB-synchronized transfer, each EDMA3 sync event initiates the transfer of 2 dimensions or one frame. In other words, each event/TR packet conveys information for one entire frame of BCNT arrays of ACNT bytes. Thus, CCNT events are needed to completely service a PaRAM set.

Arrays are always separated by SRCBIDX and DSTBIDX as shown in Figure 17-6. Frames are always separated by SRCCIDX and DSTCIDX.

Note that for AB-synchronized transfers, after a TR for the frame is submitted, the address update is to add SRCCIDX/DSTCIDX to the beginning address of the beginning array in the frame. This is different from A-synchronized transfers where the address is updated by adding SRCCIDX/DSTCIDX to the start address of the last array in the frame. See Section 17.2.3.6 for details on parameter set updates.

Figure 17-6 shows an AB-synchronized transfer of 3 (CCNT) frames of 4 (BCNT) arrays of *n* (ACNT) bytes. In this example, a total of 3 sync events (CCNT) exhaust a PaRAM set; that is, a total of 3 transfers of 4 arrays each completes the transfer.

Figure 17-6. AB-Synchronized Transfers (ACNT = n, BCNT = 4, CCNT = 3)



NOTE: ABC-synchronized transfers are not directly supported. But can be logically achieved by chaining between multiple AB-synchronized transfers.

Copyright © 2013–2016, Texas Instruments Incorporated

17.2.3 Parameter RAM (PaRAM)

The EDMA3 controller is a RAM-based architecture. The transfer context (source/destination addresses, count, indexes, etc.) for DMA or QDMA channels is programmed in a parameter RAM table within the EDMA3CC, referred to as PaRAM. The PaRAM table is segmented into multiple PaRAM sets. Each PaRAM set includes eight 4-byte PaRAM set entries (32-bytes total per PaRAM set), which includes typical DMA transfer parameters such as source address, destination address, transfer counts, indexes, options, etc. See your device-specific data manual for the addresses of the PaRAM set entries.

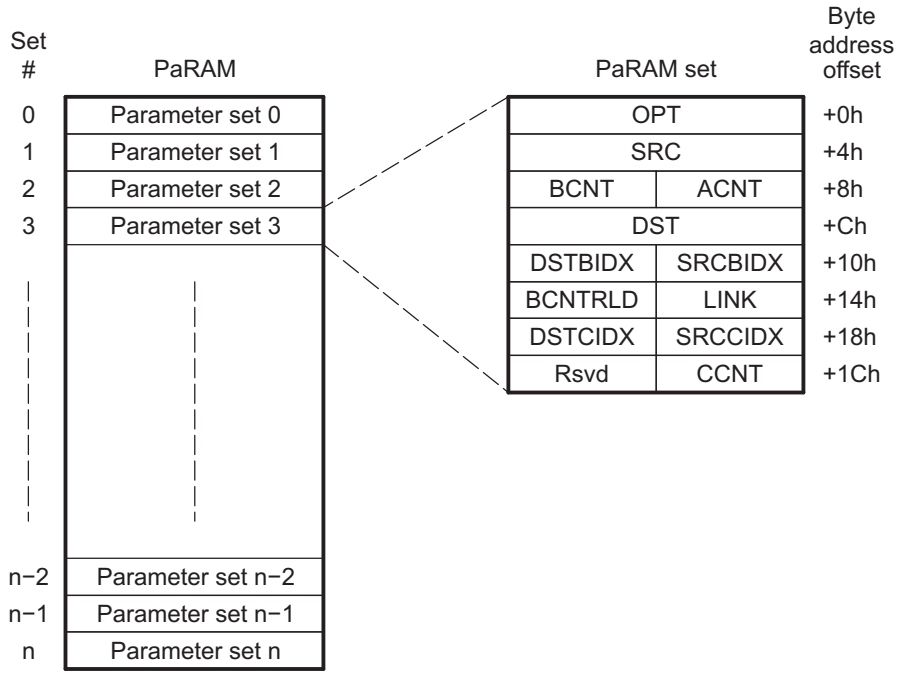
The PaRAM structure supports flexible ping-pong, circular buffering, channel chaining, and autoreloading (linking). The first *n* PaRAM sets are directly mapped to the DMA channels (where *n* is the number of DMA channels supported in the EDMA3CC for a specific device). The remaining PaRAM sets can be used for link entries or associated with QDMA channels. Additionally if the DMA channels are not used, the PaRAM sets associated with the unused DMA channels can also be used for link entries or QDMA channels.

NOTE: By default, QDMA channels are mapped to PaRAM set 0. These should be remapped before use, see Section 17.2.6.2.

17.2.3.1 PaRAM Set

Each parameter set of PaRAM is organized into eight 32-bit words or 32 bytes, as shown in Figure 17-7 and described in Table 17-1. Each PaRAM set consists of 16-bit and 32-bit parameters.

Figure 17-7. PaRAM Set



Note: *n* is the number of PaRAM sets supported in the EDMA3CC for a specific device.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 17-1. EDMA3 Channel Parameter Description

Offset Address (bytes)	Acronym	Parameter	Description
0h	OPT	Channel Options	Transfer Configuration Options
4h	SRC	Channel Source Address	The byte address from which data is transferred.
8h[(1)]	ACNT	Count for 1st Dimension	Unsigned value specifying the number of contiguous bytes within an array (first dimension of the transfer). Valid values range from 1 to 65 535.
	BCNT	Count for 2nd Dimension	Unsigned value specifying the number of arrays in a frame, where an array is ACNT bytes. Valid values range from 1 to 65 535.
Ch	DST	Channel Destination Address	The byte address to which data is transferred.
10h[(1)]	SRCBIDX	Source BCNT Index	Signed value specifying the byte address offset between source arrays within a frame (2nd dimension). Valid values range from –32 768 and 32 767.
	DSTBIDX	Destination BCNT Index	Signed value specifying the byte address offset between destination arrays within a frame (2nd dimension). Valid values range from –32 768 and 32 767.
14h[(1)]	LINK	Link Address	The PaRAM address containing the PaRAM set to be linked (copied from) when the current PaRAM set is exhausted. A value of FFFFh specifies a null link.
	BCNTRLD	BCNT Reload	The count value used to reload BCNT when BCNT decrements to 0 (TR submitted for the last array in 2nd dimension). Only relevant in A-synchronized transfers.
18h[(1)]	SRCCIDX	Source CCNT Index	Signed value specifying the byte address offset between frames within a block (3rd dimension). Valid values range from –32 768 and 32 767. A-synchronized transfers: The byte address offset from the beginning of the last source array in a frame to the beginning of the first source array in the next frame. AB-synchronized transfers: The byte address offset from the beginning of the first source array in a frame to the beginning of the first source array in the next frame.
	DSTCIDX	Destination CCNT index	Signed value specifying the byte address offset between frames within a block (3rd dimension). Valid values range from –32 768 and 32 767. A-synchronized transfers: The byte address offset from the beginning of the last destination array in a frame to the beginning of the first destination array in the next frame. AB-synchronized transfers: The byte address offset from the beginning of the first destination array in a frame to the beginning of the first destination array in the next frame.
1Ch	CCNT	Count for 3rd Dimension	Unsigned value specifying the number of frames in a block, where a frame is BCNT arrays of ACNT bytes. Valid values range from 1 to 65 535.
	RSVD	Reserved	Reserved

(1) If OPT, SRC, or DST is the trigger word for a QDMA transfer then it is required to do a 32-bit access to that field. Furthermore, it is recommended to perform only 32-bit accesses on the parameter RAM for best code compatibility. For example, switching the endianness of the processor swaps addresses of the 16-bit fields, but 32-bit accesses avoid the issue entirely.

Copyright © 2013–2016, Texas Instruments Incorporated

17.2.3.2 EDMA3 Channel Parameter Set Fields

17.2.3.2.1 Channel Options Parameter (OPT)

The 32-bit channel options parameter (OPT) specifies the transfer configuration options. The channel options parameter (OPT) is described in Section 17.4.1.1.

17.2.3.2.2 Channel Source Address (SRC)

The 32-bit source address parameter specifies the starting byte address of the source. For SAM in increment mode, there are no alignment restrictions imposed by EDMA3. For SAM in constant addressing mode, you must program the source address to be aligned to a 256-bit aligned address (5 LSBs of address must be 0). The EDMA3TC will signal an error, if this rule is violated. See Section 17.2.11.2 for additional details.

17.2.3.2.3 Channel Destination Address (DST)

The 32-bit destination address parameter specifies the starting byte address of the destination. For DAM in increment mode, there are no alignment restrictions imposed by EDMA3. For DAM in constant addressing mode, you must program the destination address to be aligned to a 256-bit aligned address (5 LSBs of address must be 0). The EDMA3TC will signal an error, if this rule is violated. See Section 17.2.11.2 for additional details.

17.2.3.2.4 Count for 1st Dimension (ACNT)

ACNT represents the number of bytes within the 1st dimension of a transfer. ACNT is a 16-bit unsigned value with valid values between 0 and 65 535. Therefore, the maximum number of bytes in an array is 65 535 bytes (64K – 1 bytes). ACNT must be greater than or equal to 1 for a TR to be submitted to EDMA3TC. A transfer with ACNT equal to 0 is considered either a null or dummy transfer.

See Section 17.2.3.5 and Section 17.2.5.3 for details on dummy/null completion conditions.

17.2.3.2.5 Count for 2nd Dimension (BCNT)

BCNT is a 16-bit unsigned value that specifies the number of arrays of length ACNT. For normal operation, valid values for BCNT are between 1 and 65 535. Therefore, the maximum number of arrays in a frame is 65 535 (64K – 1 arrays). A transfer with BCNT equal to 0 is considered either a null or dummy transfer.

See Section 17.2.3.5 and Section 17.2.5.3 for details on dummy/null completion conditions.

17.2.3.2.6 Count for 3rd Dimension (CCNT)

CCNT is a 16-bit unsigned value that specifies the number of frames in a block. Valid values for CCNT are between 1 and 65 535. Therefore, the maximum number of frames in a block is 65 535 (64K – 1 frames). A transfer with CCNT equal to 0 is considered either a null or dummy transfer.

See Section 17.2.3.5 and Section 17.2.5.3 for details on dummy/null completion conditions.

17.2.3.2.7 BCNT Reload (BCNTRLD)

BCNTRLD is a 16-bit unsigned value used to reload the BCNT field once the last array in the 2nd dimension is transferred. This field is only used for A-synchronized transfers. In this case, the EDMA3CC decrements the BCNT value by 1 on each TR submission. When BCNT reaches 0, the EDMA3CC decrements CCNT and uses the BCNTRLD value to reinitialize the BCNT value.

For AB-synchronized transfers, the EDMA3CC submits the BCNT in the TR and the EDMA3TC decrements BCNT appropriately. For AB-synchronized transfers, BCNTRLD is not used.

Copyright © 2013–2016, Texas Instruments Incorporated

17.2.3.2.8 Source B Index (SRCBIDX)

SRCBIDX is a 16-bit signed value (2s complement) used for source address modification between each array in the 2nd dimension. Valid values for SRCBIDX are between –32 768 and 32 767. It provides a byte address offset from the beginning of the source array to the beginning of the next source array. It applies to both A-synchronized and AB-synchronized transfers. Some examples:

- SRCBIDX = 0000h (0): no address offset from the beginning of an array to the beginning of the next array. All arrays are fixed to the same beginning address.
- SRCBIDX = 0003h (+3): the address offset from the beginning of an array to the beginning of the next array in a frame is 3 bytes. For example, if the current array begins at address 1000h, the next array begins at 1003h.
- SRCBIDX = FFFFh (–1): the address offset from the beginning of an array to the beginning of the next array in a frame is –1 byte. For example, if the current array begins at address 5054h, the next array begins at 5053h.

17.2.3.2.9 Destination B Index (DSTBIDX)

DSTBIDX is a 16-bit signed value (2s complement) used for destination address modification between each array in the 2nd dimension. Valid values for DSTBIDX are between –32 768 and 32 767. It provides a byte address offset from the beginning of the destination array to the beginning of the next destination array within the current frame. It applies to both A-synchronized and AB-synchronized transfers. See SRCBIDX (Section 17.2.3.2.8) for examples.

17.2.3.2.10 Source C Index (SRCCIDX)

SRCCIDX is a 16-bit signed value (2s complement) used for source address modification in the 3rd dimension. Valid values for SRCCIDX are between –32 768 and 32 767. It provides a byte address offset from the beginning of the current array (pointed to by SRC address) to the beginning of the first source array in the next frame. It applies to both A-synchronized and AB-synchronized transfers. Note that when SRCCIDX is applied, the current array in an A-synchronized transfer is the last array in the frame (Figure 17-5), while the current array in an AB-synchronized transfer is the first array in the frame (Figure 17-6).

17.2.3.2.11 Destination C Index (DSTCIDX)

DSTCIDX is a 16-bit signed value (2s complement) used for destination address modification in the 3rd dimension. Valid values are between –32 768 and 32 767. It provides a byte address offset from the beginning of the current array (pointed to by DST address) to the beginning of the first destination array TR in the next frame. It applies to both A-synchronized and AB-synchronized transfers. Note that when DSTCIDX is applied, the current array in an A-synchronized transfer is the last array in the frame (Figure 17-5), while the current array in a AB-synchronized transfer is the first array in the frame (Figure 17-6).

17.2.3.2.12 Link Address (LINK)

The EDMA3CC provides a mechanism, called linking, to reload the current PaRAM set upon its natural termination (that is, after the count fields are decremented to 0) with a new PaRAM set. The 16-bit parameter LINK specifies the byte address offset in the PaRAM from which the EDMA3CC loads/reloads the next PaRAM set during linking.

You must program the link address to point to a valid aligned 32-byte PaRAM set. The 5 LSBs of the LINK field should be cleared to 0.

The EDMA3CC ignores the upper 2 bits of the LINK entry, allowing the programmer the flexibility of programming the link address as either an absolute/literal byte address or use the PaRAM-base-relative offset address. Therefore, if you make use of the literal address with a range from 4000h to 7FFFh, it will be treated as a PaRAM-base-relative value of 0000h to 3FFFh.

You should make sure to program the LINK field correctly, so that link update is requested from a PaRAM address that falls in the range of the available PaRAM addresses on the device.

Copyright © 2013–2016, Texas Instruments Incorporated

A LINK value of FFFFh is referred to as a NULL link that should cause the EDMA3CC to perform an internal write of 0 to all entries of the current PaRAM set, except for the LINK field that is set to FFFFh. Also, see Section 17.2.5 for details on terminating a transfer.

17.2.3.3 Null PaRAM Set

A null PaRAM set is defined as a PaRAM set where all count fields (ACNT, BCNT, and CCNT) are cleared to 0. If a PaRAM set associated with a channel is a NULL set, then when serviced by the EDMA3CC, the bit corresponding to the channel is set in the associated event missed register (EMR or QEMR). This bit remains set in the associated secondary event register (SER or QSER). *This implies that any future events on the same channel are ignored by the EDMA3CC and you are required to clear the bit in SER or QSER for the channel.* This is considered an error condition, since events are not expected on a channel that is configured as a null transfer. See Section 17.4.2.5.8 and Section 17.4.2.2.1 for more information on the SER and EMR registers, respectively.

17.2.3.4 Dummy PaRAM Set

A dummy PaRAM set is defined as a PaRAM set where at least one of the count fields (ACNT, BCNT, or CCNT) is cleared to 0 and at least one of the count fields is nonzero.

If a PaRAM set associated with a channel is a dummy set, then when serviced by the EDMA3CC, it will not set the bit corresponding to the channel (DMA/QDMA) in the event missed register (EMR or QEMR) and the secondary event register (SER or QSER) bit gets cleared similar to a normal transfer. Future events on that channel are serviced. A dummy transfer is a legal transfer of 0 bytes. See Section 17.4.2.5.8 and Section 17.4.2.2.1 for more information on the SER and EMR registers, respectively.

17.2.3.5 Dummy Versus Null Transfer Comparison

There are some differences in the way the EDMA3CC logic treats a dummy versus a null transfer request. A null transfer request is an error condition, but a dummy transfer is a legal transfer of 0 bytes. A null transfer causes an error bit (E*n*) in EMR to get set and the E*n* bit in SER remains set, essentially preventing any further transfers on that channel without clearing the associated error registers.

Table 17-2 summarizes the conditions and effects of null and dummy transfer requests.

Table 17-2. Dummy and Null Transfer Request

Feature	Null TR	Dummy TR
EMR/QEMR is set	Yes	No
SER/QSER remains set	Yes	No
Link update (STATIC = 0 in OPT)	Yes	Yes
QER is set	Yes	Yes
IPR and CER is set using early completion	Yes	Yes

17.2.3.6 Parameter Set Updates

When a TR is submitted for a given DMA/QDMA channel and its corresponding PaRAM set, the EDMA3CC is responsible for updating the PaRAM set in anticipation of the next trigger event. For nonfinal events, this includes address and count updates; for final events, this includes the link update.

The specific PaRAM set entries that are updated depend on the channel's synchronization type (A-synchronized or B-synchronized) and the current state of the PaRAM set. A B-update refers to the decrementing of BCNT in the case of A-synchronized transfers after the submission of successive TRs. A C-update refers to the decrementing of CCNT in the case of A-synchronized transfers after BCNT TRs for ACNT byte transfers have submitted. For AB-synchronized transfers, a C-update refers to the decrementing of CCNT after submission of every transfer request.

See Table 17-3 for details and conditions on the parameter updates. A link update occurs when the PaRAM set is exhausted, as described in Section 17.2.3.7.

Copyright © 2013–2016, Texas Instruments Incorporated

After the TR is read from the PaRAM (and is in process of being submitted to EDMA3TC), the following fields are updated if needed:

- A-synchronized: BCNT, CCNT, SRC, DST
- AB-synchronized: CCNT, SRC, DST

The following fields are not updated (except for during linking, where all fields are overwritten by the link PaRAM set):

- A-synchronized: ACNT, BCNTRLD, SRCBIDX, DSTBIDX, SRCCIDX, DSTCIDX, OPT, LINK
- AB-synchronized: ACNT, BCNT, BCNTRLD, SRCBIDX, DSTBIDX, SRCCIDX, DSTCIDX, OPT, LINK

Note that PaRAM updates only pertain to the information that is needed to properly submit the next transfer request to the EDMA3TC. Updates that occur while data is moved within a transfer request are tracked within the transfer controller, and is detailed in Section 17.2.11. For A-synchronized transfers, the EDMA3CC always submits a TRP for ACNT bytes (BCNT = 1 and CCNT = 1). For AB-synchronized transfers, the EDMA3CC always submits a TRP for ACNT bytes of BCNT arrays (CCNT = 1). The EDMA3TC is responsible for updating source and destination addresses within the array based on ACNT and FWID (in OPT). For AB-synchronized transfers, the EDMA3TC is also responsible to update source and destination addresses between arrays based on SRCBIDX and DSTBIDX.

Table 17-3 shows the details of parameter updates that occur within EDMA3CC for A-synchronized and AB-synchronized transfers.

Table 17-3. Parameter Updates in EDMA3CC (for Non-Null, Non-Dummy PaRAM Set)

	A-Synchronized Transfer			AB-Synchronized Transfer		
	B-Update	**C-Update**	**Link Update**	**B-Update**	**C-Update**	**Link Update**
Condition:	**BCNT > 1**	**BCNT == 1 && CCNT > 1**	**BCNT == 1 && CCNT == 1**	**N/A**	**CCNT > 1**	**CCNT == 1**
SRC	+= SRCBIDX	+= SRCCIDX	= Link.SRC	in EDMA3TC	+= SRCCIDX	= Link.SRC
DST	+= DSTBIDX	+= DSTCIDX	= Link.DST	in EDMA3TC	+= DSTCIDX	= Link.DST
ACNT	None	None	= Link.ACNT	None	None	= Link.ACNT
BCNT	–= 1	= BCNTRLD	= Link.BCNT	in EDMA3TC	N/A	= Link.BCNT
CCNT	None	–= 1	= Link.CCNT	in EDMA3TC	–=1	= Link.CCNT
SRCBIDX	None	None	= Link.SRCBIDX	in EDMA3TC	None	= Link.SRCBIDX
DSTBIDX	None	None	= Link.DSTBIDX	None	None	= Link.DSTBIDX
SRCCIDX	None	None	= Link.SRCBIDX	in EDMA3TC	None	= Link.SRCBIDX
DSTCIDX	None	None	= Link.DSTBIDX	None	None	= Link.DSTBIDX
LINK	None	None	= Link.LINK	None	None	= Link.LINK
BCNTRLD	None	None	= Link.BCNTRLD	None	None	= Link.BCNTRLD
OPT[1]	None	None	= LINK.OPT	None	None	= LINK.OPT

(1) In all cases, no updates occur if OPT.STATIC == 1 for the current PaRAM set.

NOTE: The EDMA3CC includes no special hardware to detect when an indexed address update calculation overflows/underflows. The address update will wrap across boundaries as programmed by the user. You should ensure that no transfer is allowed to cross internal port boundaries between peripherals. A single TR must target a single source/destination slave endpoint.

Copyright © 2013–2016, Texas Instruments Incorporated

17.2.3.7 Linking Transfers

The EDMA3CC provides a mechanism known as linking, which allows the entire PaRAM set to be reloaded from a location within the PaRAM memory map (for both DMA and QDMA channels). Linking is especially useful for maintaining ping-pong buffers, circular buffering, and repetitive/continuous transfers all with no CPU intervention. Upon completion of a transfer, the current transfer parameters are reloaded with the parameter set pointed to by the 16-bit link address field (of the current parameter set). Linking only occurs when the STATIC bit in OPT is cleared to 0.

NOTE: A transfer (DMA or QDMA) should always be linked to another useful transfer. If it is required to terminate a transfer, the transfer should be linked to a NULL set.

The link update occurs after the current PaRAM set event parameters have been exhausted. An event's parameters are exhausted when the EDMA3 channel controller has submitted all the transfers associated with the PaRAM set.

A link update occurs for null and dummy transfers depending on the state of the STATIC bit in OPT and the LINK field. In both cases (null or dummy), if the value of LINK is FFFFh then a null PaRAM set (with all 0s and LINK set to FFFFh) is written to the current PaRAM set. Similarly, if LINK is set to a value other than FFFFh then the appropriate PaRAM location pointed to by LINK is copied to the current PaRAM set.

Once the channel completion conditions are met for an event, the transfer parameters located at the link address are loaded into the current DMA or QDMA channel's associated parameter set. The EDMA3CC reads the entire PaRAM set (8 words) from the PaRAM set specified by LINK and writes all 8 words to the PaRAM set associated with the current channel. Figure 17-8 shows an example of a linked transfer.

Any PaRAM set in the PaRAM can be used as a link/reload parameter set; however, it is recommended that the PaRAM sets associated with peripheral synchronization events (see Section 17.2.6) should only be used for linking if the synchronization event isolated with the channel mapped to that PaRAM set is disabled.

If a PaRAM set location is mapped to a QDMA channel (by QCHMAP*n*), then copying the link PaRAM set onto the current QDMA channel PaRAM set is recognized as a trigger event and is latched in QER since a write to the trigger word was performed. This feature can be used to create a linked list of transfers using a single QDMA channel and multiple PaRAM sets.

Link-to-self transfers replicate the behavior of autoinitialization, which facilitates the use of circular buffering and repetitive transfers. After an EDMA3 channel exhausts its current PaRAM set, it reloads all the parameter set entries from another PaRAM set, which is initialized with values identical to the original PaRAM set. Figure 17-9 shows an example of a linked-to-self transfer. In Figure 17-9, parameter set 127 has the LINK field address pointing to the address of parameter set 127, that is, linked-to-self.

NOTE: If the STATIC bit in OPT is set for a PaRAM set, then link updates are not performed. The link updates performed internally by the EDMA3CC are atomic. This implies that when the EDMA3CC is updating a PaRAM set, accesses to PaRAM by other EDMA3 programmer's (for example, CPU configuration accesses) are not allowed. Also for QDMA, for example, if the first word of the PaRAM entry is defined as a trigger word, EDMA3CC logic assures that all 8 PaRAM words are updated before the new QDMA event can trigger the transfer for that PaRAM entry.

Copyright © 2013–2016, Texas Instruments Incorporated

17.2.3.7.1 Constant Addressing Mode Transfers/Alignment Issues

If either SAM or DAM is set to 1 (constant addressing mode), then the source or destination address must be aligned to a 256-bit aligned address, respectively, and the corresponding BIDX should be an even multiple of 32 bytes (256 bit). The EDMA3CC does not recognize errors here but the EDMA3TC asserts an error, if this is not true. See Section 17.2.11.2.

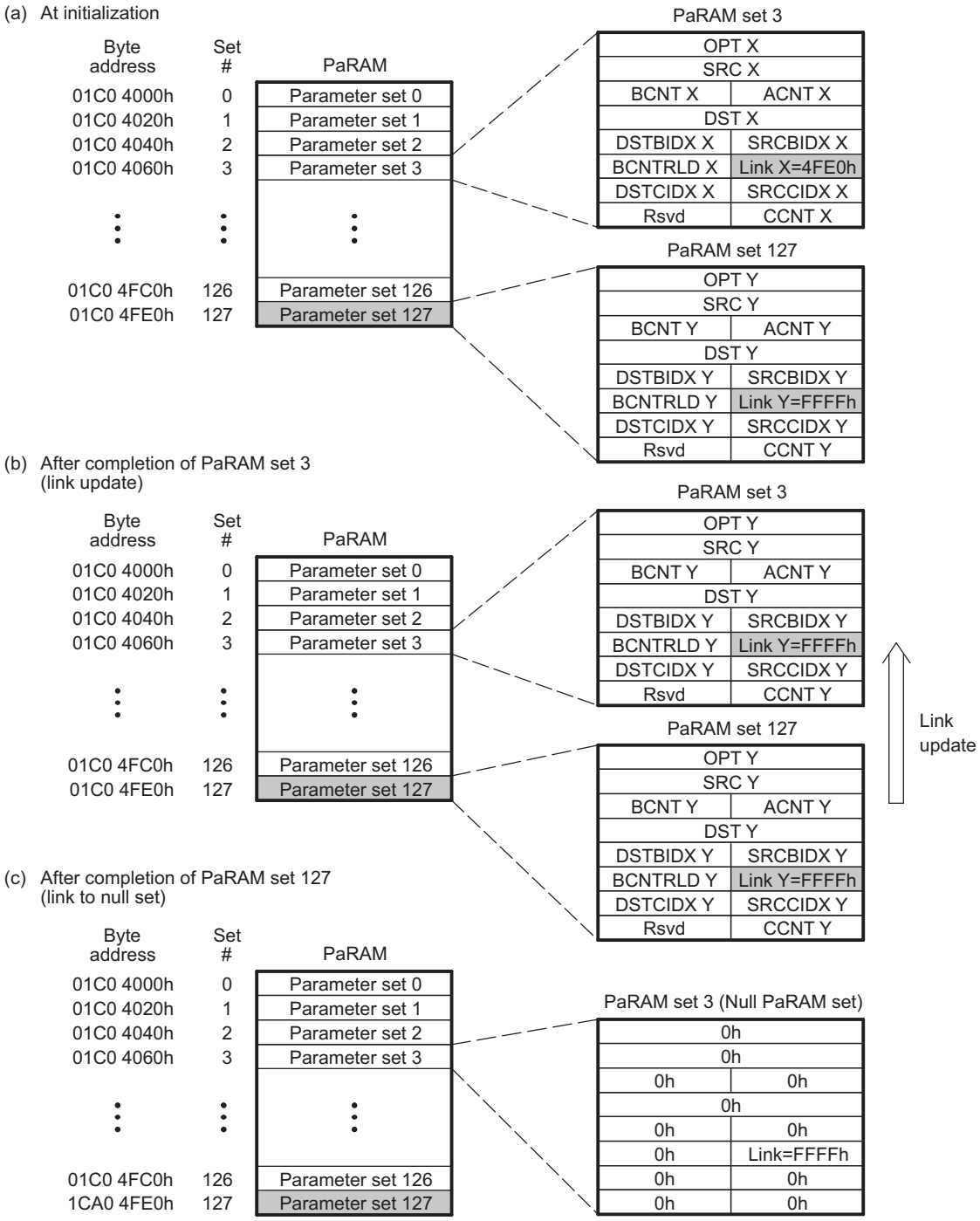
NOTE: The constant addressing (CONST) mode has limited applicability. The EDMA3 should be configured for the constant addressing mode (SAM/DAM = 1) only if the transfer source or destination (on-chip memory, off-chip memory controllers, slave peripherals) support the constant addressing mode. See your device-specific data manual to verify if constant addressing mode is supported. If the constant addressing mode is not supported, the similar logical transfer can be achieved using the increment (INCR) mode (SAM/DAM = 0) by appropriately programming the count and indices values.

17.2.3.7.2 Element Size

The EDMA3 controller does not use the concept of element-size and element-indexing. Instead, all transfers are defined in terms of all three dimensions: ACNT, BCNT, and CCNT. An element-indexed transfer is logically achieved by programming ACNT to the size of the element and BCNT to the number of elements that need to be transferred. For example, if you have 16-bit audio data and 256 audio samples that needed to be transferred to a serial port, this can be done by programming the ACNT = 2 (2 bytes) and BCNT = 256.

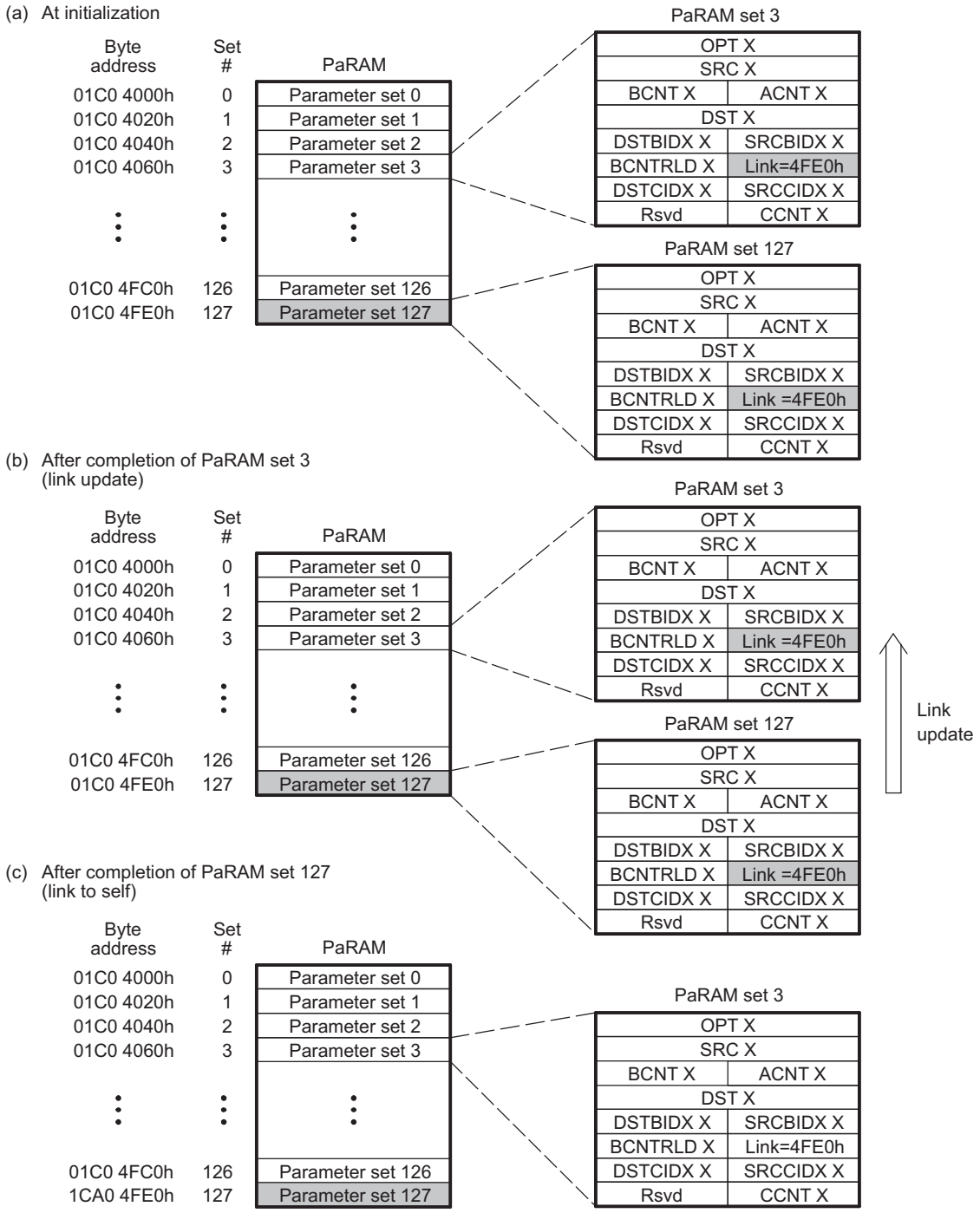
Copyright © 2013–2016, Texas Instruments Incorporated

Figure 17-8. Linked Transfer Example



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 17-9. Link-to-Self Transfer Example



Copyright © 2013–2016, Texas Instruments Incorporated

17.2.4 Initiating a DMA Transfer

There are multiple ways to initiate a programmed data transfer using the EDMA3 channel controller. Transfers on DMA channels are initiated by three sources:

- **Event-triggered transfer request** (this is the more typical usage of EDMA3): Allows for a peripheral, system, or externally-generated event to trigger a transfer request.
- **Manually-triggered transfer request:** The CPU manually triggers a transfer by writing a 1 to the corresponding bit in the event set register (ESR).
- **Chain-triggered transfer request:** A transfer is triggered on the completion of another transfer or subtransfer.

Transfers on QDMA channels are initiated by two sources:

- **Autotriggered transfer request:** A transfer is triggered when the PaRAM set entry programmed trigger word is written to.
- **Link-triggered transfer requests:** When linking occurs, the transfer is triggered when the PaRAM set entry programmed trigger word is written to.

17.2.4.1 DMA Channel

17.2.4.1.1 Event-Triggered Transfer Request

When an event is asserted from a peripheral or device pins, it gets latched in the corresponding bit of the event register (ER.E*n* = 1). If the corresponding event in the event enable register (EER) is enabled (EER.E*n* = 1), then the EDMA3CC prioritizes and queues the event in the appropriate event queue. When the event reaches the head of the queue, it is evaluated for submission as a transfer request to the transfer controller.

If the PaRAM set is valid (not a NULL set), then a transfer request packet (TRP) is submitted to the EDMA3TC and the E*n* bit in ER is cleared. At this point, a new event can be safely received by the EDMA3CC.

If the PaRAM set associated with the channel is a NULL set (see Section 17.2.3.3), then no transfer request (TR) is submitted and the corresponding E*n* bit in ER is cleared and simultaneously the corresponding channel bit is set in the event miss register (EMR.E*n* = 1) to indicate that the event was discarded due to a null TR being serviced. Good programming practices should include cleaning the event missed error before retriggering the DMA channel.

When an event is received, the corresponding event bit in the event register is set (ER.E*n* = 1), regardless of the state of EER.E*n*. If the event is disabled when an external event is received (ER.E*n* = 1 and EER.E*n* = 0), the ER.E*n* bit remains set. If the event is subsequently enabled (EER.E*n* = 1), then the pending event is processed by the EDMA3CC and the TR is processed/submitted, after which the ER.E*n* bit is cleared.

If an event is being processed (prioritized or is in the event queue) and another sync event is received for the same channel prior to the original being cleared (ER.E*n* != 0), then the second event is registered as a missed event in the corresponding bit of the event missed register (EMR.E*n* = 1).

For the synchronization events associated with each of the programmable DMA channels, see your device-specific data manual to determine the event to channel mapping.

Copyright © 2013–2016, Texas Instruments Incorporated

17.2.4.1.2 Manually-Triggered Transfer Request

A DMA transfer is initiated by a write to the event set register (ESR) by the CPU (or any EDMA programmer). Writing a 1 to an event bit in the ESR results in the event being prioritized/queued in the appropriate event queue, regardless of the state of the EER.E*n* bit. When the event reaches the head of the queue, it is evaluated for submission as a transfer request to the transfer controller.

As in the event-triggered transfers, if the PaRAM set associated with the channel is valid (it is not a null set) then the TR is submitted to the associated EDMA3TC and the channel can be triggered again.

If the PaRAM set associated with the channel is a NULL set (see Section 17.2.3.3), then no transfer request (TR) is submitted and the corresponding E*n* bit in ER is cleared and simultaneously the corresponding channel bit is set in the event miss register (EMR.E*n* = 1) to indicate that the event was discarded due to a null TR being serviced. Good programming practices should include clearing the event missed error before retriggering the DMA channel.

If an event is being processed (prioritized or is in the event queue) and the same channel is manually set by a write to the corresponding channel bit of the event set register (ESR.E*n* = 1) prior to the original being cleared (ESR.E*n* = 0), then the second event is registered as a missed event in the corresponding bit of the event missed register (EMR.E*n* = 1).

17.2.4.1.3 Chain-Triggered Transfer Request

Chaining is a mechanism by which the completion of one transfer automatically sets the event for another channel. When a chained completion code is detected, the value of which is dictated by the transfer completion code (TCC[5:0] in OPT of the PaRAM set associated with the channel), it results in the corresponding bit in the chained event register (CER) to be set (CER.E[TCC] = 1).

Once a bit is set in CER, the EDMA3CC prioritizes and queues the event in the appropriate event queue. When the event reaches the head of the queue, it is evaluated for submission as a transfer request to the transfer controller.

As in the event-triggered transfers, if the PaRAM set associated with the channel is valid (it is not a null set) then the TR is submitted to the associated EDMA3TC and the channel can be triggered again.

If the PaRAM set associated with the channel is a NULL set (see Section 17.2.3.3), then no transfer request (TR) is submitted and the corresponding E*n* bit in CER is cleared and simultaneously the corresponding channel bit is set in the event miss register (EMR.E*n* = 1) to indicate that the event was discarded due to a null TR being serviced. In this case, the error condition must be cleared by you before the DMA channel can be retriggered. Good programming practices might include clearing the event missed error before retriggering the DMA channel.

If a chaining event is being processed (prioritized or queued) and another chained event is received for the same channel prior to the original being cleared (CER.E*n* != 0), then the second chained event is registered as a missed event in the corresponding channel bit of the event missed register (EMR.E*n* = 1).

NOTE: Chained event registers, event registers, and event set registers operate independently. An event (E*n*) can be triggered by any of the trigger sources (event-triggered, manually-triggered, or chain-triggered).

Copyright © 2013–2016, Texas Instruments Incorporated

17.2.4.2 QDMA Channels

17.2.4.2.1 Autotriggered and Link-Triggered Transfer Request

NOTE: If OPT, SRC, or DST is the trigger word for a QDMA transfer then it is required to do a 32-bit access to that field.

QDMA-based transfer requests are issued when a QDMA event gets latched in the QDMA event register (QER.E*n* = 1). A bit corresponding to a QDMA channel is set in the QDMA event register (QER) when the following occurs:

- A CPU (or any EDMA3 programmer) write occurs to a PaRAM address that is defined as a QDMA channel trigger word (programmed in the QDMA channel *n* mapping register (QCHMAP*n*)) for the particular QDMA channel and the QDMA channel is enabled via the QDMA event enable register (QEER.E*n* = 1).
- EDMA3CC performs a link update on a PaRAM set address that is configured as a QDMA channel (matches QCHMAP*n* settings) and the corresponding channel is enabled via the QDMA event enable register (QEER.E*n* = 1).

Once a bit is set in QER, the EDMA3CC prioritizes and queues the event in the appropriate event queue. When the event reaches the head of the queue, it is evaluated for submission as a transfer request to the transfer controller.

As in the event-triggered transfers, if the PaRAM set associated with the channel is valid (it is not a null set) then the TR is submitted to the associated EDMA3TC and the channel can be triggered again.

If a bit is already set in QER (QER.E*n* = 1) and a second QDMA event for the same QDMA channel occurs prior to the original being cleared, the second QDMA event gets captured in the QDMA event miss register (QEMR.E*n* = 1).

17.2.4.3 Comparison Between DMA and QDMA Channels

The primary difference between DMA and QDMA channels is the event/channel synchronization. QDMA events are either autotriggered or link triggered. Autotriggering allows QDMA channels to be triggered by CPU(s) with a minimum number of linear writes to PaRAM. Link triggering allows a linked list of transfers to be executed, using a single QDMA PaRAM set and multiple link PaRAM sets.

A QDMA transfer is triggered when a CPU (or other EDMA3 programmer) writes to the trigger word of the QDMA channel parameter set (autotriggered) or when the EDMA3CC performs a link update on a PaRAM set that has been mapped to a QDMA channel (link triggered). Note that for CPU triggered (manually triggered) DMA channels, in addition to writing to the PaRAM set, it is required to write to the event set register (ESR) to kick-off the transfer.

QDMA channels are typically for cases where a single event will accomplish a complete transfer since the CPU (or EDMA3 programmer) must reprogram some portion of the QDMA PaRAM set in order to retrigger the channel. In other words, QDMA transfers are programmed with BCNT = CCNT = 1 for A-synchronized transfers, and CCNT = 1 for AB-synchronized transfers.

Additionally, since linking is also supported (if STATIC = 0 in OPT) for QDMA transfers, it allows you to initiate a linked list of QDMAs, so when EDMA3CC copies over a link PaRAM set (including the write to the trigger word), the current PaRAM set mapped to the QDMA channel will automatically be recognized as a valid QDMA event and initiate another set of transfers as specified by the linked set.

Copyright © 2013–2016, Texas Instruments Incorporated

17.2.5 Completion of a DMA Transfer

A parameter set for a given channel is complete when the required number of transfer requests is submitted (based on receiving the number of synchronization events). The expected number of TRs for a non-null/non-dummy transfer is shown in Table 17-4 for both synchronization types along with state of the PaRAM set prior to the final TR being submitted. When the counts (BCNT and/or CCNT) are this value, the next TR results in a:

- Final chaining or interrupt codes to be sent by the transfer controllers (instead of intermediate).
- Link updates (linking to either null or another valid link set).

Table 17-4. Expected Number of Transfers for Non-Null Transfer

Sync Mode	Counts at time 0	Total # Transfers	Counts prior to final TR
A-synchronized	ACNT BCNT CCNT	(BCNT × CCNT) TRs of ACNT bytes each	BCNT == 1 && CCNT == 1
AB-synchronized	ACNT BCNT CCNT	CCNT TRs for ACNT × BCNT bytes each	CCNT == 1

You must program the PaRAM OPT field with a specific transfer completion code (TCC) along with the other OPT fields (TCCHEN, TCINTEN, ITCCHEN, and ITCINTEN bits) to indicate whether the completion code is to be used for generating a chained event or/and for generating an interrupt upon completion of a transfer.

The specific TCC value (6-bit binary value) programmed dictates which of the 64-bits in the chain event register (CER[TCC]) and/or interrupt pending register (IPR[TCC]) is set.

See Section 17.2.9 for details on interrupts and Section 17.2.8 for details on chaining.

You can also selectively program whether the transfer controller sends back completion codes on completion of the final transfer request (TR) of a parameter set (TCCHEN or TCINTEN), for all but the final transfer request (TR) of a parameter set (ITCCHEN or ITCINTEN), or for all TRs of a parameter set (both). See Section 17.2.8 for details on chaining (intermediate/final chaining) and Section 17.2.9 for details on intermediate/final interrupt completion.

A completion detection interface exists between the EDMA3 channel controller and transfer controller(s). This interface sends back information from the transfer controller to the channel controller to indicate that a specific transfer is completed.

All DMA/QDMA PaRAM sets must also specify a link address value. For repetitive transfers such as ping-pong buffers, the link address value should point to another predefined PaRAM set. Alternatively, a nonrepetitive transfer should set the link address value to the null link value. The null link value is defined as FFFFh. See Section 17.2.3.7 for more details.

NOTE: Any incoming events that are mapped to a null PaRAM set results in an error condition. The error condition should be cleared before the corresponding channel is used again. See Section 17.2.3.5.

There are three ways the EDMA3CC gets updated/informed about a transfer completion: normal completion, early completion, and dummy/null completion. This applies to both chained events and completion interrupt generation.

Copyright © 2013–2016, Texas Instruments Incorporated

17.2.5.1 Normal Completion

In normal completion mode (TCCMODE = 0 in OPT), the transfer or sub-transfer is considered to be complete when the EDMA3 channel controller receives the completion codes from the EDMA3 transfer controller. In this mode, the completion code to the channel controller is posted by the transfer controller after it receives a signal from the destination peripheral. Normal completion is typically used to generate an interrupt to inform the CPU that a set of data is ready for processing.

17.2.5.2 Early Completion

In early completion mode (TCCMODE = 1 in OPT), the transfer is considered to be complete when the EDMA3 channel controller submits the transfer request (TR) to the EDMA3 transfer controller. In this mode, the channel controller generates the completion code internally. Early completion is typically useful for chaining, as it allows subsequent transfers to be chained-triggered while the previous transfer is still in progress within the transfer controller, maximizing the overall throughput of the set of the transfers.

17.2.5.3 Dummy or Null Completion

This is a variation of early completion. Dummy or null completion is associated with a dummy set (Section 17.2.3.4) or null set (Section 17.2.3.3). In both cases, the EDMA3 channel controller does not submit the associated transfer request to the EDMA3 transfer controller(s). However, if the set (dummy/null) has the OPT field programmed to return completion code (intermediate/final interrupt/chaining completion), then it will set the appropriate bits in the interrupt pending register (IPR) or chained event register (CER). The internal early completion path is used by the channel controller to return the completion codes internally (that is, EDMA3CC generates the completion code).

17.2.6 Event, Channel, and PaRAM Mapping

Most of the DMA channels are tied to a specific hardware peripheral event, thus allowing transfers to be triggered by events from device peripherals or external hardware. A DMA channel typically requests a data transfer when it receives its event (apart from manually-triggered, chain-triggered, and other transfers). The amount of data transferred per synchronization event depends on the channel's configuration (ACNT, BCNT, CCNT, etc.) and the synchronization type (A-synchronized or AB-synchronized).

The association of an event to a channel is fixed. Each of the DMA channels has one specific event associated with it. For the synchronization events associated with each of the programmable DMA channels, see your device-specific data manual to determine the event to channel mapping.

If in an application, a channel does not make use of the associated synchronization event or does not have an associated synchronization event (unused), that channel can be used for manually-triggered or chained-triggered transfers, for linking/reloading, or as a QDMA channel.

Copyright © 2013–2016, Texas Instruments Incorporated

17.2.6.1 DMA Channel to PaRAM Mapping

The mapping between the DMA channel numbers and the PaRAM sets is a fixed, one-to-one mapping (see Table 17-5). In other words, channel (event) 0 is mapped to PaRAM set 0, channel (event 1) is mapped to PaRAM set 1, etc. So, for example, in order to program a transfer for event number 3, DMA channel 3 is associated with PaRAM set number 3 and you need to program this PaRAM set for configuring transfers associated with event number 3. See your device-specific data manual for the addresses of the PaRAM set entries.

Table 17-5. EDMA3 DMA Channel to PaRAM Mapping

PaRAM Set Number	Mapping
PaRAM Set 0	DMA Channel 0/Reload/QDMA
PaRAM Set 1	DMA Channel 1/Reload/QDMA
PaRAM Set 2	DMA Channel 2/Reload/QDMA
PaRAM Set 3	DMA Channel 3/Reload/QDMA
PaRAM Set 4	DMA Channel 4/Reload/QDMA
PaRAM Set 5	DMA Channel 5/Reload/QDMA
PaRAM Set 6	DMA Channel 6/Reload/QDMA
PaRAM Set 7	DMA Channel 7/Reload/QDMA
PaRAM Set 8	DMA Channel 8/Reload/QDMA
PaRAM Set 9	DMA Channel 9/Reload/QDMA
PaRAM Set 10	DMA Channel 10/Reload/QDMA
PaRAM Set 11	DMA Channel 11/Reload/QDMA
PaRAM Set 12	DMA Channel 12/Reload/QDMA
PaRAM Set 13	DMA Channel 13/Reload/QDMA
PaRAM Set 14	DMA Channel 14/Reload/QDMA
PaRAM Set 15	DMA Channel 15/Reload/QDMA
PaRAM Set 16	DMA Channel 16/Reload/QDMA
...	...
PaRAM Set 30	DMA Channel 30/Reload/QDMA
PaRAM Set 31	DMA Channel 31/Reload/QDMA
PaRAM Set 32	Reload/QDMA
PaRAM Set 33	Reload/QDMA
...	...
PaRAM Set *n* - 2	Reload/QDMA
PaRAM Set *n* - 1	Reload/QDMA
PaRAM Set *n*	Reload/QDMA

Copyright © 2013–2016, Texas Instruments Incorporated

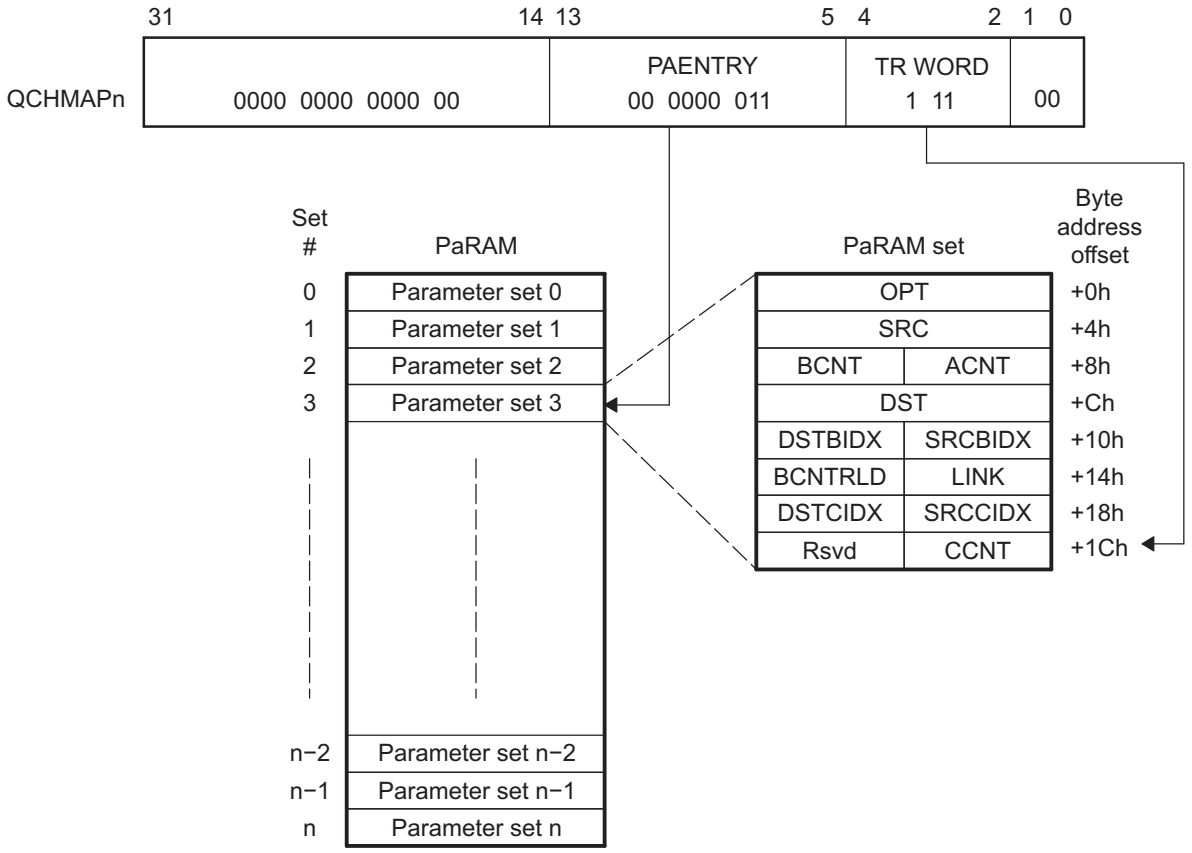
17.2.6.2 QDMA Channel to PaRAM Mapping

The mapping between the QDMA channels and the PaRAM sets is programmable .The QDMA channel *n* mapping register (QCHMAP*n*) in the EDMA3CC provides programmability for the QDMA channels to be mapped to any of the PaRAM sets in the PaRAM memory map. Figure 17-10 illustrates the use of QCHMAP.

Additionally, QCHMAP allows you to program the trigger word in the PaRAM set for the QDMA channel. A trigger word is one of the 8 words in the PaRAM set. For a QDMA transfer to occur, a valid TR synchronization event for EDMA3CC is a write to the trigger word in the PaRAM set pointed to by QCHMAP for a particular QDMA channel.

NOTE: By default, QDMA channels are mapped to PaRAM set 0. Care must be taken to appropriately remap PaRAM set 0 before it is used.

Figure 17-10. QDMA Channel to PaRAM Mapping



Note: *n* is the number of PaRAM sets supported in the EDMA3CC for a specific device.

Copyright © 2013–2016, Texas Instruments Incorporated

17.2.7 EDMA3 Channel Controller Regions

The EDMA3 channel controller (EDMA3CC) divides its address space into multiple regions. Individual channel resources can be exclusively assigned to a specific region, where each region is typically assigned to a specific EDMA programmer. This allows partitioning of EDMA channel (DMA/QDMA) resources in hetero- or multi-core devices, and devices where certain additional masters (for example, coprocessors) can also program/initiate EDMA3 transfers. The application software running on these cores/coprocessors can operate in these exclusive shadow region memory-maps, minimizing possibilities of resource conflicts.

17.2.7.1 Region Overview

The EDMA3CC memory-mapped registers are divided in three main categories:

1. Global registers
2. Global region channel registers
3. Shadow region channel registers

The global registers are located at a single/fixed location in the EDMA3CC memory map. These registers control EDMA3 resource mapping and provide debug visibility and error tracking information. See your device-specific data manual for the EDMA3CC memory map.

The channel registers (including DMA, QDMA, and interrupt registers) are accessible via the global channel region address range, or in the shadow *n* channel region address range(s). For example, the event enable register (EER) is visible in the global region register space at offset 1020h, or region addresses at offset 2020h for region 0 and at offset 2220h for region 1.

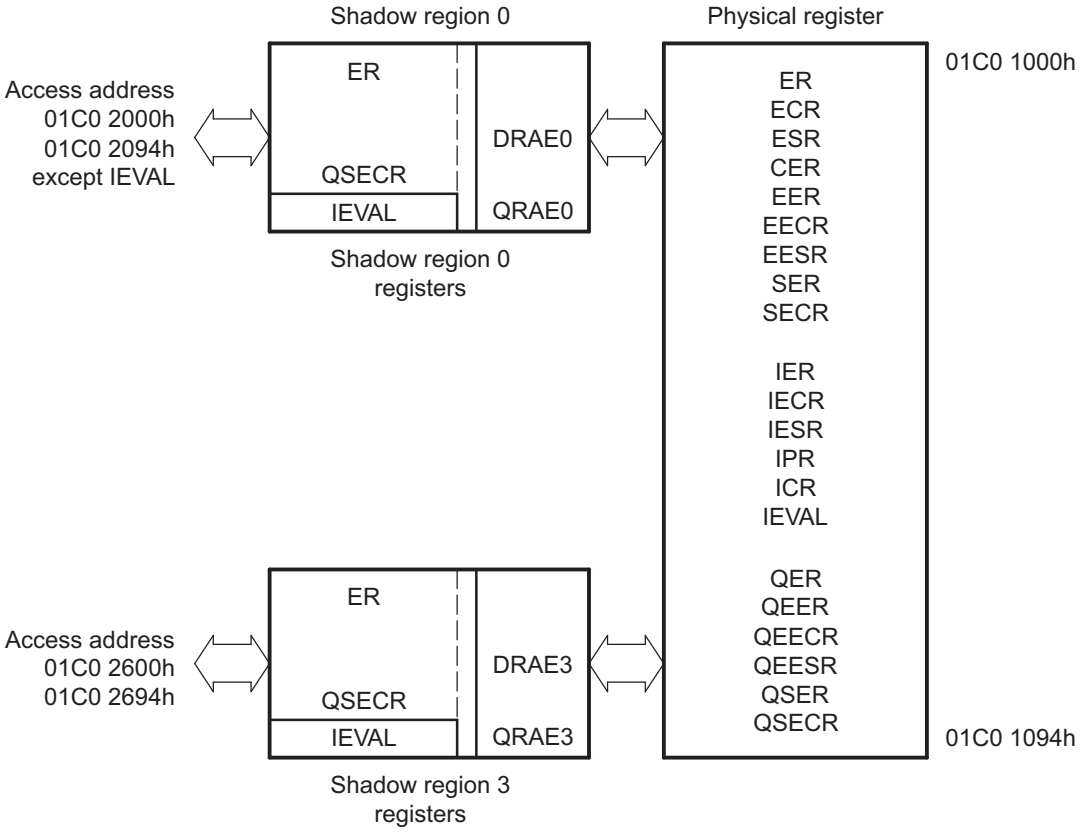
The underlying control register bits that are accessible via the shadow region address space (except for IEVAL*n*) are controlled by the DMA region access enable registers (DRAE*m*) and QDMA region access enable registers (QRAE*m*). Table 17-6 lists the registers in the shadow region memory-map. (See EDMA3CC memory-map figure for the complete global and shadow region memory-maps.) Figure 17-11 illustrates the conceptual view of the regions (where *n* is the number of shadow regions supported in the EDMA3CC for a specific device).

Table 17-6. Shadow Region Registers

DRAE*m*	**QRAE*m***
ER	QER
ECR	QEER
ESR	QEECR
CER	QEESR
EER	
EECR	
EESR	
SER	
SECR	
IER	
IECR	
IESR	
IPR	
ICR	
Register not affected by DRAE	
IEVAL	

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 17-11. Shadow Region Registers



17.2.7.2 Channel Controller Shadow Regions

For each EDMA3 shadow region (and associated memory-maps) there is a set of registers associated with the shadow region that allows association of the DMA/QDMA channels and interrupt completion codes to the region. These registers are user-programmed per region to assign ownership of the DMA/QDMA channels and TCC values to a region.

- DRAE*m*: One register exists for each of the shadow regions. The number of bits in each register matches the number of DMA channels. These registers need to be programmed to assign ownership of DMA channels to the respective region. Accesses to DMA event registers and interrupt registers via the shadow region address map are filtered through DRAE. A value of 1 in the corresponding DRAE bit implies that the corresponding DMA/interrupt channel is accessible; a value of 0 in the corresponding DRAE bit forces writes to be discarded and returns a value of 0 for reads.
- QRAE*m*: One register exists for every region. The number of bits in each register matches the number of QDMA channels. These registers must be programmed to assign ownership of QDMA channels to the respective region. To enable a channel in a shadow region using shadow region 0 QEER, the respective bit in QRAE must be set or writing into QEESR will not have the desired effect.

It is typical for an application to have a unique assignment of QDMA/DMA channels (and, therefore, a given bit position) to a given region.

The use of shadow regions allows for restricted access to EDMA3 resources (DMA channels, QDMA channels, TCC, interrupts) by tasks/cores/EDMA3 programmers in a system by setting or clearing bits in the DRAE/QRAE registers. If exclusive access to any given channel/TCC code is required for a region, then only that region's DRAE/QRAE should have the associated bit set.

Copyright © 2013–2016, Texas Instruments Incorporated

Additionally, with each shadow region, there is an associated shadow region completion interrupt (EDMA3CC_INT*n* where *n* denotes the shadow region number). For multi-core/hetero-core devices, the various shadow region interrupts might be tied to the interrupt controllers for different cores. For single core devices, all shadow region interrupts would be routed to the device interrupt controller. See your device-specific data manual for the shadow region interrupt hookup to the device interrupt controller(s). The DRAE associated with each shadow region acts as a secondary interrupt enable (along with the interrupt enable register) for the respective shadow region interrupts. See Section 17.2.9 for more information on interrupts.

Example 17-1. Resource Pool Division Across Two Regions

This example illustrates a resource pool division across two regions, assuming region 0 must be allocated 16 DMA channels (0-15) and 1 QDMA channel (0), and 16 TCC codes (0-15). Region 1 needs to be allocated 16 DMA channels (16-31) and 7 QDMA channels (1-7), and 16 TCC codes (16-31). DRAE should be equal to the OR of the bits that are required for the DMA channels and the TCC codes:

```
Region 0: DRAE = 0x0000FFFF QRAE = 0x00000001 Region 1: DRAE = 0xFFFF0000 QRAE = 0x000000FE
```

17.2.8 Chaining EDMA3 Channels

The channel chaining capability for the EDMA3 allows the completion of an EDMA3 channel transfer to trigger another EDMA3 channel transfer. The purpose is to allow you the ability to chain several events through one event occurrence.

Chaining is different from linking (Section 17.2.3.7). The EDMA3 link feature reloads the current channel parameter set with the linked parameter set. The EDMA3 chaining feature does not modify or update any channel parameter set; it provides a synchronization event to the chained channel (see Section 17.2.4.1.3 for chain-triggered transfer requests).

Chaining is achieved at either final transfer completion or intermediate transfer completion, or both, of the current channel. Consider a channel *m* (DMA/QDMA) required to chain to channel *n*. Channel number *n* (0-31) needs to be programmed into the TCC field of channel *m* channel options parameter (OPT) set.

- If final transfer completion chaining (TCCHEN = 1 and ITCCHEN = 0 in channel *m* OPT) is enabled, the chain-triggered event occurs after the *last* transfer request of channel *m* is submitted (early completion) or completed (normal completion).
- If intermediate transfer completion chaining (TCCHEN = 0 and ITCCHEN = 0 in channel *m* OPT) is enabled, the chain-triggered event occurs after every *intermediate* transfer request of channel *m* is submitted (early completion) or completed (normal completion).
- If both final and intermediate transfer completion chaining (TCCHEN = 1 and ITCCHEN = 1 in channel *m* OPT) are enabled, the chain-trigger event occurs after *every* transfer request of channel *m* is submitted (early completion) or completed (normal completion).

Table 17-7 shows the number of chain event triggers occurring in different synchronized scenarios. Consider channel 31 programmed with ACNT = 3, BCNT = 4, CCNT = 5, and TCC = 30.

Table 17-7. Chain Event Triggers

	(Number of chained event triggers on channel 30)	
Options	**A-Synchronized**	**AB-Synchronized**
TCCHEN = 1, ITCCHEN = 0	1 (Last TR)	1 (Last TR)
TCCHEN = 0, ITCCHEN = 1	19 (All but the last TR)	4 (All but the last TR)
TCCHEN = 1, ITCCHEN = 1	20 (All TRs)	5 (All TRs)

17.2.9 EDMA3 Interrupts

The EDMA3 interrupts are divided into 2 categories:

- Transfer completion interrupts

Copyright © 2013–2016, Texas Instruments Incorporated

- Error interrupts

The transfer completion interrupts are listed in Table 17-8. The error interrupts are listed in Table 17-9.

Table 17-8. EDMA3 Transfer Completion Interrupts

Name	Description	AINTC
EDMA3_0_CC0_INT0	EDMA3_0 Channel Controller 0 Shadow Region 0 Transfer Completion Interrupt	11
EDMA3_0_CC0_INT1	EDMA3_0 Channel Controller 0 Shadow Region 1 Transfer Completion Interrupt	—
EDMA3_0_CC0_INT2	EDMA3_0 Channel Controller 0 Shadow Region 2 Transfer Completion Interrupt	—
EDMA3_0_CC0_INT3	EDMA3_0 Channel Controller 0 Shadow Region 3 Transfer Completion Interrupt	—
EDMA3_1_CC0_INT0	EDMA3_1 Channel Controller 0 Shadow Region 0 Transfer Completion Interrupt	93
EDMA3_1_CC0_INT1	EDMA3_1 Channel Controller 0 Shadow Region 1 Transfer Completion Interrupt	—
EDMA3_1_CC0_INT2	EDMA3_1 Channel Controller 0 Shadow Region 2 Transfer Completion Interrupt	—
EDMA3_1_CC0_INT3	EDMA3_1 Channel Controller 0 Shadow Region 3 Transfer Completion Interrupt	—

Table 17-9. EDMA3 Error Interrupts

Name	Description	AINTC
EDMA3_0_CC0_ERRINT	EDMA3_0 Channel Controller 0 Error Interrupt	12
EDMA3_0_TC0_ERRINT	EDMA3_0 Transfer Controller 0 Error Interrupt	13
EDMA3_0_TC1_ERRINT	EDMA3_0 Transfer Controller 1 Error Interrupt	32
EDMA3_1_CC0_ERRINT	EDMA3_1 Channel Controller 0 Error Interrupt	94
EDMA3_1_TC0_ERRINT	EDMA3_1 Transfer Controller 0 Error Interrupt	95

17.2.9.1 Transfer Completion Interrupts

The EDMA3CC is responsible for generating transfer completion interrupts to the CPU. The EDMA3 generates a single completion interrupt per shadow region on behalf of all DMA/QDMA channels. Various control registers and bit fields facilitate EDMA3 interrupt generation.

The transfer completion code (TCC) value is directly mapped to the bits of the interrupt pending register (IPR), as shown in Table 17-10. For example, if TCC = 00 0000b, IPR[0] is set after transfer completion, and results in an interrupt generation to the CPU if in the EDMA3CC and device interrupt controller are configured to allow a CPU interrupt. See Section 17.2.9.1.1 for details on enabling EDMA3 transfer completion interrupts.

When a completion code is returned (as a result of early or normal completion), the corresponding bit in IPR is set. For the completion code to be returned, the PaRAM set associated with the transfer must enable the transfer completion interrupt (final/intermediate) in the channel options parameter (OPT).

The transfer completion code (TCC) can be programmed to any value for a DMA/QDMA channel. There does not need to be a direct relation between the channel number and the transfer completion code value. This allows multiple channels having the same transfer completion code value to cause a CPU to execute the same interrupt service routine (ISR) for different channels.

NOTE: The TCC field in the channel options parameter (OPT) is a 6-bit field and can be programmed for any value between 0-64. For devices with 32 DMA channels, the TCC should have a value between 0 to 31 so that it sets the appropriate bits (0 to 31) in IPR (and can interrupt the CPU(s) on enabling the IER register bits (0-31)).

Copyright © 2013–2016, Texas Instruments Incorporated

Table 17-10. Transfer Complete Code (TCC) to EDMA3CC Interrupt Mapping

TCC Bits in OPT (TCINTEN/ITCINTEN = 1)	IPR Bit Set
00 0000b	IPR0
00 0001b	IPR1
00 0010b	IPR2
00 0011b	IPR3
00 0100b	IPR4
…	…
…	…
01 1110b	IPR30
01 1111b	IPR31

You can enable interrupt generation at either final transfer completion or intermediate transfer completion, or both. Consider channel *m* as an example.

- If the final transfer interrupt (TCINTEN = 1 and ITCINTEN = 0 in OPT) is enabled, the interrupt occurs after the *last* transfer request of channel *m* is either submitted or completed (depending on early or normal completion).
- If the intermediate transfer interrupt (TCINTEN = 0 and ITCINTEN = 1 in OPT) is enabled, the interrupt occurs after every *intermediate* transfer request of channel *m* is either submitted or completed (depending on early or normal completion).
- If both final and intermediate transfer completion interrupts (TCINTEN = 1 and ITCINTEN = 1 in OPT) are enabled, the interrupt occurs after *every* transfer request of channel *m* is submitted or completed (depending on early or normal completion).

Table 17-11 shows the number of interrupts occurring in different synchronized scenarios. Consider channel 31 programmed with ACNT = 3, BCNT = 4, CCNT = 5, and TCC = 30.

Table 17-11. Number of Interrupts

Options	A-Synchronized	AB-Synchronized
TCINTEN = 1, ITCINTEN = 0	1 (Last TR)	1 (Last TR)
TCINTEN = 0, ITCINTEN = 1	19 (All but the last TR)	4 (All but the last TR)
TCINTEN = 1, ITCINTEN = 1	20 (All TRs)	5 (All TRs)

17.2.9.1.1 Enabling Transfer Completion Interrupts

For the EDMA3 channel controller to assert a transfer completion to the external world, the interrupts have to be enabled in the EDMA3CC. This is in addition to setting up the TCINTEN and ITCINTEN bits in OPT of the associated PaRAM set.

The EDMA3 channel controller has interrupt enable registers (IER) and each bit location in IER serves as a primary enable for the corresponding interrupt pending register (IPR).

All the interrupt registers (IER, IESR, IECR, and IPR) are either manipulated from the global DMA channel region or by way of the DMA channel shadow regions. The shadow regions provide a view to the same set of physical registers that are in the global region.

The EDMA3 channel controller has a hierarchical completion interrupt scheme that makes use of a single set of interrupt pending register (IPR) and single set of interrupt enable registers (IER). A second level of interrupt masking is provided by the programmable DMA region access enable registers (DRAE). See Figure 17-12.

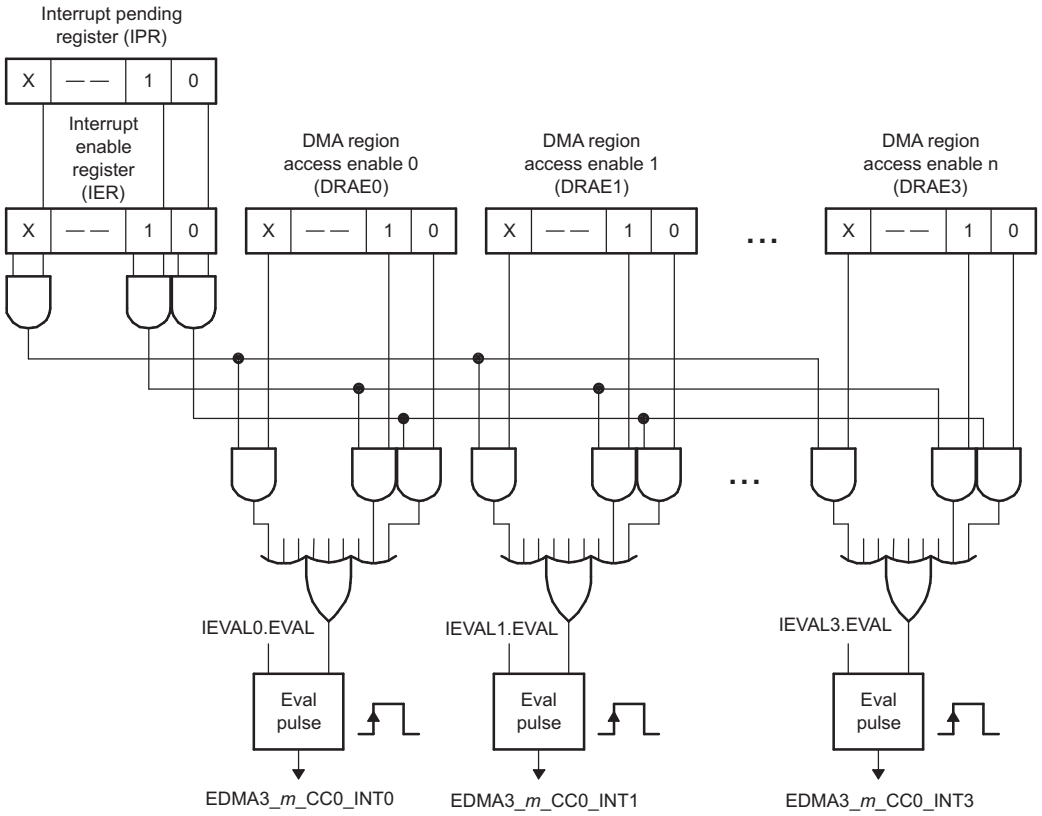
Copyright © 2013–2016, Texas Instruments Incorporated

For the EDMA3CC to generate the transfer completion interrupts that are associated with each shadow region, the following conditions must be true:

- EDMA3CC_INT0: (IPR.E0 & IER.E0 & DRAE0.E0) | (IPR.E1 & IER.E1 & DRAE0.E1) | ...| (IPR.E*n* & IER.E*n* & DRAE0.E*n*)
- EDMA3CC_INT1: (IPR.E0 & IER.E0 & DRAE1.E0) | (IPR.E1 & IER.E1 & DRAE1.E1) | ...| (IPR.E*n* & IER.E*n* & DRAE1.E*n*)

  where *n* is the number of shadow regions supported in the EDMA3CC for a specific device.

Figure 17-12. Interrupt Diagram



NOTE: The DRAE for all regions is expected to be set up at system initialization and to remain static for an extended period of time. The interrupt enable registers should be used for dynamic enable/disable of individual interrupts.

Because there is no relation between the TCC value and the DMA/QDMA channel, it is possible, for example, for DMA channel 0 to have the OPT.TCC = 31 in its associated PaRAM set. This would mean that if a transfer completion interrupt is enabled (OPT.TCINTEN or OPT.ITCINTEN is set), then based on the TCC value, IPR.E31 is set up on completion. For proper channel operations and interrupt generation using the shadow region map, you must program the DRAE that is associated with the shadow region to have read/write access to both bit 0 (corresponding to channel 0) and bit 31 (corresponding to IPR.E31 bit that is set upon completion).

17.2.9.1.2 Clearing Transfer Completion Interrupts

Transfer completion interrupts that are latched to the interrupt pending register (IPR) is cleared by writing a 1 to the corresponding bit in the interrupt pending clear register (ICR). For example, a write of 1 to ICR.E0 clears a pending interrupt in IPR.E0.

Copyright © 2013–2016, Texas Instruments Incorporated

If an incoming transfer completion code (TCC) gets latched to a bit in IPR, then additional bits that get set due to a subsequent transfer completion will not result in asserting the EDMA3CC completion interrupt. In order for the completion interrupt to be pulsed, the required transition is from a state where no enabled interrupts are set to a state where at least one enabled interrupt is set.

17.2.9.2 EDMA3 Interrupt Servicing

On completion of a transfer (early or normal completion), the EDMA3 channel controller sets the appropriate bit in the interrupt pending register (IPR) as specified by the transfer completion codes. If the completion interrupts are appropriately enabled, then the CPU enters the interrupt service routine (ISR) when the completion interrupt is asserted. Since there is a single completion interrupt for all DMA/QDMA channels.

After servicing the interrupt, the ISR should clear the corresponding bit in IPR; therefore, enabling recognition of future interrupts. Only when all IPR bits are cleared, the EDMA3CC will assert additional completion interrupts.

It is possible that when one interrupt is serviced; many other transfer completions result in additional bits being set in IPR, thereby resulting in additional interrupts. It is likely that each of these bits in IPR would need different types of service; therefore, the ISR must check all pending interrupts and continue until all the posted interrupts are appropriately serviced.

Following are examples (pseudo code) for a CPU interrupt service routine for an EDMA3CC completion interrupt.

The ISR routine in Example 17-2 is more exhaustive and incurs a higher latency.

Example 17-2. Interrupt Servicing

The pseudo code:

1. Read the interrupt pending register (IPR).
2. Perform the operations needed.
3. Write to the interrupt pending clear register (ICR) to clear the corresponding IPR bit.
4. Read IPR again:

   (a) If IPR is not equal to 0, repeat from step 2 (implies occurrence of new event between step 2 to step 4).

   (b) If IPR is equal to 0, this should assure you that all enabled interrupts are inactive.

NOTE: It is possible that during step 4, an event occurs while the IPR bits are read to be 0 and the application is still in the interrupt service routine. If this happens, a new interrupt is recorded in the device interrupt controller and a new interrupt is generated as soon as the application exits the interrupt service routine.

Copyright © 2013–2016, Texas Instruments Incorporated

Example 17-3 is less rigorous, with less burden on the software in polling for set interrupt bits, but can occasionally cause a race condition, as mentioned above.

Example 17-3. Interrupt Servicing

If it is desired to leave any enabled and pending (possibly lower priority) interrupts, it is required to force the interrupt logic to reassert the interrupt pulse by setting the EVAL bit in the interrupt evaluation register (IEVAL).

The pseudo code:

1. Enter ISR.
2. Read IPR.
3. For the condition set in IPR that you desire to service:
   (a) Service interrupt as required by application.
   (b) Clear bit for serviced conditions (others may still be set, and other transfers may have resulted in returning the TCC to EDMA3CC after step 2).
4. Read IPR prior to exiting ISR:
   (a) If IPR is equal to 0, then exit ISR.
   (b) If IPR is not equal to 0, then set IEVAL so that upon exit of ISR, a new interrupt is triggered if any enabled interrupts are still pending.

The EVAL bit must not be set when IPR is read to be 0, to avoid generation of extra interrupt pulses.

NOTE: Since the DMA region access registers (DRAE) are required to enable the transfer completion region interrupts, it is assumed that there will be a unique and nonoverlapping (in most cases) assignment of the channels and interrupts among the different shadow regions. This allows the interrupt registers (IER, IESR, IECR, IPR, and ICR) in the different shadow regions to functionally operate in an independent manner and nonoverlapping. The above examples for the interrupt service routine is based on this assumption.

17.2.9.3 Interrupt Evaluation Operations

The EDMA3CC has interrupt evaluate registers (IEVAL) in each shadow region. These registers are the only registers in the DMA channel shadow region memory map that are not affected by the settings for the DMA region access enable registers (DRAE). A write of 1 to the EVAL bit in these registers associated with a particular shadow region results in pulsing the associated region interrupt, if any enabled interrupt (via IER) is still pending (IPR). This register can be used in order to assure that the interrupts are not missed by the CPU (or the EDMA3 master associated with the shadow region) if the software architecture chooses not to use all interrupts. See Example 17-3 for the use of IEVAL in the EDMA3 interrupt service routine (ISR).

Similarly an error evaluate register (EEVAL) exists in the global region. A write of 1 to the EVAL bit in EEVAL causes the pulsing of the error interrupt if any pending errors are in EMR, QEMR, or CCERR. See Section 17.2.9.4 for additional details on error interrupts.

NOTE: While using IEVAL for shadow region completion interrupts, you should make sure that the IEVAL operated upon is from that particular shadow region memory map.

Copyright © 2013–2016, Texas Instruments Incorporated

17.2.9.4 Error Interrupts

The EDMA3CC error registers provide the capability to differentiate error conditions (event missed, threshold exceed, etc.). Additionally, if the error bits are set in these registers, it results in asserting the EDMA3CC error interrupt. If EDMA3CC error interrupt is enabled in the device interrupt controller, then it allows the CPU to handle the error conditions.

The EDMA3CC has a single error interrupt (EDMA3_*m*_CC0_ERRINT) that gets asserted for all EDMA3CC error conditions. There are four conditions that cause the error interrupt to be pulsed:

- DMA missed events: for all 32 DMA channels. These get latched in the event missed registers (EMR).
- QDMA missed events: for all QDMA channels. These get latched in the QDMA event missed register (QEMR).
- Threshold exceed: for all event queues. These get latched in EDMA3CC error register (CCERR).
- TCC error: for outstanding transfer requests expected to return completion code (TCCHEN or TCINTEN bit in OPT is set to 1) exceeding the maximum limit of 31. This also gets latched in the EDMA3CC error register (CCERR).

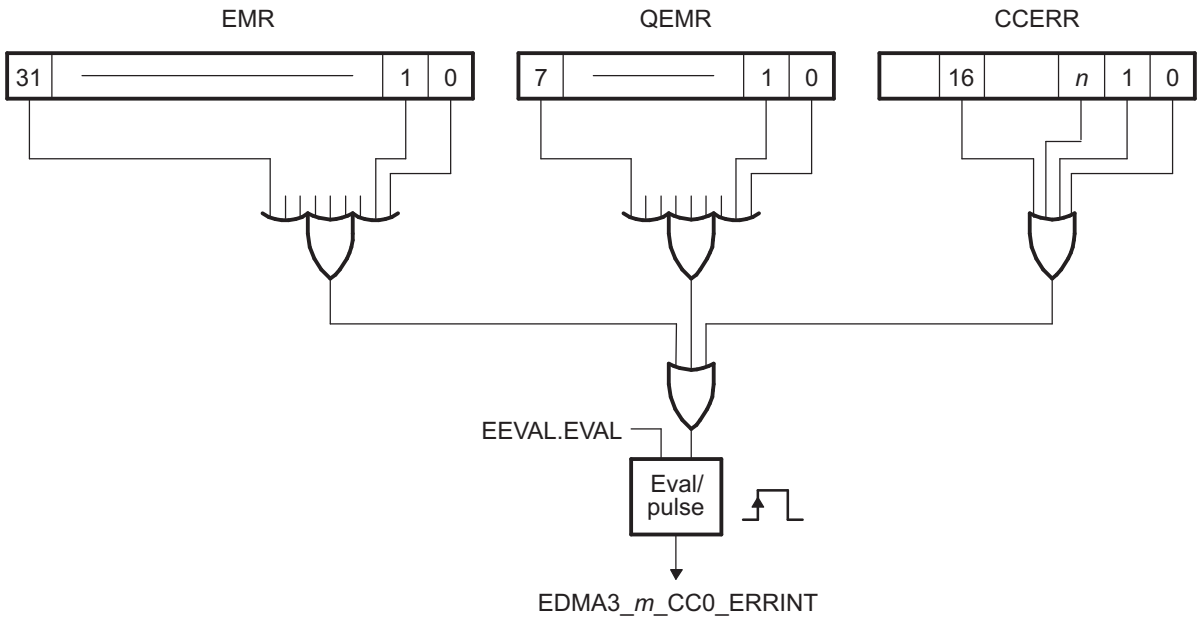
Figure 17-13 illustrates the EDMA3CC error interrupt generation operation.

If any of the bits are set in the error registers due to any error condition, the (EDMA3_*m*_CC0_ERRINT) always is asserted, as there are no enables for masking these error events. Similar to the transfer completion interrupts, the error interrupt also is pulsed only when the error interrupt condition transitions from a state where no errors are set to a state where at least one error bit is set. If additional error events are latched prior to the original error bits being cleared, the EDMA3CC does not generate additional interrupt pulses.

To reduce the burden on the software, similar to the interrupt evaluate register (IEVAL), there is an error evaluate register (EEVAL) that allows reevaluation of pending set error events/bits. This can be used so that the CPU(s) does not miss any error events.

NOTE: It is a good practice to have the error interrupt enabled in the device interrupt controller and associate an interrupt service routine with it to address the various error conditions appropriately. This puts less burden on software (polling for error status) and additionally provides a good debug mechanism for unexpected error conditions.

Figure 17-13. Error Interrupt Operation



Note: *n* is the number of queues supported in the EDMA3CC for a specific device.

Copyright © 2013–2016, Texas Instruments Incorporated

17.2.10 Event Queue(s)

Event queues are a part of the EDMA3 channel controller. Event queues form the interface between the event detection logic in the EDMA3CC and the transfer request (TR) submission logic of the EDMA3CC. Each queue is 16 entries deep, that is, a maximum of 16 queued events per event queue. If there are more than 16 events, then the events that cannot find a place in the event queue remain set in the associated event register.

The number of event queues in the EDMA3CC determines the number of transfer controllers connected to the EDMA3CC. By default, there is a one-to-one mapping between the queues and transfer controllers. Therefore, the transfer requests (TRs) associated with events in Q0 get submitted to TC0. Similarly, transfer requests associated with events in Q1 get submitted to TC1, and so on.

An event that wins prioritization against other DMA and/or QDMA pending events is placed at the end of the appropriate event queue. Each event queue is serviced in a FIFO (first in–first out) order. Once the event reaches the head of its queue and the corresponding transfer controller is ready to receive another TR, the event is dequeued and the PaRAM set corresponding to the dequeued event is processed and submitted as a transfer request packet (TRP) to the associated EDMA3 transfer controller.

A lower numbered queue has a higher dequeuing priority then a higher numbered queue. For example, Q0 has higher priority than Q1, if Q0 and Q1 both have at least one event entry and if both TC0 and TC1 can accept transfer requests, then the event in Q0 is dequeued first and its associated PaRAM set is processed and submitted as a transfer request (TR) to TC0.

All the event entries in all the event queues are software readable (not writeable) by accessing the event queue entry registers (Q*x*E*y*). Each event entry register characterizes the queued event in terms of the type of event (manual, event, chained or autotriggered) and the event number. See Section 17.4.2.4.1 for a description of the bit fields in the queue event entry registers.

17.2.10.1 DMA/QDMA Channel to Event Queue Mapping

Each DMA channel and QDMA channel is independently programmed to map to a specific queue using the DMA queue number register *n* (DMAQNUM*n*) and the QDMA channel queue number register (QDMANUM). The mapping of DMA/QDMA channels is critical to achieving the desired performance level for the EDMA and most importantly in meeting real-time deadlines.

NOTE: If an event is ready to be queued and both the event queue and the EDMA3 transfer controller associated to the event queue are empty, then the event bypasses the event queue, and goes to the PaRAM processing logic and eventually to the transfer request submission logic for submission to the EDMA3TC. In this case, the event is not logged in the event queue status registers.

Copyright © 2013–2016, Texas Instruments Incorporated

17.2.10.2 Queue RAM Debug Visibility

Each event queue has 16 entries. These 16 entries are managed in a circular FIFO manner. All event queue entries for all event queues are software readable by the event queue entry register (Q*x*E*x*). Additionally, for each queue there is a queue status register (QSTAT*n*).

These registers provide user visibility and may be helpful while debugging real-time issues (typically post-mortem), involving multiple events and event sources. The event queue entry register (Q*x*E*x*) uniquely identifies the specific event type (event-triggered, manually-triggered, chain-triggered, and QDMA events) along with the event number (for DMA/QDMA channels) that are in the queue or have been de-queued (passed through the queue). QSTAT*n* includes fields for the start pointer (STRTPTR) that provides the offset to the head entry of an event. It also includes a NUMVAL field that provides the total number of valid entries residing in the event queue at a given instance of time. The STRTPTR field may be used to index appropriately into the 16 event entries. The NUMVAL number of entries starting from STRTPTR are indicative of events still queued in the respective queue. The remaining entries may be read to determine which events have already been de-queued and submitted to the associated transfer controller.

17.2.10.3 Queue Resource Tracking

The EDMA3CC event queue includes watermarking/threshold logic that allows you to keep track of maximum usage of all event queues. This is useful for debugging real-time deadline violations that may result from head-of-line blocking on a given EDMA3 event queue.

You can program the maximum number of events that can queue up in an event queue by programming the threshold value (between 0 to 15) in the queue watermark threshold A register (QWMTHRA). The maximum queue usage is recorded actively in the watermark (WM) field of the queue status register (QSTAT*n*) that keeps getting updated based on a comparison of number of valid entries, which is also visible in the NUMVAL bit in QSTAT*n* and the maximum number of entries (WM bit in QSTAT*n*).

If the queue usage is exceeded, this status is visible in the EDMA3CC registers: the QTHRXCD*n* bit in the channel controller error register (CCERR) and the THRXCD bit in QSTAT*n*, where *n* stands for the event queue number. Any bits that are set in CCERR also generate an EDMA3CC error interrupt.

Copyright © 2013–2016, Texas Instruments Incorporated

17.2.11 EDMA3 Transfer Controller (EDMA3TC)

The EDMA3 channel controller is the user-interface of the EDMA3 and the EDMA3 transfer controller (EDMA3TC) is the data movement engine of the EDMA3. The EDMA3CC submits transfer requests (TR) to the EDMA3TC and the EDMA3TC performs the data transfers dictated by the TR.

17.2.11.1 Architecture Details

17.2.11.1.1 EDMA3TC Configuration

Each transfer controller on a device is designed differently based on considerations like performance requirements, system topology (main SCR bus width, external memory bus width), gate count, etc. The parameters that determine the TC configurations are:

- FIFOSIZE: Determines the size in bytes for the Data FIFO that is the temporary buffer for the in-flight data. The data FIFO is where the read return data read by the TC read controller from the source endpoint is stored and subsequently written out to the destination endpoint by the TC write controller.
- Default Burst Size (DBS): The DBS is the maximum number of bytes per read/write command issued by a transfer controller.
- BUSWIDTH: The width of the read and write data buses in bytes, for the TC read and write controller, respectively. This is typically equal to the bus width of the main SCR interface.
- DSTREGDEPTH: This determines the number of Destination FIFO register set. The number of Destination FIFO register set for a transfer controller, determines the maximum number of outstanding transfer requests (TR pipelining).

Of the four parameters, the FIFOSIZE, BUSWIDTH, and DSTREGDEPTH values are fixed in design for a given device. The default burst size (DBS) for EDMA3_0_TC0 and EDMA3_0_TC1 is configurable by the chip configuration 0 register (CFGCHIP0) in the System Configuration Module and for EDMA3_1_TC0 is configurable by the chip configuration 1 register (CFGCHIP1) in the System Configuration Module. Table 17-12 provides the configuration of the individual EDMA3 transfer controllers on the device.

The burst size for each transfer controlled can be programmed to be 16-, 32-, or 64-bytes. The default values for DBS are typically chosen for optimal performance in most intended-use conditions; therefore, if you decide to use a value other then the default, you should evaluate the impact on performance. Depending on the FIFOSIZE and source/destination locations the performance for the transfer can vary significantly for different burst size values.

NOTE: It is expected that the DBS value for a transfer controller is static and should be based on the application requirement. It is not recommended that the DBS value be changed on-the-fly.

Table 17-12. EDMA3 Transfer Controller Configurations

Parameter	EDMA3_0_TC0	EDMA3_0_TC1	EDMA3_1_TC0
FIFOSIZE	128 bytes	128 bytes	256 bytes
BUSWIDTH	8 bytes (64 bits)	8 bytes (64 bits)	8 bytes (64 bits)
DSTREGDEPTH	4 entries	4 entries	4 entries
DBS (default)	16 bytes	16 bytes	16 bytes
Error interrupt	EDMA3_0_TC0_ERRINT	EDMA3_0_TC1_ERRINT	EDMA3_1_TC0_ERRINT
EDMA3 channel controller used	EDMA3_0_CC0	EDMA3_0_CC0	EDMA3_1_CC0

Copyright © 2013–2016, Texas Instruments Incorporated

17.2.11.1.2 Command Fragmentation

The TC read and write controllers in conjunction with the source and destination register sets are responsible for issuing optimally-sized reads and writes to the slave endpoints. The transfer controller read/write transaction as specified by the transfer request packet is internally broken down into smaller bursts; this determines the default burst size (DBS) for the transfer controller. See Section 17.2.11.1.1 for the DBS value of each EDMA3TC.

The EDMA3TC attempts to issue the largest possible command size as limited by the DBS value or the ACNT/BCNT value of the TR. EDMA3TC obeys the following rules:

- The read/write controllers always issue commands less than or equal to the DBS value.
- The first command of a 1D transfer is always issued so that subsequent commands align to the DBS value.

Example 17-4 shows the command fragmentation for a DBS of 32 bytes. In summary, if the ACNT value is larger than the DBS value, then the EDMA3TC breaks the ACNT array into DBS-sized commands to the source/destination addresses. Each BCNT number of arrays are then serviced in succession.

Example 17-4. Command Fragmentation (DBS = 32)

The pseudo code:

1. ACNT = 8, BCNT = 8, SRCBIDX = 8, DSTBIDX = 10, SRCADDR = 64, DSTADDR = 191

   Read Controller: This is optimized from a 2D-transfer to a 1D-transfer such that the read side is equivalent to ACNT = 64, BCNT = 1.

   Cmd0 = 32 byte, Cmd0 = 32 byte

   Write Controller: Since DSTBIDX != ACNT, it is not optimized.

   Cmd0 = 8 byte, Cmd1 = 8 byte, Cmd2 = 8 byte, Cmd3 = 8 byte, Cmd4 = 8 byte, Cmd5 = 8 byte, Cmd6 = 8 byte, Cmd7 = 8 byte.

2. ACNT = 64, BCNT = 1, SRCADDR = 31, DSTADDR = 513

   Read Controller: Read address is not aligned.

   Cmd0 = 1 byte, (now the SRCADDR is aligned to 32 for the next command)

   Cmd1 = 32 bytes

   Cmd2 = 31 bytes

   Write Controller: The write address is also not aligned.

   Cmd0 = 31 bytes, (now the DSTADDR is aligned to 32 for the next command)

   Cmd1 = 32 bytes

   Cmd2 = 1 byte

17.2.11.1.3 TR Pipelining and Data Ordering

The transfer controller(s) can issue back-to-back transfer requests (TR). The number of outstanding TRs for a TC is limited by the number of destination FIFO register entries that is controlled by the DSTREGDEPTH parameter (fixed in design for a given transfer controller). TR pipelining refers to the ability of the TC read controller to issue read commands for a subsequent TR, while the TC write controller is still performing writes for the previous TR. Consider the case of 2 TRs (TR0 followed by TR1), because of TR pipelining, the TC read controller can start issuing the read commands for TR1 as soon as the last read command for TR0 has been issued, meanwhile the write commands and write data for TR0 are tracked by the destination FIFO registers. In summary, the TC read controller is able to process *n* TRs ahead of the write controller, where *n* is the number of destination FIFO register entries (typically 4).

TR pipelining is useful for maintaining throughput on back-to-back small TRs. It eliminates the read overhead because reads start in the background of a previous TR writes.

Copyright © 2013–2016, Texas Instruments Incorporated

It should be noted that back-to-back TRs are targeted to different end points even though the read return data for the two TRs might get returned out of order (that is, read data for TR1 might come in before read data for TR0), the transfer controller issues that the write commands are issued in order (that is, write commands for TR0 will be issued before write commands for TR1).

17.2.11.2 Error Generation

Similar to the channel controller, the transfer controllers are capable of detecting and reporting several error conditions. The TC errors are generated, under three main conditions:

- BUSERR: The TC read or write controllers detect an error signaled by the source or destination address. The additional details on the type of error is also recorded in the ERRDET register, which indicates whether it is a read error (source address errors) or write error (destination address error).
- MMRAERR: CPU accesses illegal/reserved addresses in the EDMA3CC/TC memory-map.
- TRERR: A transfer request packet is detected to be violating the constant addressing mode transfer rules (the source/destination addresses and source/destination indexes must be aligned to 32 bytes).

You can poll for the errors, as the status of the errors can be read from the ERRSTAT registers, additionally if the error bits are enabled in the ERREN register, a bit set in the ERRSTAT will cause the error condition to interrupt the CPU(s). You can decide to enable/disable either or all error types.

17.2.11.3 Debug Features

The DMA program register set, DMA source active register set, and the destination FIFO register set are used to derive a brief history of TRs serviced through the transfer controller.

Additionally, the EDMA3TC status register (TCSTAT) has dedicated bit fields to indicate the ongoing activity within different parts of the transfer controller:

- The SRCACTV bit indicates whether the source active set is active.
- The DSTACTV bit indicates the number of TRs resident in the destination register active set at a given instance.
- The PROGBUSY bit indicates whether a valid TR is present in the DMA program set.

If the TRs are in progression, caution must be used and you must realize that there is a chance that the values read from the EDMA3TC status registers will be inconsistent since the EDMA3TC may change the values of these registers due to ongoing activities.

It is recommended that you ensure no additional submission of TRs to the EDMA3TC in order to facilitate ease of debug.

17.2.11.3.1 Destination FIFO Register Pointer

The destination FIFO register pointer is implemented as a circular buffer with the start pointer being DFSTRTPTR and a buffer depth of usually 2 or 4. The EDMA3TC maintains two important status details in TCSTAT that may be used during advanced debugging, if necessary. The DFSTRTPTR is a start pointer, that is, the index to the head of the destination FIFO register. The DSTACTV is a counter for the number of valid (occupied) entries. These registers may be used to get a brief history of transfers.

Examples of some register field values and their interpretation:

- DFSTRTPTR = 0 and DSTACTV = 0 implies that no TRs are stored in the destination FIFO register.
- DFSTRTPTR = 1 and DSTACTV = 2h implies that two TRs are present. The first pending TR is read from the destination FIFO register entry 1 and the second pending TR is read from the destination FIFO register entry 2.
- DFSTRTPTR = 3h and DSTACTV = 2h implies that two TRs are present. The first pending TR is read from the destination FIFO register entry 3 and the second pending TR is read from the destination FIFO register entry 0.

Copyright © 2013–2016, Texas Instruments Incorporated

17.2.12 Event Dataflow

This section summarizes the data flow of a single event, from the time the event is latched to the channel controller to the time the transfer completion code is returned. The following steps list the sequence of EDMA3CC activity:

1. Event is asserted from an external source (peripheral or external interrupt). This also is similar for a manually-triggered, chained-triggered, or QDMA-triggered event. The event is latched into the ER.E*n* (or CER.E*n*, ESR.E*n*, QER.E*n*) bit.
2. Once an event is prioritized and queued into the appropriate event queue, the SER.E*n* (or QSER.E*n*) bit is set to inform the event prioritization/processing logic to disregard this event since it is already in the queue. Alternatively, if the transfer controller and the event queue are empty, then the event bypasses the queue.
3. The EDMA3CC processing and the submission logic evaluates the appropriate PaRAM set and determines whether it is a non-null and non-dummy transfer request (TR).
4. The EDMA3CC clears the ER.E*n* (or CER.E*n*, ESR.E*n*, QER.E*n*) bit and the SER.E*n* bit as soon as it determines the TR is non-null. In the case of a null set, the SER.E*n* bit remains set. It submits the non-null/non-dummy TR to the associated transfer controller. If the TR was programmed for early completion, the EDMA3CC immediately sets the interrupt pending register (IPR.I[TCC]).
5. If the TR was programmed for normal completion, the EDMA3CC sets the interrupt pending register (IPR.I[TCC]) when the EDMA3TC informs the EDMA3CC about completion of the transfer (returns transfer completion codes).
6. The EDMA3CC programs the associated EDMA3TC*n* Program Register Set with the TR.
7. The TR is then passed to the Source Active set and the Dst FIFO Register Set, if both the register sets are available.
8. The Read Controller processes the TR by issuing read commands to the source slave endpoint. The Read Data lands in the Data FIFO of the EDMA3TC*n*.
9. As soon as sufficient data is available, the Write Controller begins processing the TR by issuing write commands to the destination slave endpoint.
10. This continues until the TR completes and on receiving the acknowledgement signal from the destination slave end point, the EDMA3TC*n* then signals completion status to the EDMA3CC.

Copyright © 2013–2016, Texas Instruments Incorporated

17.2.13 EDMA3 Prioritization

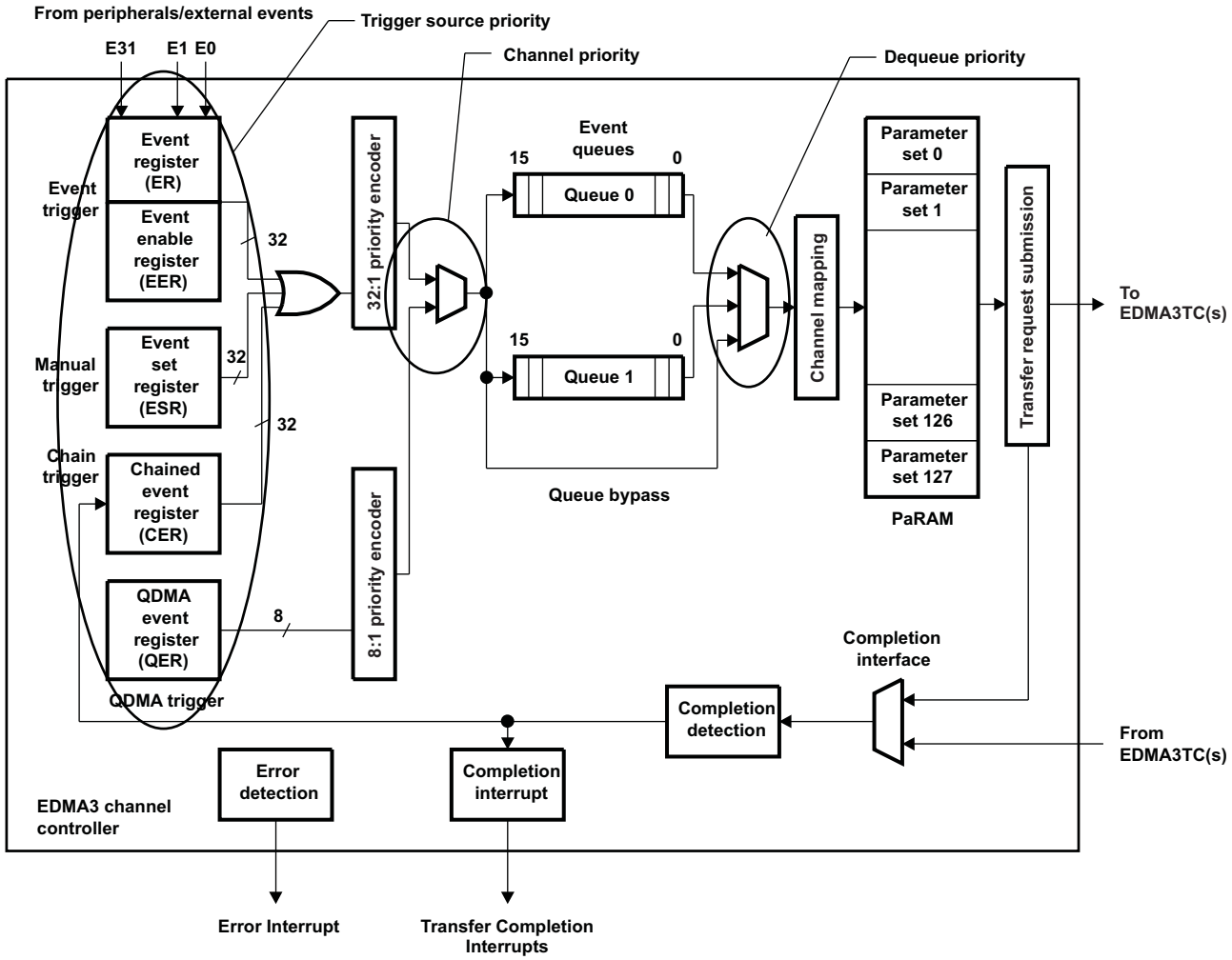
The EDMA3 controller has many implementation rules to deal with concurrent events/channels, transfers, etc. The following subsections detail various arbitration details whenever there might be occurrence of concurrent activity. Figure 17-14 shows the different places EDMA3 priorities come into play.

17.2.13.1 Channel Priority

The DMA event register (ER) captures all external/peripheral events connected to the EDMA3CC; likewise, the QDMA event register (QER) captures QDMA events for all QDMA channels; therefore, it is possible for events to occur simultaneously on the DMA/QDMA event inputs. For events arriving simultaneously, the event associated with the lowest channel number is prioritized for submission to the event queues (for DMA events, channel 0 has the highest priority and channel 31 has the lowest priority; similarly, for QDMA events, channel 0 has the highest priority and channel 7 has the lowest priority). This mechanism only sorts simultaneous events for submission to the event queues.

If a DMA and QDMA event occurs simultaneously, the DMA event always has prioritization against the QDMA event for submission to the event queues.

Figure 17-14. EDMA3 Prioritization



Copyright © 2013–2016, Texas Instruments Incorporated

17.2.13.2 Trigger Source Priority

If a DMA channel is associated with more than one trigger source (event trigger, manual trigger, and chain trigger), and if multiple events are set simultaneously for the same channel (ER.E*n* = 1, ESR.E*n* = 1, CER.E*n* = 1), then the EDMA3CC always services these events in the following priority order: event trigger (via ER) is higher priority than chain trigger (via CER) and chain trigger is higher priority than manual trigger (via ESR).

This implies that if for channel 0, both ER.E0 = 1 and CER.E0 = 1 at the same time, then the ER.E0 event is always queued before the CER.E0 event.

17.2.13.3 Dequeue Priority

The priority of the associated transfer request (TR) is further mitigated by which event queue is being used for event submission (dictated by DMAQNUM*n* and QDMAQNUM). For submission of a TR to the transfer controller, events need to be dequeued from the event queues. A lower numbered queue has a higher dequeuing priority then a higher numbered queue. For example, if there are events in Q0 and Q1 and the respective transfer controllers (TC0 and TC1) are ready to receive the next TR from the EDMA3CC, then the transfer requests associated with events in Q0 will get submitted to TC0 prior to any transfer requests associated with events in Q1 getting submitted to TC1.

NOTE: At any given time, if there are outstanding events in multiple queues, when the transfer controller associated with the lower numbered (higher priority) queue is busy processing earlier transfer requests and the transfer controller associated with the higher numbered (lower priority) queue is idle, then the event in the higher numbered (lower priority) queue will dequeue first.

17.2.13.4 Master (Transfer Controller) Priority

All master peripherals on the device have a programmable priority level. When multiple masters are trying to access common shared resources (slave memory or peripherals), this priority value allows the system interconnect to arbitrate requests from different masters based on their priority. This priority assignment is determined in the Master Priority Registers (MSTPRI0-MSTPRI2) in the System Configuration Module (see the *System Configuration (SYSCFG) Module* chapter), where each master has an allocated priority value (power on reset default value), which can be re programmed based on the applications prioritization requirements. The priority value can be configured between 0 to 7, with 0 being the highest priority and 7 being the lowest priority.

Each transfer controller on the device is also a master peripheral. The priority of the transfer requests (read/write commands) issued by the individual EDMA3TC read/write ports in the system can be programmed via these registers.

The dequeue priority has a relatively secondary effect as compared to this Master priority; therefore, it is important to program the priority of each transfer controller with respect to each other and also with respect to other masters in the system.

NOTE: On previous architectures, the EDMA3TC priority was controlled by the QUEPRI register in the EDMA3CC memory-map. However for this device, the priority control for the transfer controllers is controlled by the chip-level registers in the System Configuration Module.

Copyright © 2013–2016, Texas Instruments Incorporated

17.2.14 EDMA3CC and EDMA3TC Performance and System Considerations

17.2.14.1 System Priority Considerations

The main switched central resource (SCR) (see your device-specific data manual) arbitrates bus requests from all the masters (CPU, master peripherals, and the EDMA3 transfer controllers) to the shared slave resources (peripherals and memories). The priorities of transfer requests (read and write commands) from the EDMA3 transfer controllers with respect to each other and the other masters within the system is configured as explained in Section 17.2.13.4.

It is recommended that this priority be altered based on system level considerations. For example, peripherals servicing audio/video/display threads that typically have real-time deadlines should be programmed as highest priority requestors in the systems, where as, peripherals responsible for doing bulk/block/paging transfers with no real-time deadlines, should be programmed as a lower system priority.

The default priority for all transfer controllers is the same, 0 or highest priority relative to other masters; therefore, it is recommended that a TC servicing audio data requests from serial ports should be configured at a higher priority as compared to TC service memory to memory (paging/bulk) transfer requests.

17.2.14.2 TC Transfer Optimization Considerations

The transfer controller can internally optimize the way it issues read commands and write commands for a given transfer under certain conditions. For 2D transfers (that is, BCNT arrays of ACNT bytes), if the ACNT value is less than or equal to the DBS value, then the transfer controller will try to optimize the TR into a 1D transfer in order to maximize efficiency. The optimization only takes place if the EDMA3TC recognizes that the 2D transfer is organized as a single dimension (SAM/DAM = 0, increment mode), SRC/DST BIDX = ACNT, the ACNT value is a power of 2, and the BCNT value is less than or equal to 1023. If these conditions are met, then instead of issuing ACNT bytes worth read and/or write commands, the TC will try to optimize the bus usage by issuing commands as if ACNT' = ACNT × BCNT and BCNT = 1.

Table 17-13 summarizes the conditions in which the optimizations are performed.

Table 17-13. Read/Write Command Optimization Rules

ACNT ≤ DBS	ACNT is power of 2	BIDX = ACNT	BCNT ≤ 1023	SAM/DAM = 0 (Increment)	Description
Yes	Yes	Yes	Yes	Yes	Optimized
Yes	No	x	x	Yes	Not Optimized
Yes	x	No	x	Yes	Not Optimized
No	x	x	x	Yes	Not Optimized
x	x	x	x	No	Not Optimized

Consider a case in which it is needed to transfer 4096 bytes where the data is arranged linearly in both the source and destination locations (SAM/DAM = 0, SRC/DST BIDX = ACNT): Scenario A programs the ACNT = 4, BCNT = 1024 , AB-synchronized transfer; and Scenario B programs the ACNT = 64, BCNT = 64. Scenario B will yield a much optimized transfer and higher throughput, as the transfer meets all the optimization rules, which would result in TC internally treating it as a transfer with an ACNT' = 4096 (ACNT × BCNT). The TC will optimally size, default burst size worth read and write commands. In the case of Scenario B, since one of the optimization rules is not met (BCNT value is greater then 1023), the TC will end up issuing several ACNT byte (4 byte) size commands to complete the transfers, which will result in inefficient usage of the read/write buses.

Copyright © 2013–2016, Texas Instruments Incorporated

17.2.14.3 Throttling the Read Command Rate in a Transfer Controller

By default, the transfer controller issues reads as fast as possible. In some cases, the reads issued by the EDMA3TCC could fill the available command buffering for a slave, delaying other (potentially higher priority) masters from successfully submitting commands to that slave. The rate at which read commands are issued by the EDMA3TC is controlled by the read command rate register (RDRATE), and this can be used to throttle the rate at which the commands are issued from the TC read interface. RDRATE defines the number of cycles that the EDMA3TC read controller waits before issuing subsequent commands for a given TR, thus minimizing the chance of the EDMA3TC consuming all available slave resources. The RDRATE value should be set to a relatively small value (or kept at default, which implies issuing read requests as fast as possible) if the transfer controller is targeted for high-priority transfers and set to a high value if the transfer controller is targeted for low-priority transfers. In contrast, the write Interface does not have any performance turning knobs because writes always have an interval between commands as write commands are submitted along with the associated write data.

17.2.15 EDMA3 Operating Frequency (Clock Control)

The EDMA3 channel controller and transfer controller are clocked from PLL controller 0 (PLLC0). For details, see the *Phase-Locked Loop Controller (PLLC)* chapter.

17.2.16 Reset Considerations

A hardware reset resets the EDMA3 (EDMA3CC and EDMA3TC) and the EDMA3 configuration registers. The PaRAM memory contents are undefined after device reset and you should not rely on parameters to be reset to a known state. The PaRAM set must be initialized to a desired value before it is used.

17.2.17 Power Management

The EDMA3 (EDMA3CC and EDMA3TC) can be placed in reduced-power modes to conserve power during periods of low activity. The power management of the peripheral is controlled by the device Power and Sleep Controller (PSC). The PSC acts as a master controller for power management for all peripherals on the device. For detailed information on power management procedures using the PSC, see the *Power and Sleep Controller (PSC)* chapter.

The EDMA3 controller can be idled on receiving a clock stop request from the PSC. The requests to EDMA3CC and EDMA3TC are separate. In general, you should verify that there are no pending activities in the EDMA3 controller before issuing a clock stop request via PSC.

The EDMA3CC checks for the following conditions:

- No pending DMA/QDMA events
- No outstanding events in the event queues
- Transfer request processing logic is not active
- No completion requests outstanding (early or normal completion)
- No configuration bus requests in progress

The first four conditions are software readable by the channel controller status register (CCSTAT) in the EDMA3CC.

Similarly, from the EDMA3TC perspective, you should check that there are no outstanding TRs that are getting processed and essentially the read/write controller is not busy processing a TR. The activity of EDMA3TC logic is read in TCSTAT for each EDMA3TC.

It is generally recommended to first disable the EDMA3CC and then the EDMA3TC(s) to put the EDMA3 controller in reduced-power modes.

Additionally, when EDMA3 is involved in servicing a peripheral and it is required to power-down both the peripheral and the EDMA, the recommended sequence is to first disable the peripheral, then disable the DMA channel associated with the peripheral (clearing the EER bit for the channel), then disable the EDMA3CC, and finally disable the EDMA3TC(s).

Copyright © 2013–2016, Texas Instruments Incorporated

17.2.18 Emulation Considerations

During debug when using the emulator, the CPU(s) may be halted on an execute packet boundary for single-stepping, benchmarking, profiling, or other debug purposes. During an emulation halt, the EDMA3 channel controller and transfer controller operations continue. Events continue to be latched and processed and transfer requests continue to be submitted and serviced.

Since EDMA3 is involved in servicing multiple master and slave peripherals, it is not feasible to have an independent behavior of the EDMA3 for emulation halts. EDMA3 functionality would be coupled with the peripherals it is servicing, which might have different behavior during emulation halts. For example, if a multichannel buffered serial port (McBSP) is halted during an emulation access (FREE = 0 and SOFT = 0 or 1 in the McBSP registers), the McBSP stops generating the McBSP receive or transmit events (REVT or XEVT) to the EDMA. From the point of view of the McBSP, the EDMA3 is suspended, but other peripherals (for example, a timer) still assert events and will be serviced by the EDMA.

17.3 Transfer Examples

The EDMA3 channel controller performs a variety of transfers depending on the parameter configuration. The following sections provides a description and PaRAM configuration for some typical use case scenarios.

17.3.1 Block Move Example

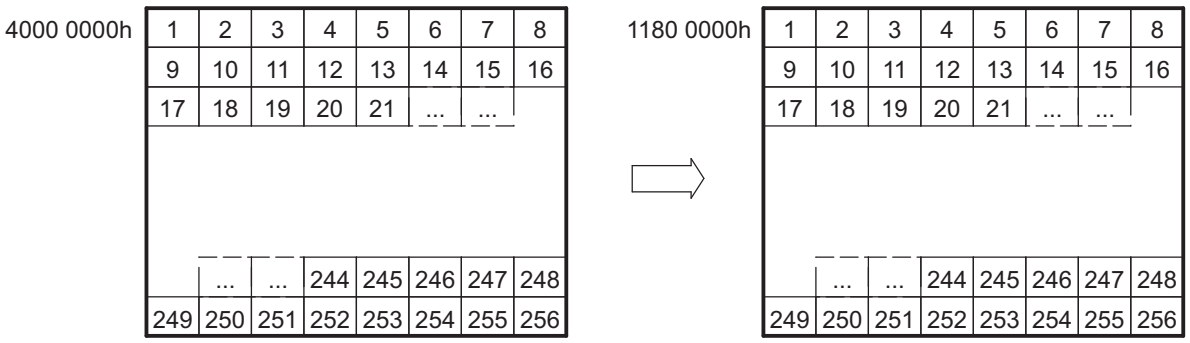
The most basic transfer performed by the EDMA3 is a block move. During device operation it is often necessary to transfer a block of data from one location to another, usually between on-chip and off-chip memory.

In this example, a section of data is to be copied from external memory to internal L2 SRAM. A data block of 256 bytes residing at address 4000 0000h (external memory) needs to be transferred to internal address 1180 0000h (L2), as shown in Figure 17-15. Figure 17-16 shows the parameters for this transfer.

The source address for the transfer is set to the start of the data block in external memory, and the destination address is set to the start of the data block in L2. If the data block is less than 64K bytes, the PaRAM configuration in Figure 17-16 holds true with the synchronization type set to A-synchronized and indexes cleared to 0. If the amount of data is greater than 64K bytes, BCNT and the B-indexes need to be set appropriately with the synchronization type set to AB-synchronized. The STATIC bit in OPT is set to prevent linking.

This transfer example may also be set up using QDMA. For successive transfer submissions, of a similar nature, the number of cycles used to submit the transfer are fewer depending on the number of changing transfer parameters. You may program the QDMA trigger word to be the highest numbered offset in the PaRAM set that undergoes change.

Figure 17-15. Block Move Example



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 17-16. Block Move Example PaRAM Configuration

(a) EDMA Parameters

Parameter Contents		Parameter	
0010 0008h		Channel Options Parameter (OPT)	
4000 0000h		Channel Source Address (SRC)	
0001h	0100h	Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
1180 0000h		Channel Destination Address (DST)	
0000h	0000h	Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
0000h	FFFFh	BCNT Reload (BCNTRLD)	Link Address (LINK)
0000h	0000h	Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
0000h	0001h	Reserved	Count for 3rd Dimension (CCNT)

(b) Channel Options Parameter (OPT) Content

31	30 28	27 24	23	22	21	20	19 18	17 16
0	000	0000	0	0	0	1	00	00
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC

15 12	11	10 8	7 4	3	2	1	0
0000	0	000	0000	1	0	0	0
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM

Copyright © 2013–2016, Texas Instruments Incorporated

17.3.2 Subframe Extraction Example

The EDMA3 can efficiently extract a small frame of data from a larger frame of data. By performing a 2D-to-1D transfer, the EDMA3 retrieves a portion of data for the CPU to process. In this example, a 640 × 480-pixel frame of video data is stored in external memory, SDRAM. Each pixel is represented by a 16-bit halfword. The CPU extracts a 16 × 12-pixel subframe of the image for processing. To facilitate more efficient processing time by the CPU, the EDMA3 places the subframe in internal L2 SRAM. Figure 17-17 shows the transfer of a subframe from external memory to L2. Figure 17-18 shows the parameters for this transfer.

The same PaRAM set options are used for QDMA channels, as well as DMA channels. The STATIC bit in OPT is set to 1 to prevent linking. For successive transfers, only changed parameters need to be programmed before triggering the channel.

Figure 17-17. Subframe Extraction Example

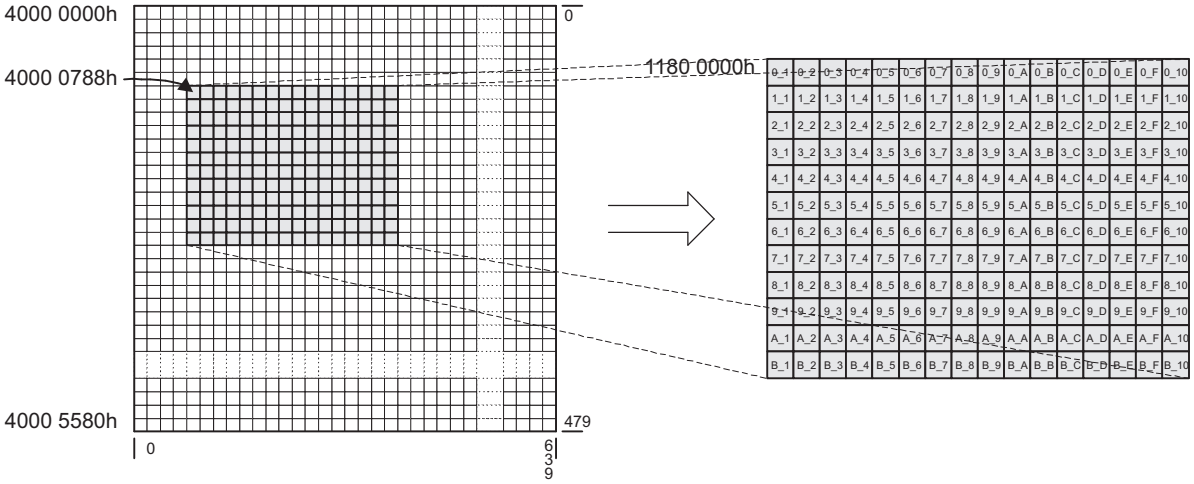


Figure 17-18. Subframe Extraction Example PaRAM Configuration

(a) EDMA Parameters

Parameter Contents	
0010 000Ch	
4000 0788h	
000Ch	0020h
1180 0000h	
0020h	0500h
0000h	FFFFh
0000h	0000h
0000h	0001h

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

(b) Channel Options Parameter (OPT) Content

31	30 28	27 24	23	22	21	20	19 18	17 16
0	000	0000	0	0	0	1	00	00
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC

15 12	11	10 8	7 4	3	2	1	0
0000	0	000	0000	1	1	0	0
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM

Copyright © 2013–2016, Texas Instruments Incorporated

17.3.3 Data Sorting Example

Many applications require the use of multiple data arrays; it is often desirable to have the arrays arranged such that the first elements of each array are adjacent, the second elements are adjacent, and so on. Often this is not how the data is presented to the device. Either data is transferred via a peripheral with the data arrays arriving one after the other or the arrays are located in memory with each array occupying a portion of contiguous memory spaces. For these instances, the EDMA3 can reorganize the data into the desired format. Figure 17-19 shows the data sorting.

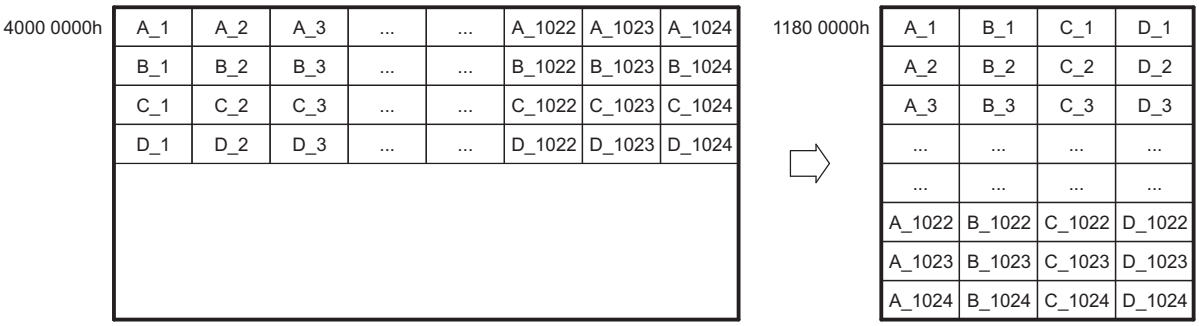
In order to determine the parameter entry values, the following need to be considered:

- ACNT – Program this to be the size in bytes of an array.
- BCNT – Program this to be the number of arrays in a frame.
- CCNT – Program this to be the number of frames.
- SRCBIDX – Program this to be the size of the array or ACNT.
- DSTBIDX = CCNT × ACNT
- SRCCIDX = ACNT × BCNT
- DSTCIDX = ACNT

The synchronization type needs to be AB-synchronized and the STATIC bit is 0 to allow updates to the parameter set. It is advised to use normal DMA channels for sorting.

It is not possible to sort this with a single trigger event. Instead, the channel can be programmed to be chained to itself. After BCNT arrays get sorted, intermediate chaining could be used to trigger the channel again causing the transfer of the next BCNT arrays and so on. Figure 17-20 shows the parameter set programming for this transfer, assuming channel 0 and an array size of 4 bytes.

Figure 17-19. Data Sorting Example



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 17-20. Data Sorting Example PaRAM Configuration

(a) EDMA Parameters

Parameter Contents	Parameter
0090 0004h	Channel Options Parameter (OPT)
4000 0000h	Channel Source Address (SRC)
0400h | 0004h	Count for 2nd Dimension (BCNT) | Count for 1st Dimension (ACNT)
1180 0000h	Channel Destination Address (DST)
0010h | 0004h	Destination BCNT Index (DSTBIDX) | Source BCNT Index (SRCBIDX)
0000h | FFFFh	BCNT Reload (BCNTRLD) | Link Address (LINK)
0004h | 1000h	Destination CCNT Index (DSTCIDX) | Source CCNT Index (SRCCIDX)
0000h | 0004h	Reserved | Count for 3rd Dimension (CCNT)

(b) Channel Options Parameter (OPT) Content

31	30 28	27 24	23	22	21	20	19 18	17 16
0	000	0000	1	0	0	1	00	00
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC

15 12	11	10 8	7 4	3	2	1	0
0000	0	000	0000	0	1	0	0
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM

Copyright © 2013–2016, Texas Instruments Incorporated

17.3.4 Peripheral Servicing Example

NOTE: Examples in this section are sample examples. The peripherals, channels, and addresses used in these examples may not apply to your specific device. See your device-specific data manual for supported peripherals.

The EDMA3 channel controller also services peripherals in the background of CPU operation, without requiring any CPU intervention. Through proper initialization of the DMA channels, they can be configured to continuously service on-chip and off-chip peripherals throughout the device operation. Each event available to the EDMA3 has its own dedicated channel, and all channels operate simultaneously. The only requirements are to use the proper channel for a particular transfer and to enable the channel event in the event enable register (EER). When programming a DMA channel to service a peripheral, it is necessary to know how data is to be presented to the CPU. Data is always provided with some kind of synchronization event as either one element per event (nonbursting) or multiple elements per event (bursting).

17.3.4.1 Nonbursting Peripherals

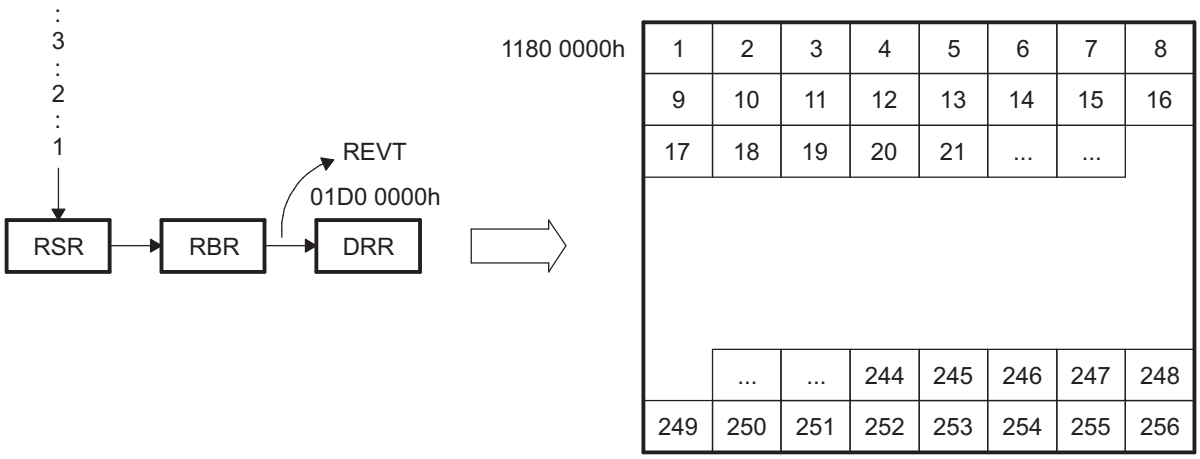
Nonbursting peripherals include the on-chip multichannel buffered serial port (McBSP) and many external devices, such as codecs. Regardless of the peripheral, the DMA channel configuration is the same.

The McBSP transmit and receive data streams are treated independently by the EDMA3. The transmit and receive data streams can have completely different counts, data sizes, and formats. Figure 17-21 shows servicing incoming McBSP data.

To transfer the incoming data stream to its proper location in L2 memory, the DMA channel must be set up for a 1D-to-1D transfer with A-synchronization. Since an event (REVT) is generated for every word as it arrives, it is necessary to have the EDMA3 issue the transfer request for each element individually. Figure 17-22 shows the parameters for this transfer. The source address of the DMA channel is set to the data receive register (DRR) address for the McBSP, and the destination address is set to the start of the data block in L2. Since the address of DRR is fixed, the source B index is cleared to 0 (no modification) and the destination B index is set to 01b (increment).

Based on the premise that serial data is typically a high priority, the DMA channel should be programmed to be on queue 0.

Figure 17-21. Servicing Incoming McBSP Data Example



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 17-22. Servicing Incoming McBSP Data Example PaRAM

(a) EDMA Parameters

Parameter Contents		Parameter	
0010 0000h		Channel Options Parameter (OPT)	
01D0 0000h		Channel Source Address (SRC)	
0100h	0001h	Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
1180 0000h		Channel Destination Address (DST)	
0001h	0000h	Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
0000h	FFFFh	BCNT Reload (BCNTRLD)	Link Address (LINK)
0000h	0000h	Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
0000h	0004h	Reserved	Count for 3rd Dimension (CCNT)

(b) Channel Options Parameter (OPT) Content

31	30 28	27 24	23	22	21	20	19 18	17 16
0	000	0000	0	0	0	1	00	00
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC

15 12	11	10 8	7 4	3	2	1	0
0000	0	000	0000	0	0	0	0
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM

Copyright © 2013–2016, Texas Instruments Incorporated

17.3.4.2 Bursting Peripherals

Higher bandwidth applications require that multiple data elements be presented to the CPU for every synchronization event. This frame of data can either be from multiple sources that are working simultaneously or from a single high-throughput peripheral that streams data to/from the CPU. In this example, a port is receiving a video frame from a camera and presenting it to the CPU one array at a time. The video image is 640 × 480 pixels, with each pixel represented by a 16-bit element. The image is to be stored in external memory. Figure 17-23 shows this example.

To transfer data from an external peripheral to an external buffer one array at a time based on EVT*n* , channel *n* must be configured. Due to the nature of the data (a video frame made up of arrays of pixels) the destination is essentially a 2D entity. Figure 17-24 shows the parameters to service the incoming data with a 1D-to-2D transfer using AB-synchronization. The source address is set to the location of the video framer peripheral, and the destination address is set to the start of the data buffer. Since the input address is static, the SRCBIDX is 0 (no modification to the source address). The destination is made up of arrays of contiguous, linear elements; therefore, the DSTBIDX is set to pixel size, 2 bytes. ANCT is equal to the pixel size, 2 bytes. BCNT is set to the number of pixels in an array, 640. CCNT is equal to the total number of arrays in the block, 480. SRCCIDX is 0 since the source address undergoes no increment. The DSTCIDX is equal to the difference between the starting addresses of each array. Since a pixel is 16 bits (2 bytes), DSTCIDX is equal to 640 × 2.

Figure 17-23. Servicing Peripheral Burst Example

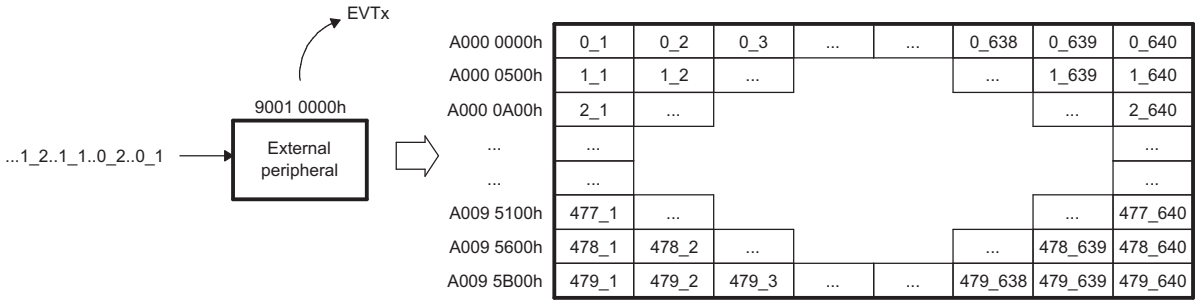


Figure 17-24. Servicing Peripheral Burst Example PaRAM

(a) EDMA Parameters

Parameter Contents	
0010 0004h	
Channel Source Address	
0280h	0002h
4000 0000h	
0002h	0000h
0000h	FFFFh
0500h	0000h
0000h	01E0h

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

(b) Channel Options Parameter (OPT) Content

31	30 28	27 24	23	22	21	20	19 18	17 16
0	000	0000	0	0	0	1	00	00
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC

15 12	11	10 8	7 4	3	2	1	0
0000	0	000	0000	0	1	0	0
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM

Copyright © 2013–2016, Texas Instruments Incorporated

17.3.4.3 Continuous Operation

Configuring a DMA channel to receive a single frame of data is useful, and is applicable to some systems. A majority of the time, however, data is going to be continuously transmitted and received throughout the entire operation of the CPU. In this case, it is necessary to implement some form of linking such that the DMA channels continuously reload the necessary parameter sets. In this example, the multichannel buffered serial port (McBSP) is configured to transmit and receive data on a array. To simplify the example, only two channels are active for both transmit and receive data streams. Each channel receives packets of 128 elements. The packets are transferred from the serial port to L2 memory and from L2 memory to the serial port, as shown in Figure 17-25.

The McBSP generates REVT for every element received and generates XEVT for every element transmitted. To service the data streams, the DMA channels associated with the McBSP must be set up for 1D-to-1D transfers with A-synchronization.

Figure 17-26 shows the parameters for the parameter entries for the channel for these transfers. In order to service the McBSP continuously throughout CPU operation, the channels must be linked to a duplicate PaRAM set in the PaRAM. After all frames have been transferred, the DMA channels reload and continue. Figure 17-27 shows the reload parameters for the channel.

17.3.4.3.1 Receive Channel

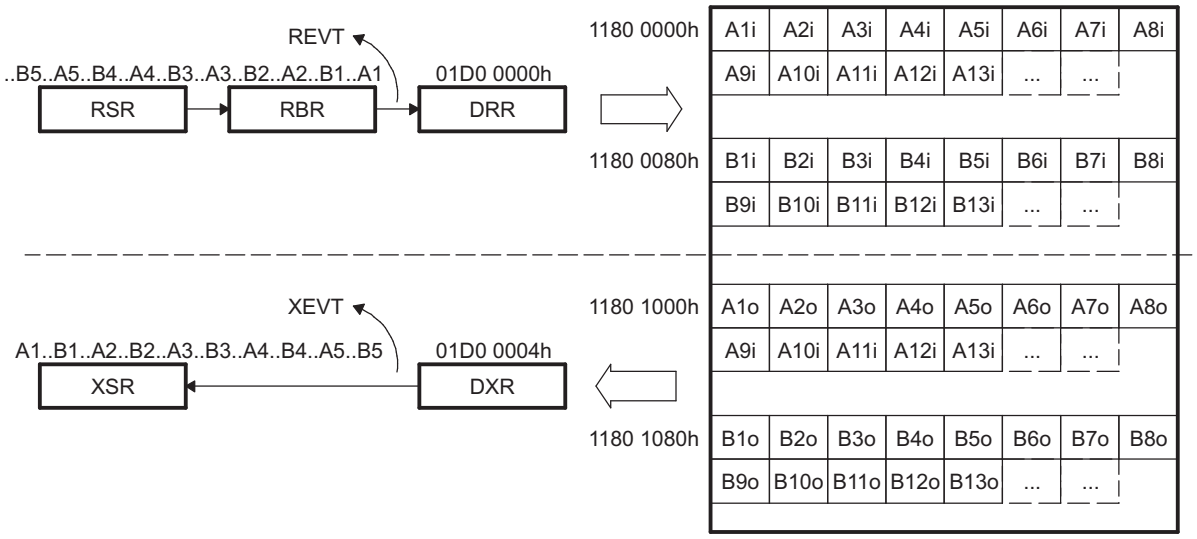
DMA channel 3 services the incoming data stream of the McBSP. The source address is set to that of the data receiver register (DRR), and the destination address is set to the first element of the data block. Since there are two data channels being serviced, A and B, they are to be located separately within the L2 SRAM.

In order to facilitate continuous operation, a copy of the PaRAM set for the channel is placed in PaRAM set 64. The LINK option is set and the link address is provided in the PaRAM set. Upon exhausting the channel 3 parameter set, the parameters located at the link address are loaded into the channel 3 parameter set and operation continues. This function continues throughout device operation until halted by the CPU.

17.3.4.3.2 Transmit Channel

DMA channel 2 services the outgoing data stream of the McBSP. In this case the destination address needs no update, hence, the parameter set changes accordingly. Linking is also used to allow continuous operation by the DMA channel, with duplicate PaRAM set entries at PaRAM set 65.

Figure 17-25. Servicing Continuous McBSP Data Example



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 17-26. Servicing Continuous McBSP Data Example PaRAM

(a) EDMA Parameters for Receive Channel (PaRAM Set 3) being Linked to PaRAM Set 64

Parameter Contents	Parameter
0010 0000h	Channel Options Parameter (OPT)
01D0 0000h	Channel Source Address (SRC)
0080h | 0001h	Count for 2nd Dimension (BCNT) | Count for 1st Dimension (ACNT)
1180 0000h	Channel Destination Address (DST)
0001h | 0000h	Destination BCNT Index (DSTBIDX) | Source BCNT Index (SRCBIDX)
0080h | 4800h	BCNT Reload (BCNTRLD) | Link Address (LINK)
0000h | 0000h	Destination CCNT Index (DSTCIDX) | Source CCNT Index (SRCCIDX)
0000h | 0001h	Reserved | Count for 3rd Dimension (CCNT)

(b) Channel Options Parameter (OPT) Content for Receive Channel (PaRAM Set 3)

31	30 28	27 24	23	22	21	20	19 18	17 16
0	000	0000	0	0	0	1	00	00
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC

15 12	11	10 8	7 4	3	2	1	0
0000	0	000	0000	0	0	0	0
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM

(c) EDMA Parameters for Transmit Channel (PaRAM Set 2) being Linked to PaRAM Set 65

Parameter Contents	Parameter
0010 1000h	Channel Options Parameter (OPT)
1180 1000h	Channel Source Address (SRC)
0080h | 0001h	Count for 2nd Dimension (BCNT) | Count for 1st Dimension (ACNT)
01D0 0004h	Channel Destination Address (DST)
0000h | 0001h	Destination BCNT Index (DSTBIDX) | Source BCNT Index (SRCBIDX)
0080h | 4820h	BCNT Reload (BCNTRLD) | Link Address (LINK)
0000h | 0000h	Destination CCNT Index (DSTCIDX) | Source CCNT Index (SRCCIDX)
0000h | 0001h	Reserved | Count for 3rd Dimension (CCNT)

(d) Channel Options Parameter (OPT) Content for Transmit Channel (PaRAM Set 2)

31	30 28	27 24	23	22	21	20	19 18	17 16
0	000	0000	0	0	0	1	00	00
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC

15 12	11	10 8	7 4	3	2	1	0
0001	0	000	0000	0	0	0	0
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM

Figure 17-27. Servicing Continuous McBSP Data Example Reload PaRAM

(a) EDMA Reload Parameters (PaRAM Set 64) for Receive Channel

Copyright © 2013–2016, Texas Instruments Incorporated

Parameter Contents	
0010 0000h	
01D0 0000h	
0080h	0001h
1180 0000h	
0001h	0000h
0080h	4800h
0000h	0000h
0000h	0001h

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

(b) Channel Options Parameter (OPT) Content for Receive Channel (PaRAM Set 64)

31	30 28	27 24	23	22	21	20	19 18	17 16
0	000	0000	0	0	0	1	00	00
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC

15 12	11	10 8	7 4	3	2	1	0
0000	0	000	0000	0	0	0	0
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM

(c) EDMA Reload Parameters (PaRAM Set 65) for Transmit Channel

Parameter Contents	
0010 1000h	
1180 1000h	
0080h	0001h
01D0 0004h	
0000h	0001h
0080h	4820h
0000h	0000h
0000h	0001h

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

(d) Channel Options Parameter (OPT) Content for Transmit Channel (PaRAM Set 65)

31	30 28	27 24	23	22	21	20	19 18	17 16
0	000	0000	0	0	0	1	00	00
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC

15 12	11	10 8	7 4	3	2	1	0
0001	0	000	0000	0	0	0	0
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM

Copyright © 2013–2016, Texas Instruments Incorporated

17.3.4.4 Ping-Pong Buffering

Although the previous configuration allows the EDMA3 to service a peripheral continuously, it presents a number of restrictions to the CPU. Since the input and output buffers are continuously being filled/emptied, the CPU must match the pace of the EDMA3 very closely in order to process the data. The EDMA3 receive data must always be placed in memory before the CPU accesses it, and the CPU must provide the output data before the EDMA3 transfers it. Though not impossible, this is an unnecessary challenge. It is particularly difficult in a 2-level cache scheme.

Ping-pong buffering is a simple technique that allows the CPU activity to be distanced from the EDMA3 activity. This means that there are multiple (usually two) sets of data buffers for all incoming and outgoing data streams. While the EDMA3 transfers the data into and out of the ping buffers, the CPU manipulates the data in the pong buffers. When both CPU and EDMA3 activity completes, they switch. The EDMA3 then writes over the old input data and transfers the new output data. Figure 17-28 shows the ping-pong scheme for this example.

To change the continuous operation example, such that a ping-pong buffering scheme is used, the DMA channels need only a moderate change. Instead of one link parameter set, there are two; one for transferring data to/from the ping buffers and one for transferring data to/from the pong buffers. As soon as one transfer completes, the channel loads the PaRAM set for the other and the data transfers continue. Figure 17-29 shows the DMA channel configuration required.

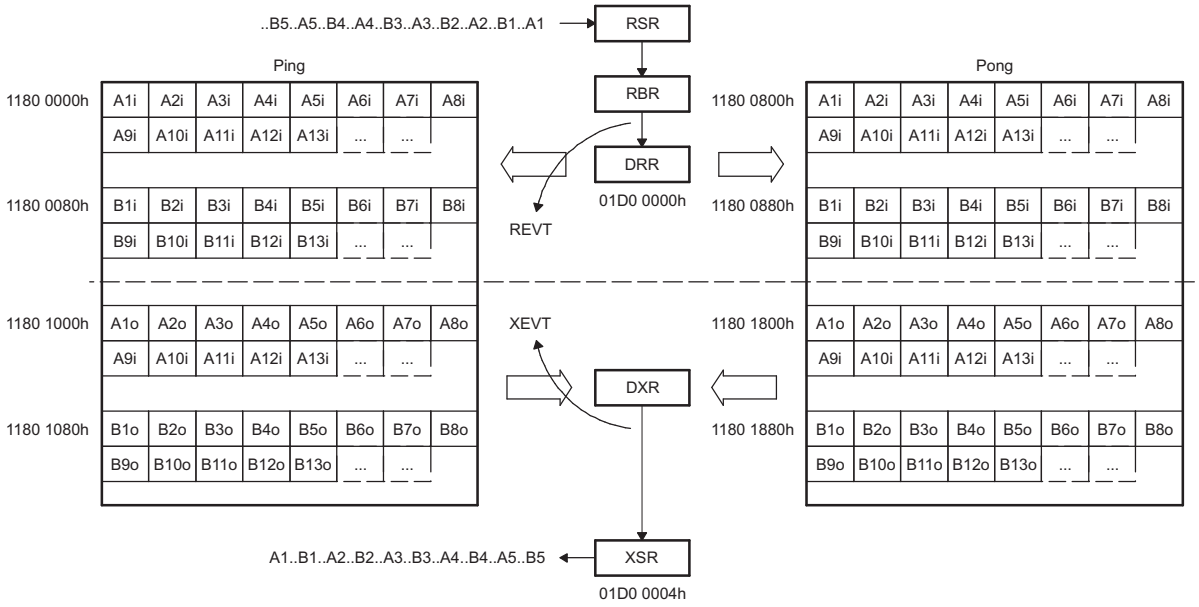
Each channel has two link parameter sets, ping and pong. The DMA channel is initially loaded with the ping parameters (Figure 17-29). The link address for the ping set is set to the PaRAM offset of the pong parameter set (Figure 17-30). The link address for the pong set is set to the PaRAM offset of the ping parameter set (Figure 17-31). The channel options, count values, and index values are all identical between the ping and pong parameters for each channel. The only differences are the link address provided and the address of the data buffer.

Copyright © 2013–2016, Texas Instruments Incorporated

17.3.4.4.1 Synchronization with the CPU

In order to utilize the ping-pong buffering technique, the system must signal the CPU when to begin to access the new data set. After the CPU finishes processing an input buffer (ping), it waits for the EDMA3 to complete before switching to the alternate (pong) buffer. In this example, both channels provide their channel numbers as their report word and set the TCINTEN bit to 1 to generate an interrupt after completion. When channel 3 fills an input buffer, the E3 bit in the interrupt pending register (IPR) is set to 1; when channel 2 empties an output buffer, the E2 bit in IPR is set to 1. The CPU must manually clear these bits. With the channel parameters set, the CPU polls IPR to determine when to switch. The EDMA3 and CPU could alternatively be configured such that the channel completion interrupts the CPU. By doing this, the CPU could service a background task while waiting for the EDMA3 to complete.

Figure 17-28. Ping-Pong Buffering for McBSP Data Example



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 17-29. Ping-Pong Buffering for McBSP Example PaRAM

(a) EDMA Parameters for Channel 3 (Using PaRAM Set 3 Linked to Pong Set 64)

Parameter Contents
0010 3000h
01D0 0000h
0080h | 0001h
1180 0000h
0001h | 0000h
0080h | 4800h
0000h | 0000h
0000h | 0001h

Parameter
Channel Options Parameter (OPT)
Channel Source Address (SRC)
Count for 2nd Dimension (BCNT) | Count for 1st Dimension (ACNT)
Channel Destination Address (DST)
Destination BCNT Index (DSTBIDX) | Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD) | Link Address (LINK)
Destination CCNT Index (DSTCIDX) | Source CCNT Index (SRCCIDX)
Reserved | Count for 3rd Dimension (CCNT)

(b) Channel Options Parameter (OPT) Content for Channel 3

31	30 28	27 24	23	22	21	20	19 18	17 16
0	000	0000	0	0	0	1	00	00
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC

15 12	11	10 8	7 4	3	2	1	0
0011	0	000	0000	0	0	0	0
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM

(c) EDMA Parameters for Channel 2 (Using PaRAM Set 2 Linked to Pong Set 65)

Parameter Contents
0010 2000h
1180 1000h
0080h | 0001h
01D0 0004h
0000h | 0001h
0080h | 4840h
0000h | 0000h
0000h | 0001h

Parameter
Channel Options Parameter (OPT)
Channel Source Address (SRC)
Count for 2nd Dimension (BCNT) | Count for 1st Dimension (ACNT)
Channel Destination Address (DST)
Destination BCNT Index (DSTBIDX) | Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD) | Link Address (LINK)
Destination CCNT Index (DSTCIDX) | Source CCNT Index (SRCCIDX)
Reserved | Count for 3rd Dimension (CCNT)

(d) Channel Options Parameter (OPT) Content for Channel 2

31	30 28	27 24	23	22	21	20	19 18	17 16
0	000	0000	0	0	0	1	00	00
PRIV	Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC

15 12	11	10 8	7 4	3	2	1	0
0010	0	000	0000	0	0	0	0
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM

Figure 17-30. Ping-Pong Buffering for McBSP Example Pong PaRAM

(a) EDMA Pong Parameters for Channel 3 at Set 64 Linked to Set 65

Copyright © 2013–2016, Texas Instruments Incorporated

Parameter Contents	
0010 D000h	
01D0 0000h	
0080h	0001h
1180 0800h	
0001h	0000h
0080h	4820h
0000h	0000h
0000h	0001h

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

(b) EDMA Pong Parameters for Channel 2 at Set 66 Linked to Set 67

Parameter Contents	
0010 C000h	
1180 1800h	
0080h	0001h
01D0 0004h	
0000h	0001h
0080h	4860h
0000h	0000h
0000h	0001h

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

Figure 17-31. Ping-Pong Buffering for McBSP Example Ping PaRAM

(a) EDMA Ping Parameters for Channel 3 at Set 65 Linked to Set 64

Parameter Contents	
0010 D000h	
01D0 0000h	
0080h	0001h
1180 0000h	
0001h	0000h
0080h	4800h
0000h	0000h
0000h	0001h

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

(b) EDMA Ping Parameters for Channel 2 at Set 67 Linked to Set 66

Parameter Contents	
0010 C000h	
1180 1000h	
0080h	0001h
01D0 0004h	
0000h	0001h
0080h	4840h
0000h	0000h
0000h	0001h

Parameter	
Channel Options Parameter (OPT)	
Channel Source Address (SRC)	
Count for 2nd Dimension (BCNT)	Count for 1st Dimension (ACNT)
Channel Destination Address (DST)	
Destination BCNT Index (DSTBIDX)	Source BCNT Index (SRCBIDX)
BCNT Reload (BCNTRLD)	Link Address (LINK)
Destination CCNT Index (DSTCIDX)	Source CCNT Index (SRCCIDX)
Reserved	Count for 3rd Dimension (CCNT)

Copyright © 2013–2016, Texas Instruments Incorporated

17.3.4.5 Transfer Chaining Examples

The following examples explain the intermediate transfer complete chaining function.

17.3.4.5.1 Servicing Input/Output FIFOs with a Single Event

Many systems require the use of a pair of external FIFOs that must be serviced at the same rate. One FIFO buffers data input, and the other buffers data output. The EDMA3 channels that service these FIFOs can be set up for AB-synchronized transfers. While each FIFO is serviced with a different set of parameters, both can be signaled from a single event. For example, an external interrupt pin can be tied to the status flags of one of the FIFOs. When this event arrives, the EDMA3 needs to perform servicing for both the input and output streams. Without the intermediate transfer complete chaining feature this would require two events, and thus two external interrupt pins. The intermediate transfer complete chaining feature allows the use of a single external event (for example, a GPIO event). Figure 17-32 shows the EDMA3 setup and illustration for this example.

A GPIO event (in this case, GPINT0) triggers an array transfer. Upon completion of each intermediate array transfer of channel 16, intermediate transfer complete chaining sets the E31 bit (specified by TCC of 31) in the chained event register (CER) and provides a synchronization event to channel 31. Upon completion of the last array transfer of channel 16, transfer complete chaining—not intermediate transfer complete chaining—sets the E31 bit in CER (specified by TCCMODE:TCC) and provides a synchronization event to channel 31. The completion of channel 31 sets the I31 bit (specified by TCCMODE:TCC) in the interrupt pending register (IPR), which can generate an interrupt to the CPU, if the I31 bit in the interrupt enable register (IER) is set to 1.

17.3.4.5.2 Breaking Up Large Transfers with Intermediate Chaining

Another feature of intermediate transfer chaining (ITCCHEN) is for breaking up large transfers. A large transfer may lock out other transfers of the same priority level for the duration of the transfer. For example, a large transfer on queue 0 from the internal memory to the external memory using the EMIF may starve other EDMA3 transfers on the same queue. In addition, this large high-priority transfer may prevent the EMIF for a long duration to service other lower priority transfers. When a large transfer is considered to be high priority, it should be split into multiple smaller transfers. Figure 17-33 shows the EDMA3 setup and illustration of an example single large block transfer.

The intermediate transfer chaining enable (ITCCHEN) provides a method to break up a large transfer into smaller transfers. For example, to move a single large block of memory (16K bytes), the EDMA3 performs an A-synchronized transfer. The element count is set to a reasonable value, where reasonable derives from the amount of time it would take to move this smaller amount of data. Assume 1K byte is a reasonable small transfer in this example. The EDMA3 is set up to transfer 16 arrays of 1K byte elements, for a total of 16K byte elements. The TCC field in the channel options parameter (OPT) is set to the same value as the channel number and ITCCHEN are set. In this example, DMA channel 25 is used and TCC is also set to 25. The TCINTEN may also be set to trigger interrupt 25 when the last 1K byte array is transferred. The CPU starts the EDMA3 transfer by writing to the appropriate bit of the event set register (ESR.E25). The EDMA3 transfers the first 1K byte array. Upon completion of the first array, intermediate transfer complete code chaining generates a synchronization event to channel 25, a value specified by the TCC field. This intermediate transfer completion chaining event causes DMA channel 25 to transfer the next 1K byte array. This process continues until the transfer parameters are exhausted, at which point the EDMA3 has completed the 16K byte transfer. This method breaks up a large transfer into smaller packets, thus providing natural time slices in the transfer such that other events may be processed. Figure 17-34 shows the EDMA3 setup and illustration of the broken up smaller packet transfers.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 17-32. Intermediate Transfer Completion Chaining Example

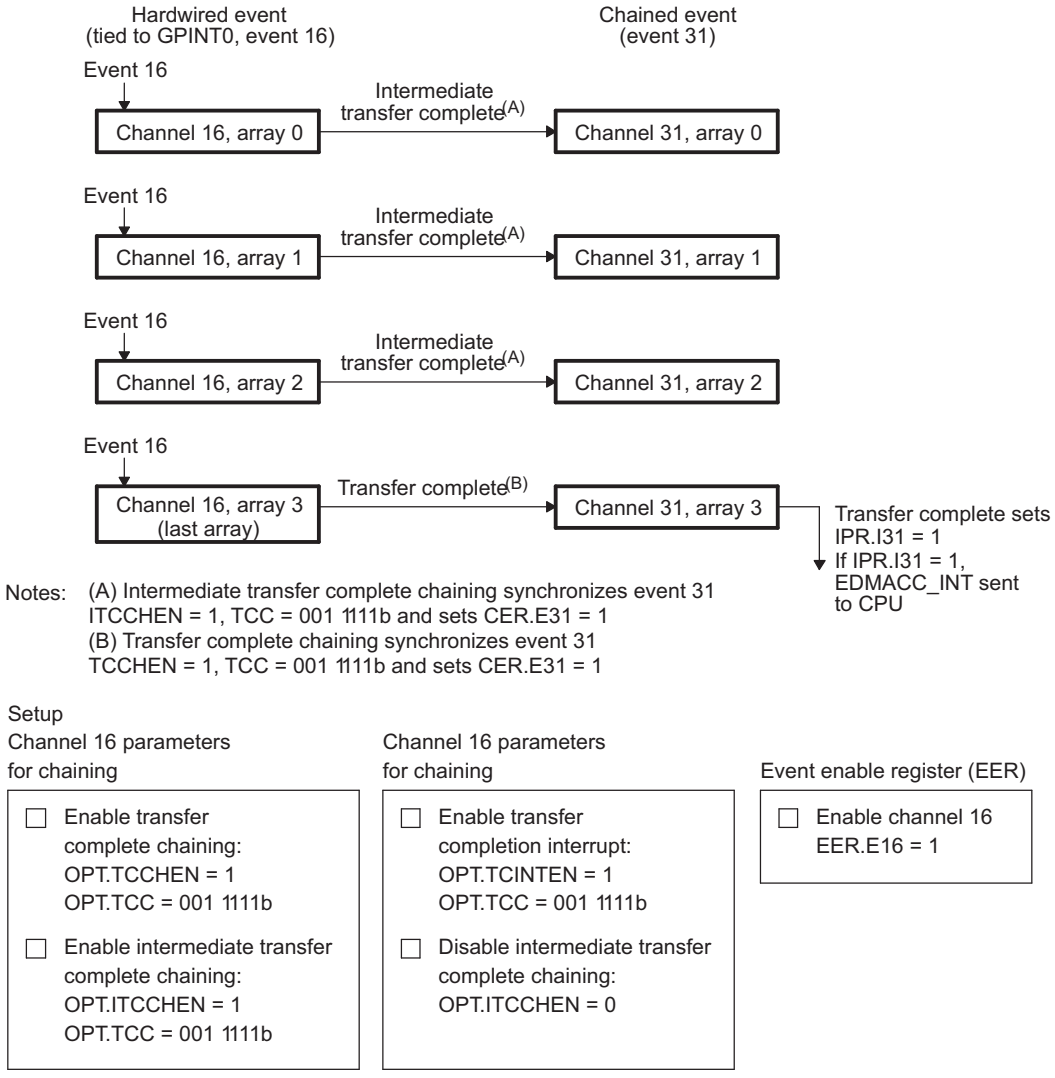
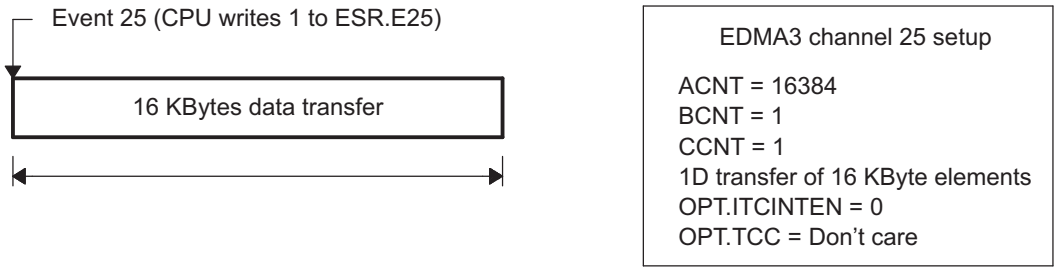
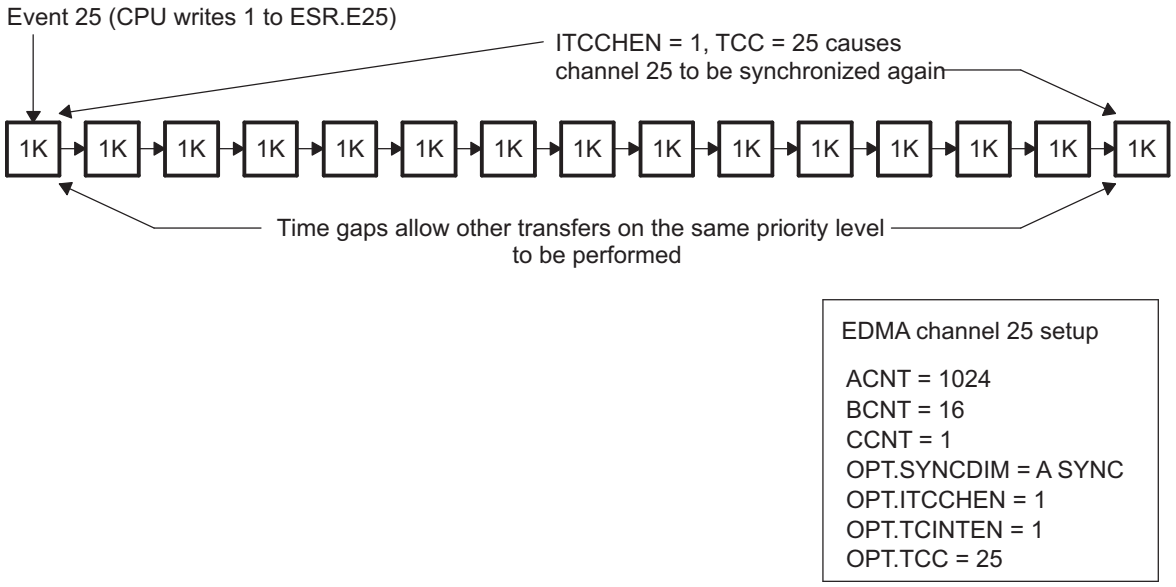


Figure 17-33. Single Large Block Transfer Example



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 17-34. Smaller Packet Data Transfers Example



17.4 Registers

This section discusses the registers of the EDMA3 controller.

17.4.1 Parameter RAM (PaRAM) Entries

Table 17-14 lists the parameter RAM (PaRAM) entries for the EDMA3 channel controller (EDMA3CC). See your device-specific data manual for the memory address of these registers.

Table 17-14. EDMA3 Channel Controller (EDMA3CC) Parameter RAM (PaRAM) Entries

Offset	Acronym	Parameter	Section
0h	OPT	Channel Options	Section 17.4.1.1
4h	SRC	Channel Source Address	Section 17.4.1.2
8h	A_B_CNT	A Count/B Count	Section 17.4.1.3
Ch	DST	Channel Destination Address	Section 17.4.1.4
10h	SRC_DST_BIDX	Source B Index/Destination B Index	Section 17.4.1.5
14h	LINK_BCNTRLD	Link Address/B Count Reload	Section 17.4.1.6
18h	SRC_DST_CIDX	Source C Index/Destination C Index	Section 17.4.1.7
1Ch	CCNT	C Count	Section 17.4.1.8

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.1.1 Channel Options Parameter (OPT)

The channel options parameter (OPT) is shown in Figure 17-35 and described in Table 17-15.

NOTE: The TCC field in OPT is a 6-bit field and can be programmed for any value between 0-64. For devices with 32 DMA channels, the TCC field should have a value between 0 to 31 so that it sets the appropriate bits (0 to 31) in the interrupt pending register (IPR) (and can interrupt the CPU(s) on enabling the interrupt enable register (IER) bits (0-31)).

Figure 17-35. Channel Options Parameter (OPT)

31–28	27–24	23	22	21	20	19–18	17–16
Reserved	PRIVID	ITCCHEN	TCCHEN	ITCINTEN	TCINTEN	Reserved	TCC
R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0 R-0	R/W-0

15–12	11–10	9–8	7–4	3	2	1	0
TCC	TCCMOD	FWID	Reserved	STATIC	SYNCDIM	DAM	SAM
R/W-0	R/W-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 17-15. Channel Options Parameters (OPT) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reserved
27-24	PRIVID	0-Fh	Privilege identification for the external host/CPU/DMA that programmed this PaRAM set. This value is set with the EDMA3 master's privilege identification value when any part of the PaRAM set is written.
23	ITCCHEN		Intermediate transfer completion chaining enable.
		0	Intermediate transfer complete chaining is disabled.
		1	Intermediate transfer complete chaining is enabled.
			When enabled, the chained event register (CER) bit is set on every intermediate chained transfer completion (upon completion of every intermediate TR in the PaRAM set, except the final TR in the PaRAM set). The bit (position) set in CER is the TCC value specified.
22	TCCHEN		Transfer complete chaining enable.
		0	Transfer complete chaining is disabled.
		1	Transfer complete chaining is enabled.
			When enabled, the chained event register (CER) bit is set on final chained transfer completion (upon completion of the final TR in the PaRAM set). The bit (position) set in CER is the TCC value specified.
21	ITCINTEN		Intermediate transfer completion interrupt enable.
		0	Intermediate transfer complete interrupt is disabled.
		1	Intermediate transfer complete interrupt is enabled.
			When enabled, the interrupt pending register (IPR) bit is set on every intermediate transfer completion (upon completion of every intermediate TR in the PaRAM set, except the final TR in the PaRAM set). The bit (position) set in IPR is the TCC value specified. In order to generate a completion interrupt to the CPU, the corresponding IER[TCC] bit must be set to 1.
20	TCINTEN		Transfer complete interrupt enable.
		0	Transfer complete interrupt is disabled.
		1	Transfer complete interrupt is enabled.
			When enabled, the interrupt pending register (IPR) bit is set on transfer completion (upon completion of the final TR in the PaRAM set). The bit (position) set in IPR is the TCC value specified. In order to generate a completion interrupt to the CPU, the corresponding IER[TCC] bit must be set to 1.
19	Reserved	0	Reserved. Always write 0 to this bit.
18	Reserved	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

Table 17-15. Channel Options Parameters (OPT) Field Descriptions (continued)

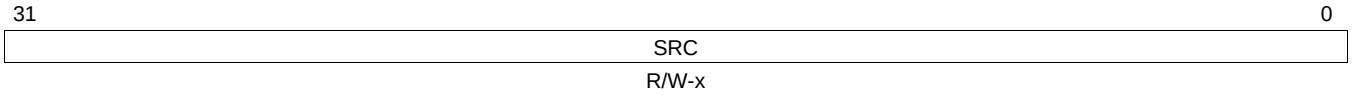
Bit	Field	Value	Description
17-12	TCC	0-3Fh	Transfer complete code. This 6-bit code is used to set the relevant bit in chaining enable register (CER[TCC]) for chaining or in interrupt pending register (IPR[TCC]) for interrupts.
		0-1Fh	Valid values
		20h-3Fh	Reserved
11	TCCMODE		Transfer complete code mode. Indicates the point at which a transfer is considered completed for chaining and interrupt generation.
		0	Normal completion: A transfer is considered completed after the data has been transferred.
		1	Early completion: A transfer is considered completed after the EDMA3CC submits a TR to the EDMA3TC. TC may still be transferring data when interrupt/chain is triggered.
10-8	FWID	0-7h	FIFO Width. Applies if either SAM or DAM is set to constant addressing mode.
		0	FIFO width is 8-bit.
		1h	FIFO width is 16-bit.
		2h	FIFO width is 32-bit.
		3h	FIFO width is 64-bit.
		4h	FIFO width is 128-bit.
		5h	FIFO width is 256-bit.
		6h-7h	Reserved
7-4	Reserved	0	Reserved
3	STATIC		Static PaRAM set.
		0	PaRAM set is not static. PaRAM set is updated or linked after TR is submitted. A value of 0 should be used for DMA channels and for nonfinal transfers in a linked list of QDMA transfers.
		1	PaRAM set is static. PaRAM set is not updated or linked after TR is submitted. A value of 1 should be used for isolated QDMA transfers or for the final transfer in a linked list of QDMA transfers.
2	SYNCDIM		Transfer synchronization dimension.
		0	A-synchronized. Each event triggers the transfer of a single array of ACNT bytes.
		1	AB-synchronized. Each event triggers the transfer of BCNT arrays of ACNT bytes.
1	DAM		Destination address mode.
		0	Increment (INCR) mode. Destination addressing within an array increments. Destination is not a FIFO.
		1	Constant addressing (CONST) mode. Destination addressing within an array wraps around upon reaching FIFO width. Note: The constant addressing (CONST) mode has limited applicability. The EDMA3 should be configured for the constant addressing mode (SAM/DAM = 1) only if the transfer source or destination (on-chip memory, off-chip memory controllers, slave peripherals) support the constant addressing mode. See your device-specific data manual to verify if constant addressing mode is supported. If the constant addressing mode is not supported, the similar logical transfer can be achieved using the increment (INCR) mode (SAM/DAM = 0) by appropriately programming the count and indices values.
0	SAM		Source address mode.
		0	Increment (INCR) mode. Source addressing within an array increments. Source is not a FIFO.
		1	Constant addressing (CONST) mode. Source addressing within an array wraps around upon reaching FIFO width. Note: The constant addressing (CONST) mode has limited applicability. The EDMA3 should be configured for the constant addressing mode (SAM/DAM = 1) only if the transfer source or destination (on-chip memory, off-chip memory controllers, slave peripherals) support the constant addressing mode. See your device-specific data manual to verify if constant addressing mode is supported. If the constant addressing mode is not supported, the similar logical transfer can be achieved using the increment (INCR) mode (SAM/DAM = 0) by appropriately programming the count and indices values.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.1.2 Channel Source Address Parameter (SRC)

The channel source address parameter (SRC) specifies the starting byte address of the source. The SRC is shown in Figure 17-36 and described in Table 17-16.

Figure 17-36. Channel Source Address Parameter (SRC)



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

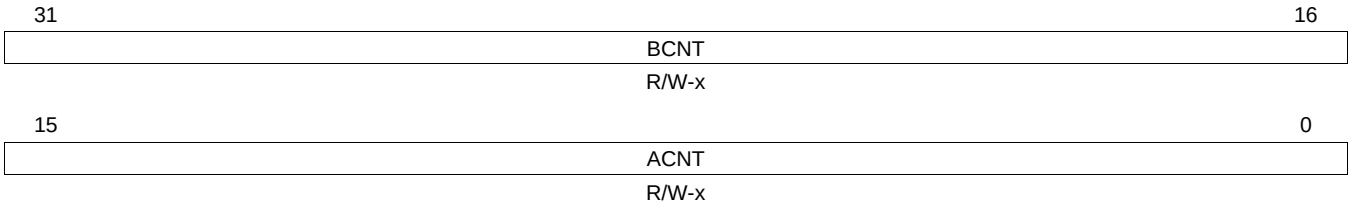
Table 17-16. Channel Source Address Parameter (SRC) Field Descriptions

Bit	Field	Value	Description
31-0	SRC	0-FFFF FFFFh	Source address. Specifies the starting byte address of the source.

17.4.1.3 A Count/B Count Parameter (A_B_CNT)

The A count/B count parameter (A_B_CNT) specifies the number of bytes within the 1st dimension of a transfer and the number of arrays of length ACNT. The A_B_CNT is shown in Figure 17-37 and described in Table 17-17.

Figure 17-37. A Count/B Count Parameter (A_B_CNT)



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

Table 17-17. A Count/B Count Parameter (A_B_CNT) Field Descriptions

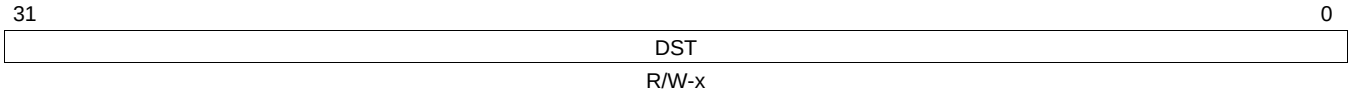
Bit	Field	Value	Description
31-16	BCNT	0-FFFFh	B count. Unsigned value specifying the number of arrays in a frame, where an array is ACNT bytes. Valid values range from 1 to 65 535.
15-0	ACNT	0-FFFFh	A count for 1st Dimension. Unsigned value specifying the number of contiguous bytes within an array (first dimension of the transfer). Valid values range from 1 to 65 535.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.1.4 Channel Destination Address Parameter (DST)

The channel destination address parameter (DST) specifies the starting byte address of the source. The DST is shown in Figure 17-38 and described in Table 17-18.

Figure 17-38. Channel Destination Address Parameter (DST)



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

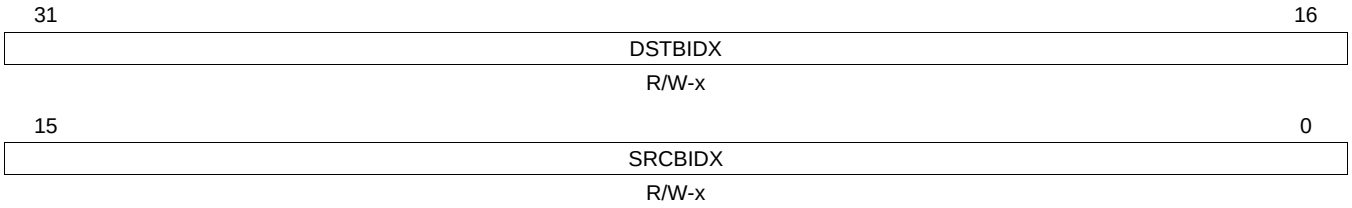
Table 17-18. Channel Destination Address Parameter (DST) Field Descriptions

Bit	Field	Value	Description
31-0	DST	0-FFFF FFFFh	Destination address. Specifies the starting byte address of the destination where data is transferred.

17.4.1.5 Source B Index/Destination B Index Parameter (SRC_DST_BIDX)

The source B index/destination B index parameter (SRC_DST_BIDX) specifies the value (2s complement) used for source address modification between each array in the 2nd dimension and the value (2s complement) used for destination address modification between each array in the 2nd dimension. The SRC_DST_BIDX is shown in Figure 17-39 and described in Table 17-19.

Figure 17-39. Source B Index/Destination B Index Parameter (SRC_DST_BIDX)



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

Table 17-19. Source B Index/Destination B Index Parameter (SRC_DST_BIDX) Field Descriptions

Bit	Field	Value	Description
31-16	DSTBIDX	0-FFFFh	Destination B index. Signed value specifying the byte address offset between destination arrays within a frame (2nd dimension). Valid values range from –32 768 and 32 767.
15-0	SRCBIDX	0-FFFFh	Source B index. Signed value specifying the byte address offset between source arrays within a frame (2nd dimension). Valid values range from –32 768 and 32 767.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.1.6 Link Address/B Count Reload Parameter (LINK_BCNTRLD)

The link address/B count reload parameter (LINK_BCNTRLD) specifies the byte address offset in the PaRAM from which the EDMA3CC loads/reloads the next PaRAM set during linking and the value used to reload the BCNT field in the A count/B count parameter (A_B_CNT) once the last array in the 2nd dimension is transferred. The LINK_BCNTRLD is shown in Figure 17-40 and described in Table 17-20.

Figure 17-40. Link Address/B Count Reload Parameter (LINK_BCNTRLD)

31	16
BCNTRLD	
R/W-x	

15	0
LINK	
R/W-x	

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

Table 17-20. Link Address/B Count Reload Parameter (LINK_BCNTRLD) Field Descriptions

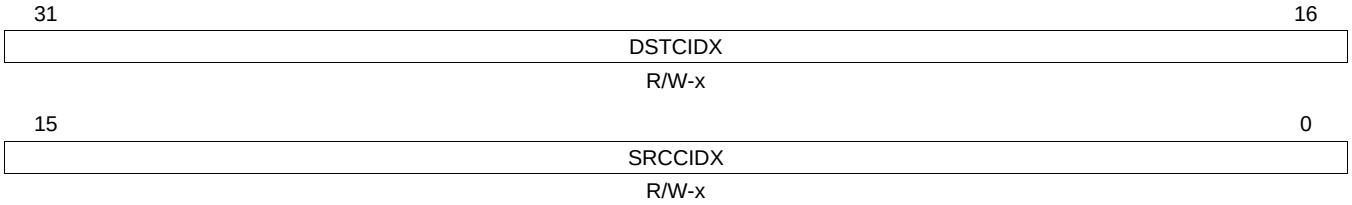
Bit	Field	Value	Description
31-16	BCNTRLD	0-FFFFh	B count reload. The count value used to reload BCNT in the A count/B count parameter (A_B_CNT) when BCNT decrements to 0 (TR submitted for the last array in 2nd dimension). Only relevant in A-synchronized transfers.
15-0	LINK	0-FFFFh	Link address. The PaRAM address containing the PaRAM set to be linked (copied from) when the current PaRAM set is exhausted. You must program the link address to point to a valid aligned 32-byte PaRAM set. The 5 LSBs of the LINK field should be cleared to 0. A value of FFFFh specifies a null link.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.1.7 Source C Index/Destination C Index Parameter (SRC_DST_CIDX)

The source C index/destination C index parameter (SRC_DST_CIDX) specifies the value (2s complement) used for source address modification between each array in the 3rd dimension and the value (2s complement) used for destination address modification between each array in the 3rd dimension. The SRC_DST_CIDX is shown in Figure 17-41 and described in Table 17-21.

Figure 17-41. Source C Index/Destination C Index Parameter (SRC_DST_CIDX)



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

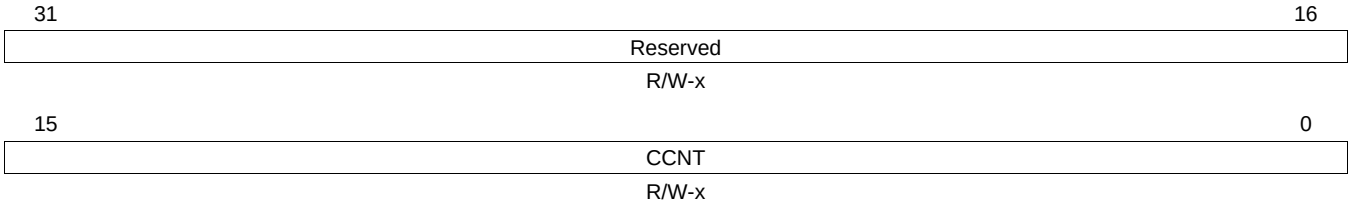
Table 17-21. Source C Index/Destination C Index Parameter (SRC_DST_CIDX) Field Descriptions

Bit	Field	Value	Description
31-16	DSTCIDX	0-FFFFh	Destination C index. Signed value specifying the byte address offset between frames within a block (3rd dimension). Valid values range from –32 768 and 32 767.
15-0	SRCCIDX	0-FFFFh	Source C index. Signed value specifying the byte address offset between frames within a block (3rd dimension). Valid values range from –32 768 and 32 767.

17.4.1.8 C Count Parameter (CCNT)

The C count parameter (CCNT) specifies the number of frames in a block. The CCNT is shown in Figure 17-42 and described in Table 17-22.

Figure 17-42. C Count Parameter (CCNT)



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

Table 17-22. C Count Parameter (CCNT) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	CCNT	0-FFFFh	C counter. Unsigned value specifying the number of frames in a block, where a frame is BCNT arrays of ACNT bytes. Valid values range from 1 to 65 535.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2 EDMA3 Channel Controller (EDMA3CC) Registers

Table 17-23 lists the memory-mapped registers for the EDMA3 channel controller (EDMA3CC). See your device-specific data manual for the memory address of these registers and for the shadow region addresses. All other register offset addresses not listed in Table 17-23 should be considered as reserved locations and the register contents should not be modified.

Table 17-23. EDMA3 Channel Controller (EDMA3CC) Registers

Offset	Acronym	Register Description	Section
0h	REVID	Revision Identification Register	Section 17.4.2.1.1
4h	CCCFG	EDMA3CC Configuration Register	Section 17.4.2.1.2
		Global Registers	
200h	QCHMAP0	QDMA Channel 0 Mapping Register	Section 17.4.2.1.3
204h	QCHMAP1	QDMA Channel 1 Mapping Register	Section 17.4.2.1.3
208h	QCHMAP2	QDMA Channel 2 Mapping Register	Section 17.4.2.1.3
20Ch	QCHMAP3	QDMA Channel 3 Mapping Register	Section 17.4.2.1.3
210h	QCHMAP4	QDMA Channel 4 Mapping Register	Section 17.4.2.1.3
214h	QCHMAP5	QDMA Channel 5 Mapping Register	Section 17.4.2.1.3
218h	QCHMAP6	QDMA Channel 6 Mapping Register	Section 17.4.2.1.3
21Ch	QCHMAP7	QDMA Channel 7 Mapping Register	Section 17.4.2.1.3
240h	DMAQNUM0	DMA Channel Queue Number Register 0	Section 17.4.2.1.4
244h	DMAQNUM1	DMA Channel Queue Number Register 1	Section 17.4.2.1.4
248h	DMAQNUM2	DMA Channel Queue Number Register 2	Section 17.4.2.1.4
24Ch	DMAQNUM3	DMA Channel Queue Number Register 3	Section 17.4.2.1.4
260h	QDMAQNUM	QDMA Channel Queue Number Register	Section 17.4.2.1.5
284h	QUEPRI	Queue Priority Register(1)	Section 17.4.2.1.6
300h	EMR	Event Missed Register	Section 17.4.2.2.1
308h	EMCR	Event Missed Clear Register	Section 17.4.2.2.2
310h	QEMR	QDMA Event Missed Register	Section 17.4.2.2.3
314h	QEMCR	QDMA Event Missed Clear Register	Section 17.4.2.2.4
318h	CCERR	EDMA3CC Error Register	Section 17.4.2.2.5
31Ch	CCERRCLR	EDMA3CC Error Clear Register	Section 17.4.2.2.6
320h	EEVAL	Error Evaluate Register	Section 17.4.2.2.7
340h	DRAE0	DMA Region Access Enable Register for Region 0	Section 17.4.2.3.1
348h	DRAE1	DMA Region Access Enable Register for Region 1	Section 17.4.2.3.1
350h	DRAE2	DMA Region Access Enable Register for Region 2	Section 17.4.2.3.1
358h	DRAE3	DMA Region Access Enable Register for Region 3	Section 17.4.2.3.1
380h	QRAE0	QDMA Region Access Enable Register for Region 0	Section 17.4.2.3.2
384h	QRAE1	QDMA Region Access Enable Register for Region 1	Section 17.4.2.3.2
388h	QRAE2	QDMA Region Access Enable Register for Region 2	Section 17.4.2.3.2
38Ch	QRAE3	QDMA Region Access Enable Register for Region 3	Section 17.4.2.3.2
400h-43Ch	Q0E0-Q0E15	Event Queue Entry Registers Q0E0-Q0E15	Section 17.4.2.4.1
440h-47Ch	Q1E0-Q1E15	Event Queue Entry Registers Q1E0-Q1E15	Section 17.4.2.4.1
600h	QSTAT0	Queue 0 Status Register	Section 17.4.2.4.2
604h	QSTAT1	Queue 1 Status Register	Section 17.4.2.4.2
620h	QWMTHRA	Queue Watermark Threshold A Register	Section 17.4.2.4.3
640h	CCSTAT	EDMA3CC Status Register	Section 17.4.2.4.4

(1) On previous architectures, the EDMA3TC priority was controlled by the queue priority register (QUEPRI) in the EDMA3CC memory-map. However for this device, the priority control for the transfer controllers is controlled by the chip-level registers in the System Configuration Module. You should use the chip-level registers and not QUEPRI to configure the TC priority.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 17-23. EDMA3 Channel Controller (EDMA3CC) Registers (continued)

Offset	Acronym	Register Description	Section
		Global Channel Registers	
1000h	ER	Event Register	Section 17.4.2.5.1
1008h	ECR	Event Clear Register	Section 17.4.2.5.2
1010h	ESR	Event Set Register	Section 17.4.2.5.3
1018h	CER	Chained Event Register	Section 17.4.2.5.4
1020h	EER	Event Enable Register	Section 17.4.2.5.5
1028h	EECR	Event Enable Clear Register	Section 17.4.2.5.6
1030h	EESR	Event Enable Set Register	Section 17.4.2.5.7
1038h	SER	Secondary Event Register	Section 17.4.2.5.8
1040h	SECR	Secondary Event Clear Register	Section 17.4.2.5.9
1050h	IER	Interrupt Enable Register	Section 17.4.2.6.1
1058h	IECR	Interrupt Enable Clear Register	Section 17.4.2.6.2
1060h	IESR	Interrupt Enable Set Register	Section 17.4.2.6.3
1068h	IPR	Interrupt Pending Register	Section 17.4.2.6.4
1070h	ICR	Interrupt Clear Register	Section 17.4.2.6.5
1078h	IEVAL	Interrupt Evaluate Register	Section 17.4.2.6.6
1080h	QER	QDMA Event Register	Section 17.4.2.7.1
1084h	QEER	QDMA Event Enable Register	Section 17.4.2.7.2
1088h	QEECR	QDMA Event Enable Clear Register	Section 17.4.2.7.3
108Ch	QEESR	QDMA Event Enable Set Register	Section 17.4.2.7.4
1090h	QSER	QDMA Secondary Event Register	Section 17.4.2.7.5
1094h	QSECR	QDMA Secondary Event Clear Register	Section 17.4.2.7.6
		Shadow Region 0 Channel Registers	
2000h	ER	Event Register	—
2008h	ECR	Event Clear Register	—
2010h	ESR	Event Set Register	—
2018h	CER	Chained Event Register	—
2020h	EER	Event Enable Register	—
2028h	EECR	Event Enable Clear Register	—
2030h	EESR	Event Enable Set Register	—
2038h	SER	Secondary Event Register	—
2040h	SECR	Secondary Event Clear Register	—
2050h	IER	Interrupt Enable Register	—
2058h	IECR	Interrupt Enable Clear Register	—
2060h	IESR	Interrupt Enable Set Register	—
2068h	IPR	Interrupt Pending Register	—
2070h	ICR	Interrupt Clear Register	—
2078h	IEVAL	Interrupt Evaluate Register	—
2080h	QER	QDMA Event Register	—
2084h	QEER	QDMA Event Enable Register	—
2088h	QEECR	QDMA Event Enable Clear Register	—
208Ch	QEESR	QDMA Event Enable Set Register	—
2090h	QSER	QDMA Secondary Event Register	—
2094h	QSECR	QDMA Secondary Event Clear Register	—

Copyright © 2013–2016, Texas Instruments Incorporated

Table 17-23. EDMA3 Channel Controller (EDMA3CC) Registers (continued)

Offset	Acronym	Register Description	Section
		Shadow Region 1 Channel Registers	
2200h	ER	Event Register	—
2208h	ECR	Event Clear Register	—
2210h	ESR	Event Set Register	—
2218h	CER	Chained Event Register	—
2220h	EER	Event Enable Register	—
2228h	EECR	Event Enable Clear Register	—
2230h	EESR	Event Enable Set Register	—
2238h	SER	Secondary Event Register	—
2240h	SECR	Secondary Event Clear Register	—
2250h	IER	Interrupt Enable Register	—
2258h	IECR	Interrupt Enable Clear Register	—
2260h	IESR	Interrupt Enable Set Register	—
2268h	IPR	Interrupt Pending Register	—
2270h	ICR	Interrupt Clear Register	—
2278h	IEVAL	Interrupt Evaluate Register	—
2280h	QER	QDMA Event Register	—
2284h	QEER	QDMA Event Enable Register	—
2288h	QEECR	QDMA Event Enable Clear Register	—
228Ch	QEESR	QDMA Event Enable Set Register	—
2290h	QSER	QDMA Secondary Event Register	—
2294h	QSECR	QDMA Secondary Event Clear Register	—
4000h-4FFFh	—	Parameter RAM (PaRAM)	—

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.1 Global Registers

17.4.2.1.1 Revision Identification Register (REVID)

The revision identification register (REVID) uniquely identifies the EDMA3CC and the specific revision of the EDMA3CC. The REVID is shown in Figure 17-43 and described in Table 17-24.

Figure 17-43. Revision ID Register (REVID)

31	0
REV	
R-4001 5300h	

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

Table 17-24. Revision ID Register (REVID) Field Descriptions

Bit	Field	Value	Description
31-0	REV		Peripheral identifier.
		4001 5300h	Uniquely identifies the EDMA3CC and the specific revision of the EDMA3CC.

17.4.2.1.2 EDMA3CC Configuration Register (CCCFG)

The EDMA3CC configuration register (CCCFG) provides the features/resources for the EDMA3CC in a particular device. The CCCFG is shown in Figure 17-44 and described in Table 17-25.

Figure 17-44. EDMA3CC Configuration Register (CCCFG)

31	26	25	24
Reserved		MP_EXIST	CHMAP_EXIST
R-x		R-0	R-0

23	22	21	20	19	18	16
Reserved		NUM_REGN		Reserved	NUM_EVQUE	
R-0		R-2h		R-x	R-1h	

15	14	12	11	10	8
Reserved	NUM_PAENTRY		Reserved	NUM_INTCH	
R-x	R-3h		R-x	R-3h	

7	6	4	3	2	0
Reserved	NUM_QDMACH		Reserved	NUM_DMACH	
R-x	R-4h		R-x	R-4h	

LEGEND: R = Read only; -*n* = value after reset; -x = value is indeterminate after reset

Copyright © 2013–2016, Texas Instruments Incorporated

Table 17-25. EDMA3CC Configuration Register (CCCFG) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0-3Fh	Reserved
25	MP_EXIST		Memory protection existence.
		0	No memory protection.
		1	Reserved
24	CHMAP_EXIST		Channel mapping existence.
		0	No channel mapping. This implies that there is fixed association for a channel number to a parameter entry number or, in other words, PaRAM entry *n* corresponds to channel *n*.
		1	Reserved
23-22	Reserved	0	Reserved
21-20	NUM_REGN	0-3h	Number of shadow regions.
		0-1h	Reserved
		2h	4 regions
		3h	Reserved
19	Reserved	0	Reserved
18-16	NUM_EVQUE	0-7h	Number of queues/number of transfer controllers.
		0	Reserved
		1h	2 event queues
		2h	2 transfer controllers
		3h-7h	Reserved
15	Reserved	0	Reserved
14-12	NUM_PAENTRY	0-7h	Number of PaRAM sets.
		0-2h	Reserved
		3h	128 PaRAM sets
		4h-7h	Reserved
11	Reserved	0	Reserved
10-8	NUM_INTCH	0-7h	Number of interrupt channels.
		0-2h	Reserved
		3h	32 interrupt channels
		4h-7h	Reserved
7	Reserved	0	Reserved
6-4	NUM_QDMACH	0-7h	Number of QDMA channels.
		0-3h	Reserved
		4h	8 QDMA channels
		5h-7h	Reserved
3	Reserved	0	Reserved
2-0	NUM_DMACH	0-7h	Number of DMA channels.
		0-3h	Reserved
		4h	32 DMA channels
		5h-7h	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.1.3 QDMA Channel *n* Mapping Register (QCHMAP*n*)

Each QDMA channel in EDMA3CC can be associated with any PaRAM set available on the device. Furthermore, the specific trigger word (0-7) of the PaRAM set can be programmed. The PaRAM set association and trigger word for every QDMA channel register is configurable using the QDMA channel *n* mapping register (QCHMAP*n*). The QCHMAP*n* is shown in Figure 17-45 and described in Table 17-26.

NOTE: At reset the QDMA channel mapping registers for all QDMA channels point to the PaRAM set 0. Prior to using any QDMA channel, QCHMAP*n* should be programmed appropriately to point to a different PaRAM set.

Figure 17-45. QDMA Channel *n* Mapping Register (QCHMAP*n*)

31	16
Reserved	
R-0	

15	14	13	5	4	2	1	0
Reserved		PAENTRY		TRWORD		Reserved	
R-0		R/W-0		R/W-0		R-0	

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

Table 17-26. QDMA Channel *n* Mapping Register (QCHMAP*n*) Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Reserved
13-5	PAENTRY	0-1FFh	PAENTRY points to the PaRAM set number for QDMA channel *n*.
		0-7Fh	PaRAM set number 0 through 127
		80h-1FFh	Reserved
4-2	TRWORD	0-7h	Points to the specific PaRAM entry or the trigger word in the PaRAM set pointed to by PAENTRY. A write to the trigger word results in a QDMA event being recognized.
1-0	Reserved	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.1.4 DMA Channel Queue Number Register *n* (DMAQNUM*n*)

The DMA channel queue number register *n* (DMAQNUM*n*) allows programmability of each of the 32 DMA channels in the EDMA3CC to submit its associated synchronization event to any event queue in the EDMA3CC. At reset, all channels point to event queue 0. The DMAQNUM*n* is shown in Figure 17-46 and described in . Table 17-28 shows the channels and their corresponding bits in DMAQNUM*n*.

NOTE: Since the event queues in EDMA3CC have a fixed association to the transfer controllers, that is, Q0 TRs are submitted to TC0 and Q1 TRs are submitted to TC1, by programming DMAQNUM*n* for a particular DMA channel also dictates which transfer controller is utilized for the data movement (or which EDMA3TC receives the TR request).

Figure 17-46. DMA Channel Queue Number Register *n* (DMAQNUM*n*)

31	30 28	27	26 24	23	22 20	19	18 16
Rsvd	E*n*	Rsvd	E*n*	Rsvd	E*n*	Rsvd	E*n*
R-0	R/W-0	R-0	R/W-0	R-0	R/W-0	R-0	R/W-0

15	14 12	11	10 8	7	6 4	3	2 0
Rsvd	E*n*	Rsvd	E*n*	Rsvd	E*n*	Rsvd	E*n*
R-0	R/W-0	R-0	R/W-0	R-0	R/W-0	R-0	R/W-0

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

Table 17-27. DMA Channel Queue Number Register *n* (DMAQNUM*n*) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*	0-7h	DMA queue number. Contains the event queue number to be used for the corresponding DMA channel. Programming DMAQNUM*n* for an event queue number to a value more then the number of queues available in the EDMA3CC results in undefined behavior.
		0	Event *n* is queued on Q0.
		1h	Event *n* is queued on Q1. Available only for EDMA3_0_CC0; for EDMA3_1_CC0, this value is reserved and all events are queued on Q0.
		2h-7h	Reserved

Table 17-28. Bits in DMAQNUM*n*

	DMAQNUM*n*			
E*n* bit	**0**	**1**	**2**	**3**
0-2	E0	E8	E16	E24
4-6	E1	E9	E17	E25
8-10	E2	E10	E18	E26
12-14	E3	E11	E19	E27
16-18	E4	E12	E20	E28
20-22	E5	E13	E21	E29
24-26	E6	E14	E22	E30
28-30	E7	E15	E23	E31

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.1.5 QDMA Channel Queue Number Register (QDMAQNUM)

The QDMA channel queue number register (QDMAQNUM) is used to program all the QDMA channels in the EDMA3CC to submit the associated QDMA event to any of the event queues in the EDMA3CC. The QDMAQNUM is shown in Figure 17-47 and described in .

Figure 17-47. QDMA Channel Queue Number Register (QDMAQNUM)

31	30 28	27	26 24	23	22 20	19	18 16
Rsvd	E7	Rsvd	E6	Rsvd	E5	Rsvd	E4
R-0	R/W-0	R-0	R/W-0	R-0	R/W-0	R-0	R/W-0

15	14 12	11	10 8	7	6 4	3	2 0
Rsvd	E3	Rsvd	E2	Rsvd	E1	Rsvd	E0
R-0	R/W-0	R-0	R/W-0	R-0	R/W-0	R-0	R/W-0

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

Table 17-29. QDMA Channel Queue Number Register (QDMAQNUM) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*	0-7h	QDMA queue number. Contains the event queue number to be used for the corresponding QDMA channel.
		0	Event *n* is queued on Q0.
		1h	Event *n* is queued on Q1. Available only for EDMA3_0_CC0; for EDMA3_1_CC0, this value is reserved and all events are queued on Q0.
		2h-7h	Reserved

17.4.2.1.6 Queue Priority Register (QUEPRI)

On previous architectures, the EDMA3TC priority was controlled by the queue priority register (QUEPRI) in the EDMA3CC memory-map. However for this device, the priority control for the transfer controllers is controlled by the chip-level registers in the System Configuration Module. You should use the chip-level registers and not QUEPRI to configure the TC priority.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.2 Error Registers

The EDMA3CC contains a set of registers that provide information on missed DMA and/or QDMA events, and instances when event queue thresholds are exceeded. If any of the bits in these registers is set, it results in the EDMA3CC generating an error interrupt.

17.4.2.2.1 Event Missed Registers (EMR)

For a particular DMA channel, if a second event is received prior to the first event getting cleared/serviced, the bit corresponding to that channel is set/asserted in the event missed register (EMR). All trigger types are treated individually, that is, manual triggered (ESR), chain triggered (CER), and event triggered (ER) are all treated separately. The EMR bit for a channel is also set if an event on that channel encounters a NULL entry (or a NULL TR is serviced). If any EMR bit is set (and all errors, including bits in other error registers (QEMR, CCERR) were previously cleared), the EDMA3CC generates an error interrupt. See Section 17.2.9.4 for details on EDMA3CC error interrupt generation.

The EMR is shown in Figure 17-48 and described in Table 17-30.

Figure 17-48. Event Missed Register (EMR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
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 = Read only; -*n* = value after reset

Table 17-30. Event Missed Register (EMR) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Channel 0-31 event missed. E*n* is cleared by writing a 1 to the corresponding bit in the event missed clear register (EMCR).
		0	No missed event.
		1	Missed event occurred.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.2.2 Event Missed Clear Registers (EMCR)

Once a missed event is posted in the event missed register (EMR), the bit remains set and you need to clear the set bit(s). This is done by way of CPU writes to the event missed clear register (EMCR). Writing a 1 to any of the bits clears the corresponding missed event (bit) in EMR; writing a 0 has no effect.

The EMCR is shown in Figure 17-49 and described in Table 17-31.

Figure 17-49. Event Missed Clear Register (EMCR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

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

Table 17-31. Event Missed Clear Register (EMCR) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Event missed 0-31 clear. All error bits must be cleared before additional error interrupts will be asserted by the EDMA3CC.
		0	No effect.
		1	Corresponding missed event bit in the event missed register (EMR) is cleared (E*n* = 0).

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.2.3 QDMA Event Missed Register (QEMR)

For a particular QDMA channel, if two QDMA events are detected without the first event getting cleared/serviced, the bit corresponding to that channel is set/asserted in the QDMA event missed register (QEMR). The QEMR bits for a channel are also set if a QDMA event on the channel encounters a NULL entry (or a NULL TR is serviced). If any QEMR bit is set (and all errors, including bits in other error registers (EMR or CCERR) were previously cleared), the EDMA3CC generates an error interrupt. See Section 17.2.9.4 for details on EDMA3CC error interrupt generation.

The QEMR is shown in Figure 17-50 and described in Table 17-32.

Figure 17-50. QDMA Event Missed Register (QEMR)

31	16
Reserved	
R-0	

15	8	7	6	5	4	3	2	1	0
Reserved		E7	E6	E5	E4	E3	E2	E1	E0
R-0		R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

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

Table 17-32. QDMA Event Missed Register (QEMR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	E*n*		Channel 0-7 QDMA event missed. E*n* is cleared by writing a 1 to the corresponding bit in the QDMA event missed clear register (QEMCR).
		0	No missed event.
		1	Missed event occurred.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.2.4 QDMA Event Missed Clear Register (QEMCR)

Once a missed event is posted in the QDMA event missed registers (QEMR), the bit remains set and you need to clear the set bit(s). This is done by way of CPU writes to the QDMA event missed clear registers (QEMCR). Writing a 1 to any of the bits clears the corresponding missed event (bit) in QEMR; writing a 0 has no effect.

The QEMCR is shown in Figure 17-51 and described in Table 17-33.

Figure 17-51. QDMA Event Missed Clear Register (QEMCR)

31	16
Reserved	
R-0	

15	8	7	6	5	4	3	2	1	0
Reserved		E7	E6	E5	E4	E3	E2	E1	E0
R-0		W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

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

Table 17-33. QDMA Event Missed Clear Register (QEMCR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	E*n*		QDMA event missed clear. All error bits must be cleared before additional error interrupts will be asserted by the EDMA3CC.
		0	No effect.
		1	Corresponding missed event bit in the QDMA event missed register (QEMR) is cleared (E*n* = 0).

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.2.5 EDMA3CC Error Register (CCERR)

The EDMA3CC error register (CCERR) indicates whether or not at any instant of time the number of events queued up in any of the event queues exceeds or equals the threshold/watermark value that is set in the queue watermark threshold register (QWMTHRA). Additionally, CCERR also indicates if when the number of outstanding TRs that have been programmed to return transfer completion code (TRs that have the TCINTEN or TCCHEN bit in OPT set to 1) to the EDMA3CC has exceeded the maximum allowed value of 31. If any bit in CCERR is set (and all errors, including bits in other error registers (EMR or QEMR) were previously cleared), the EDMA3CC generates an error interrupt. See Section 17.2.9.4 for details on EDMA3CC error interrupt generation. Once the error bits are set in CCERR, they can only be cleared by writing to the corresponding bits in the EDMA3CC error clear register (CCERRCLR).

The CCERR is shown in Figure 17-52 and described in Table 17-34.

Figure 17-52. EDMA3CC Error Register (CCERR)

31	17	16
Reserved		TCCERR
R-0		R-0

15	2	1	0
Reserved		QTHRXCD1	QTHRXCD0
R-0		R-0	R-0

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

Table 17-34. EDMA3CC Error Register (CCERR) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16	TCCERR		Transfer completion code error. TCCERR is cleared by writing a 1 to the corresponding bit in the EDMA3CC error clear register (CCERRCLR).
		0	Total number of allowed TCCs outstanding has not been reached.
		1	Total number of allowed TCCs has been reached.
15-2	Reserved	0	Reserved
1	QTHRXCD1		Queue threshold error for queue 1. QTHRXCD1 is cleared by writing a 1 to the corresponding bit in the EDMA3CC error clear register (CCERRCLR).
		0	Watermark/threshold has not been exceeded.
		1	Watermark/threshold has been exceeded.
0	QTHRXCD0		Queue threshold error for queue 0. QTHRXCD0 is cleared by writing a 1 to the corresponding bit in the EDMA3CC error clear register (CCERRCLR).
		0	Watermark/threshold has not been exceeded.
		1	Watermark/threshold has been exceeded.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.2.6 EDMA3CC Error Clear Register (CCERRCLR)

The EDMA3CC error clear register (CCERRCLR) is used to clear any error bits that are set in the EDMA3CC error register (CCERR). In addition, CCERRCLR also clears the values of some bit fields in the queue status registers (QSTAT*n*) associated with a particular event queue. Writing a 1 to any of the bits clears the corresponding bit in CCERR; writing a 0 has no effect.

The CCERRCLR is shown in Figure 17-53 and described in Table 17-35.

Figure 17-53. EDMA3CC Error Clear Register (CCERRCLR)

31	17	16
Reserved		TCCERR
W-0		W-0

15	2	1	0
Reserved		QTHRXCD1	QTHRXCD0
W-0		W-0	W-0

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

Table 17-35. EDMA3CC Error Clear Register (CCERRCLR) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16	TCCERR		Transfer completion code error clear.
		0	No effect.
		1	Clears the TCCERR bit in the EDMA3CC error register (CCERR).
15-2	Reserved	0	Reserved
1	QTHRXCD1		Queue threshold error clear for queue 1.
		0	No effect.
		1	Clears the QTHRXCD1 bit in the EDMA3CC error register (CCERR) and the WM and THRXCD bits in the queue status register 1 (QSTAT1).
0	QTHRXCD0		Queue threshold error clear for queue 0.
		0	No effect.
		1	Clears the QTHRXCD0 bit in the EDMA3CC error register (CCERR) and the WM and THRXCD bits in the queue status register 0 (QSTAT0).

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.2.7 Error Evaluate Register (EEVAL)

The EDMA3CC error interrupt is asserted whenever an error bit is set in any of the error registers (EMR, QEMR, and CCERR). For subsequent error bits that get set, the EDMA3CC error interrupt is reasserted only when transitioning from an “all the error bits cleared” to “at least one error bit is set”. Alternatively, a CPU write of 1 to the EVAL bit in the error evaluate register (EEVAL) results in reasserting the EDMA3CC error interrupt, if there are any outstanding error bits set due to subsequent error conditions. Writes of 0 have no effect.

The EEVAL is shown in Figure 17-54 and described in Table 17-36.

Figure 17-54. Error Evaluate Register (EEVAL)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		Rsvd	EVAL
R-0		R/W-0	W-0

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

Table 17-36. Error Evaluate Register (EEVAL) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
0	EVAL		Error interrupt evaluate.
		0	No effect.
		1	EDMA3CC error interrupt will be pulsed if any errors have not been cleared in any of the error registers (EMR, QEMR, or CCERR).

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.3 Region Access Enable Registers

The region access enable register group consists of the DMA access enable registers (DRAE*m*) and the QDMA access enable registers (QRAE*m*). Where *m* is the number of shadow regions in the EDMA3CC memory-map for a device. You can configure these registers to assign ownership of DMA/QDMA channels to a particular shadow region.

17.4.2.3.1 DMA Region Access Enable for Region m (DRAEm)

The DMA region access enable registers for shadow region *m* (DRAE*m*) is programmed to allow or disallow read/write accesses on a bit-by-bit bases for all DMA registers in the shadow region *m* view of the DMA channel registers. See the EDMA3CC register memory-map for a list of all the DMA channel and interrupt registers mapped in the shadow region view. Additionally, the DRAE*m* configuration determines completion of which DMA channels will result in assertion of the shadow region *m* DMA completion interrupt (see Section 17.2.9).

The DRAE*m* is shown in Figure 17-55 and described in Table 17-37.

Figure 17-55. DMA Region Access Enable Register for Region *m* (DRAE*m*)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
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	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
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
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	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 17-37. DMA Region Access Enable Register for Region *m* (DRAE*m*) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		DMA region access enable for bit *n*/channel *n* in region *m*.
		0	Accesses via region *m* address space to bit *n* in any DMA channel register are not allowed. Reads return 0 on bit *n* and writes do not modify the state of bit *n*. Enabled interrupt bits for bit *n* do not contribute to the generation of a transfer completion interrupt for shadow region *m*.
		1	Accesses via region *m* address space to bit *n* in any DMA channel register are allowed. Reads return the value from bit *n* and writes modify the state of bit *n*. Enabled interrupt bits for bit *n* contribute to the generation of a transfer completion interrupt for shadow region *m*.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.3.2 QDMA Region Access Enable Registers (QRAE*m*)

The QDMA region access enable registers for shadow region *m* (QRAE*m*) is programmed to allow or disallow read/write accesses on a bit-by-bit bases for all QDMA registers in the shadow region *m* view of the QDMA registers. This includes all 8-bit QDMA registers.

The QRAE*m* is shown in Figure 17-56 and described in Table 17-38.

Figure 17-56. QDMA Region Access Enable for Region *m* (QRAE*m*)

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								E7	E6	E5	E4	E3	E2	E1	E0
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 reset

Table 17-38. QDMA Region Access Enable for Region *m* (QRAE*m*) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	E*n*		QDMA region access enable for bit *n*/QDMA channel *n* in region *m*.
		0	Accesses via region *m* address space to bit *n* in any QDMA channel register are not allowed. Reads return 0 on bit *n* and writes do not modify the state of bit *n*.
		1	Accesses via region *m* address space to bit *n* in any QDMA channel register are allowed. Reads return the value from bit *n* and writes modify the state of bit *n*.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.4 Status/Debug Visibility Registers

The following set of registers provide visibility into the event queues and a TR lifecycle. These are useful for system debug as they provide in-depth visibility for the events queued up in the event queue and also provide information on what parts of the EDMA3CC logic are active once the event has been received by the EDMA3CC.

17.4.2.4.1 Event Queue Entry Registers (QxEy)

The event queue entry registers (Q*x*E*y*) exist for all 16 queue entries (the maximum allowed queue entries) for all event queues (Q0 and Q1) in the EDMA3CC: Q0E0 to Q0E15 and Q1E0 to Q1E15. Each register details the event number (ENUM) and the event type (ETYPE). For example, if the value in Q1E4 is read as 0000 004Fh, this means the 4th entry in queue 1 is a manually-triggered event on DMA channel 15.

The Q*x*E*y* is shown in Figure 17-57 and described in Table 17-39.

Figure 17-57. Event Queue Entry Registers (Q*x*E*y*)

31	16
Reserved	
R-0	

15	8	7	6	5	4	0
Reserved		ETYPE		Rsvd	ENUM	
R-0		R-x		R-0	R-x	

LEGEND: R = Read only; -*n* = value after reset; -x = value is indeterminate after reset

Table 17-39. Event Queue Entry Registers (Q*x*E*y*) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-6	ETYPE	0-3h	Event entry *y* in queue *x*. Specifies the specific event type for the given entry in the event queue.
		0	Event triggered via ER
		1h	Manual triggered via ESR
		2h	Chain triggered via CER
		3h	Autotriggered via QER
5	Reserved	0	Reserved
4-0	ENUM	0-1Fh	Event entry *y* in queue *x*. Event number:
		0-7h	QDMA channel number (0 to 7)
		0-1Fh	DMA channel/event number (0 to 31)

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.4.2 Queue *n* Status Registers (QSTAT*n*)

The queue *n* status register (QSTAT*n*) is shown in Figure 17-58 and described in Table 17-40.

Figure 17-58. Queue *n* Status Register (QSTAT*n*)

31	25	24	23	21	20	16
Reserved		THRXCD	Reserved		WM	
R-0		R-0	R-0		R-0	

15	13	12	8	7	4	3	0
Reserved		NUMVAL		Reserved		STRTPTR	
R-0		R-0		R-0		R-0	

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

Table 17-40. Queue *n* Status Register (QSTAT*n*) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reserved
24	THRXCD		Threshold exceeded. THRXCD is cleared by writing a 1 to the corresponding QTHRXCD*n* bit in the EDMA3CC error clear register (CCERRCLR).
		0	Threshold specified by the Q*n* bit in the queue watermark threshold A register (QWMTHRA) has not been exceeded.
		1	Threshold specified by the Q*n* bit in the queue watermark threshold A register (QWMTHRA) has been exceeded.
23-21	Reserved	0	Reserved
20-16	WM	0-1Fh	Watermark for maximum queue usage. Watermark tracks the most entries that have been in queue *n* since reset or since the last time that the watermark (WM) bit was cleared. WM is cleared by writing a 1 to the corresponding QTHRXCD*n* bit in the EDMA3CC error clear register (CCERRCLR).
		0-10h	Legal values are 0 (empty) to 10h (full).
		11h-1Fh	Reserved
15-13	Reserved	0	Reserved
12-8	NUMVAL	0-1Fh	Number of valid entries in queue *n*. The total number of entries residing in the queue manager FIFO at a given instant. Always enabled.
		0-10h	Legal values are 0 (empty) to 10h (full).
		11h-1Fh	Reserved
7-4	Reserved	0	Reserved
3-0	STRTPTR	0-Fh	Start pointer. The offset to the head entry of queue *n*, in units of entries. Always enabled. Legal values are 0 (0th entry) to Fh (15th entry).

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.4.3 Queue Watermark Threshold A Register (QWMTHRA)

The queue watermark threshold A register (QWMTHRA) is shown in Figure 17-59 and described in Table 17-41.

Figure 17-59. Queue Watermark Threshold A Register (QWMTHRA)

31	16
Reserved	
R-0	

15	13	12	8	7	5	4	0
Reserved		Q1		Reserved		Q0	
R-0		R/W-10h		R-0		R/W-10h	

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

Table 17-41. Queue Watermark Threshold A Register (QWMTHRA) Field Descriptions

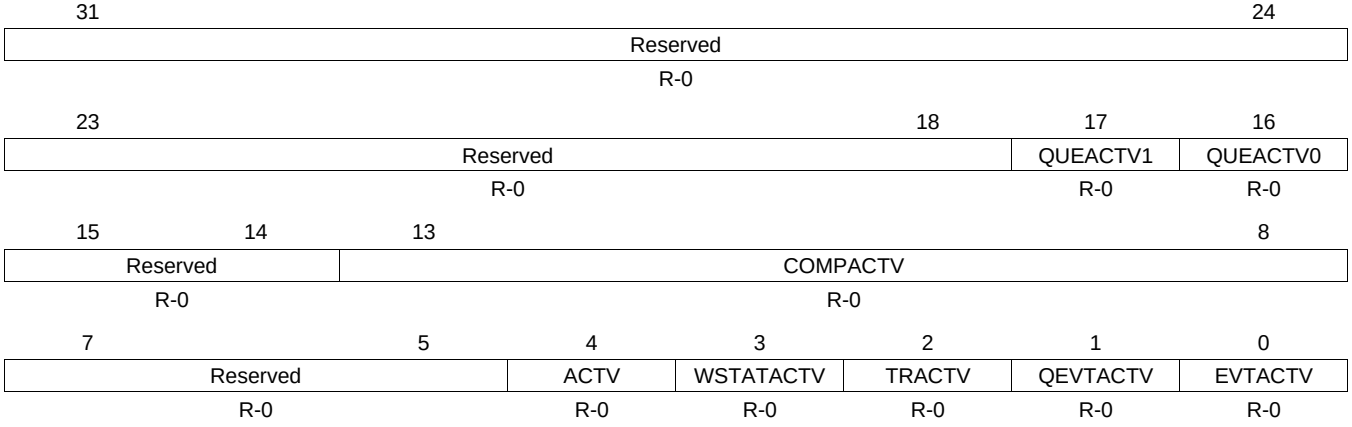
Bit	Field	Value	Description
31-13	Reserved	0	Reserved
12-8	Q1	0-1Fh	Queue threshold for queue 1 value. The QTHRXCD1 bit in the EDMA3CC error register (CCERR) and the THRXCD bit in the queue status register 1 (QSTAT1) are set when the number of events in queue 1 at an instant in time (visible via the NUMVAL bit in QSTAT1) equals or exceeds the value specified by Q1.
		0-10h	The default is 16 (maximum allowed).
		11h	Disables the threshold errors.
		12h-1Fh	Reserved
7-5	Reserved	0	Reserved
4-0	Q0	0-1Fh	Queue threshold for queue 0 value. The QTHRXCD0 bit in the EDMA3CC error register (CCERR) and the THRXCD bit in the queue status register 0 (QSTAT0) are set when the number of events in queue 0 at an instant in time (visible via the NUMVAL bit in QSTAT0) equals or exceeds the value specified by Q0.
		0-10h	The default is 16 (maximum allowed).
		11h	Disables the threshold errors.
		12h-1Fh	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.4.4 EDMA3CC Status Register (CCSTAT)

The EDMA3CC status register (CCSTAT) has a number of status bits that reflect which parts of the EDMA3CC logic is active at any given instant of time. The CCSTAT is shown in Figure 17-60 and described in Table 17-42.

Figure 17-60. EDMA3CC Status Register (CCSTAT)



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

Table 17-42. EDMA3CC Status Register (CCSTAT) Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17	QUEACTV1		Queue 1 active.
		0	No events are queued in queue 1.
		1	At least one TR is queued in queue 1.
16	QUEACTV0		Queue 0 active.
		0	No events are queued in queue 0.
		1	At least one TR is queued in queue 0.
15-14	Reserved	0	Reserved
13-8	COMPACTV	0-3Fh	Completion request active. The COMPACTV field reflects the count for the number of completion requests submitted to the transfer controllers. This count increments every time a TR is submitted and is programmed to report completion (the TCINTEN or TCCCHEN bits in OPT in the parameter entry associated with the TR are set to 1). The counter decrements for every valid TCC received back from the transfer controllers. If at any time the count reaches a value of 63, the EDMA3CC will not service any new TRs until the count is less then 63 (or return a transfer completion code from a transfer controller, which would decrement the count).
		0	No completion requests outstanding.
		1h-3Fh	Total of 1 completion request to 63 completion requests are outstanding.
7-5	Reserved	0	Reserved
4	ACTV		Channel controller active. Channel controller active is a logical-OR of each of the *ACTV bits. The ACTV bit remains high through the life of a TR.
		0	Channel is idle.
		1	Channel is busy.
3	WSTATACTV		Write status interface active.
		0	Write status req is idle and write status fifo is idle.
		1	Either the write status request is active or additional write status responses are pending in the write status fifo.
2	TRACTV		Transfer request active.
		0	Transfer request processing/submission logic is inactive.
		1	Transfer request processing/submission logic is active.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 17-42. EDMA3CC Status Register (CCSTAT) Field Descriptions (continued)

Bit	Field	Value	Description
1	QEVTACTV		QDMA event active.
		0	No enabled QDMA events are active within the EDMA3CC.
		1	At least one enabled QDMA event (QER) is active within the EDMA3CC.
0	EVTACTV		DMA event active.
		0	No enabled DMA events are active within the EDMA3CC.
		1	At least one enabled DMA event (ER and EER, ESR, CER) is active within the EDMA3CC.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.5 DMA Channel Registers

The following registers pertain to the 32 DMA channels. The 32 DMA channels consist of registers (with the exception of DMAQNUM*n*) that each have 32 bits and the bit position of each register matches the DMA channel number.

The DMA channel registers are accessible via read/writes to the global address range. They are also accessible via read/writes to the shadow address range. The read/write ability to the registers in the shadow region is controlled by the DMA region access registers (DRAE*m*). These registers are described in Section 17.4.2.3.1 and the details for shadow region/global region usage is explained in Section 17.2.7.

17.4.2.5.1 Event Register (ER)

All external events are captured in the event register (ER). The events are latched even when the events are not enabled. If the event bit corresponding to the latched event is enabled (EER.E*n* = 1), then the event is evaluated by the EDMA3CC logic for an associated transfer request submission to the transfer controllers. The event register bits are automatically cleared (ER.E*n* = 0) once the corresponding events are prioritized and serviced. If ER.E*n* are already set and another event is received on the same channel/event, then the corresponding event is latched in the event miss register (EMR.E*n*), provided that the event was enabled (EER.E*n* = 1).

Event *n* can be cleared by the CPU writing a 1 to corresponding event bit in the event clear register (ECR). The setting of an event is a higher priority relative to clear operations (via hardware or software). If set and clear conditions occur concurrently, the set condition wins. If the event was previously set, then EMR would be set since an event is lost. If the event was previously clear, then the event remains set and is prioritized for submission to the event queues.

The ER is shown in Figure 17-61 and described in Table 17-43.

Figure 17-61. Event Register (ER)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	19	E18	E17	E16
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
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
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 = Read only; -*n* = value after reset

Table 17-43. Event Register (ER) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Event 0-31. Events 0-31 are captured by the EDMA3CC and are latched into ER. The events are set (E*n* = 1) even when events are disabled (E*n* = 0 in the event enable register, EER).
		0	EDMA3CC event is not asserted.
		1	EDMA3CC event is asserted. Corresponding DMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.5.2 Event Clear Register (ECR)

Once an event has been posted in the event register (ER), the event is cleared in two ways. If the event is enabled in the event enable register (EER) and the EDMA3CC submits a transfer request for the event to the EDMA3TC, it clears the corresponding event bit in the event register. If the event is disabled in the event enable register (EER), the CPU can clear the event by way of the event clear register (ECR).

Writing a 1 to any of the bits clears the corresponding event; writing a 0 has no effect. Once an event bit is set in the event register, it remains set until EDMA3CC submits a transfer request for that event or the CPU clears the event by setting the corresponding bit in ECR.

The ECR is shown in Figure 17-62 and described in Table 17-44.

Figure 17-62. Event Clear Register (ECR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

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

Table 17-44. Event Clear Register (ECR) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Event clear for event 0-31. Any of the event bits in ECR is set to 1 to clear the event (E*n*) in the event register (ER). A write of 0 has no effect.
		0	No effect.
		1	EDMA3CC event is cleared in the event register (ER).

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.5.3 Event Set Register (ESR)

The event set register (ESR) allows the CPU (or EDMA programmers) to manually set events to initiate DMA transfer requests. CPU writes of 1 to any event set register (E*n*) bits set the corresponding bits in the registers. The set event is evaluated by the EDMA3CC logic for an associated transfer request submission to the transfer controllers. Writing a 0 has no effect.

The event set register operates independent of the event register (ER), and a write of 1 is always considered a valid event regardless of whether the event is enabled (the corresponding event bits are set or cleared in EER.E*n*).

Once the event is set in the event set register, it cannot be cleared by CPU writes, in other words, the event clear register (ECR) has no effect on the state of ESR. The bits will only be cleared once the transfer request corresponding to the event has been submitted to the transfer controller. The setting of an event is a higher priority relative to clear operations (via hardware). If set and clear conditions occur concurrently, the set condition wins. If the event was previously set, then EMR would be set since an event is lost. If the event was previously clear, then the event remains set and is prioritized for submission to the event queues.

Manually-triggered transfers via writes to ESR allow the CPU to submit DMA requests in the system, these are relevant for memory-to-memory transfer scenarios. If the ESR.E*n* bit is already set and another CPU write of 1 is attempted to the same bit, then the corresponding event is latched in the event missed registers (EMR.E*n* = 1).

The ESR is shown in Figure 17-63 and described in Table 17-45.

Figure 17-63. Event Set Register (ESR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
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	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
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
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	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 17-45. Event Set Register (ESR) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Event set for event 0-31.
		0	No effect.
		1	Corresponding DMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.5.4 Chained Event Register (CER)

When the OPTIONS parameter for a PaRAM entry is programmed to returned a chained completion code (ITCCHEN = 1 and/or TCCHEN = 1), then the value dictated by the TCC[5:0] (also programmed in OPT) forces the corresponding event bit to be set in the chained event register (CER). The set chained event is evaluated by the EDMA3CC logic for an associated transfer request submission to the transfer controllers. This results in a chained-triggered transfer.

The chained event registers do not have any enables. The generation of a chained event is essentially enabled by the PaRAM entry that has been configured for intermediate and/or final chaining on transfer completion. The E*n* bit is set (regardless of the state of EER.E*n*) when a chained completion code is returned from one of the transfer controllers or is generated by the EDMA3CC via the early completion path. The bits in the chained event register are cleared when the corresponding events are prioritized and serviced.

If the E*n* bit is already set and another chaining completion code is return for the same event, then the corresponding event is latched in the event missed register (EMR.E*n* = 1). The setting of an event is a higher priority relative to clear operations (via hardware). If set and clear conditions occur concurrently, the set condition wins. If the event was previously set, then EMR would be set since an event is lost. If the event was previously clear, then the event remains set and is prioritized for submission to the event queues.

The CER is shown in Figure 17-64 and described in Table 17-46.

Figure 17-64. Chained Event Register (CER)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
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 = Read only; -*n* = value after reset

Table 17-46. Chained Event Register (CER) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Chained event for event 0-31.
		0	No effect.
		1	Corresponding DMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.5.5 Event Enable Register (EER)

The EDMA3CC provides the option of selectively enabling/disabling each event in the event register (ER) by using the event enable register (EER). If an event bit in EER is set to 1 (using the event enable set register, EESR), it will enable that corresponding event. Alternatively, if an event bit in EER is cleared (using the event enable clear register, EECR), it will disable the corresponding event.

The event register latches all events that are captured by EDMA3CC, even if the events are disabled (although EDMA3CC does not process it). Enabling an event with a pending event already set in the event register enables the EDMA3CC to process the already set event like any other new event. The EER settings do not have any effect on chained events (CER.E*n* = 1) and manually set events (ESR.E*n* = 1).

The EER is shown in Figure 17-65 and described in Table 17-47.

Figure 17-65. Event Enable Register (EER)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
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
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
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 = Read only; -*n* = value after reset

Table 17-47. Event Enable Register (EER) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Event enable for events 0-31.
		0	Event is not enabled. An external event latched in the event register (ER) is not evaluated by the EDMA3CC.
		1	Event is enabled. An external event latched in the event register (ER) is evaluated by the EDMA3CC.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.5.6 Event Enable Clear Register (EECR)

The event enable register (EER) cannot be modified by directly writing to it. The intent is to ease the software burden for the case where multiple tasks are attempting to simultaneously modify these registers. The event enable clear register (EECR) is used to disable events. Writes of 1 to the bits in EECR clear the corresponding event bits in EER; writes of 0 have no effect.

The EECR is shown in Figure 17-66 and described in Table 17-48.

Figure 17-66. Event Enable Clear Register (EECR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

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

Table 17-48. Event Enable Clear Register (EECR) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Event enable clear for events 0-31.
		0	No effect.
		1	Event is disabled. Corresponding bit in the event enable register (EER) is cleared (E*n* = 0).

17.4.2.5.7 Event Enable Set Register (EESR)

The event enable register (EER) cannot be modified by directly writing to it. The intent is to ease the software burden for the case where multiple tasks are attempting to simultaneously modify these registers. The event enable set register (EESR) is used to enable events. Writes of 1 to the bits in EESR set the corresponding event bits in EER; writes of 0 have no effect.

The EESR is shown in Figure 17-67 and described in Table 17-49.

Figure 17-67. Event Enable Set Register (EESR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

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

Table 17-49. Event Enable Set Register (EESR) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Event enable set for events 0-31.
		0	No effect.
		1	Event is enabled. Corresponding bit in the event enable register (EER) is set (E*n* = 1).

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.5.8 Secondary Event Register (SER)

The secondary event register (SER) provides information on the state of a DMA channel or event (0 through 31). If the EDMA3CC receives a TR synchronization due to a manual-trigger, event-trigger, or chained-trigger source (ESR.E*n* = 1, ER.E*n* = 1, or CER.E*n* = 1), which results in the setting of a corresponding event bit in SER (SER.E*n* = 1), it implies that the corresponding DMA event is in the queue.

Once a bit corresponding to an event is set in SER, the EDMA3CC does not prioritize additional events on the same DMA channel. Depending on the condition that leads to the setting of the SER bits, either the EDMA3CC hardware or the software (using SECR) needs to clear the SER bits for the EDMA3CC to evaluate subsequent events and perform subsequent transfers on the same channel. Based on whether the associated TR is valid, or it is a null or dummy TR, the implications on the state of SER and the required user action in order to submit another DMA transfer might be different.

The SER is shown in Figure 17-68 and described in Table 17-50.

Figure 17-68. Secondary Event Register (SER)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
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
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
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 = Read only; -*n* = value after reset

Table 17-50. Secondary Event Register (SER) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Secondary event register. The secondary event register is used to provide information on the state of an event.
		0	Event is not currently stored in the event queue.
		1	Event is currently stored in the event queue. Event arbiter will not prioritize additional events.

17.4.2.5.9 Secondary Event Clear Register (SECR)

The secondary event clear register (SECR) clears the status of the secondary event registers (SER). CPU writes of 1 clear the corresponding set bits in SER. Writes of 0 have no effect.

The SECR is shown in Figure 17-69 and described in Table 17-51.

Figure 17-69. Secondary Event Clear Register (SECR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E31	E30	E29	E28	E27	E26	E25	E24	E23	E22	E21	E20	E19	E18	E17	E16
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E15	E14	E13	E12	E11	E10	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

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

Table 17-51. Secondary Event Clear Register (SECR) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Secondary event clear register
		0	No effect.
		1	Corresponding bit in the secondary event register (SER) is cleared (E*n* = 0).

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.6 Interrupt Registers

All DMA/QDMA channels can be set to assert an EDMA3CC completion interrupt to the CPU on transfer completion, by appropriately configuring the PaRAM entry associated with the channels. The following registers are used for the transfer completion interrupt reporting/generating by the EDMA3CC. See Section 17.2.9 for more details on EDMA3CC completion interrupt generation.

17.4.2.6.1 Interrupt Enable Registers (IER)

Interrupt enable register (IER) is used to enable/disable the transfer completion interrupt generation by the EDMA3CC for all DMA/QDMA channels. The IER cannot be written to directly. To set any interrupt bit in IER, a 1 must be written to the corresponding interrupt bit in the interrupt enable set registers (IESR). Similarly, to clear any interrupt bit in IER, a 1 must be written to the corresponding interrupt bit in the interrupt enable clear register (IECR).

The IER is shown in Figure 17-70 and described in Table 17-52.

Figure 17-70. Interrupt Enable Register (IER)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
I31	I30	I29	I28	I27I	I26	I25	I24	I23	I22	I21	I20	I19	I18	I17	I16
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
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0
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 = Read only; -*n* = value after reset

Table 17-52. Interrupt Enable Register (IER) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Interrupt enable for channels 0-31.
		0	Interrupt is not enabled.
		1	Interrupt is enabled.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.6.2 Interrupt Enable Clear Register (IECR)

The interrupt enable clear register (IECR) is used to clear interrupts. Writes of 1 to the bits in IECR clear the corresponding interrupt bits in the interrupt enable registers (IER); writes of 0 have no effect.

The IECR is shown in Figure 17-71 and described in Table 17-53.

Figure 17-71. Interrupt Enable Clear Register (IECR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
I31	I30	I29	I28	I27	I26	I25	I24	I23	I22	I21	I20	I19	I18	I17	16
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

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

Table 17-53. Interrupt Enable Clear Register (IECR) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Interrupt enable clear for channels 0-31.
		0	No effect
		1	Corresponding bit in the interrupt enable register (IER) is cleared (I*n* = 0).

17.4.2.6.3 Interrupt Enable Set Register (IESR)

The interrupt enable set register (IESR) is used to enable interrupts. Writes of 1 to the bits in IESR set the corresponding interrupt bits in the interrupt enable registers (IER); writes of 0 have no effect.

The IESR is shown in Figure 17-72 and described in Table 17-54.

Figure 17-72. Interrupt Enable Set Register (IESR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
I31	I30	I29	I28	I27	I26	I25	I24	I23	I22	I21	I20	I19	II8	I17	I16
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

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

Table 17-54. Interrupt Enable Set Register (IESR) Field Descriptions

Bit	Field	Value	Description
31-0	E*n*		Interrupt enable set for channels 0-31.
		0	No effect.
		1	Corresponding bit in the interrupt enable register (IER) is set (I*n* = 1).

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.6.4 Interrupt Pending Register (IPR)

If the TCINTEN and/or ITCINTEN bit in the channel option parameter (OPT) is set to 1 in the PaRAM entry associated with the channel (DMA or QDMA), then the EDMA3TC (for normal completion) or the EDMA3CC (for early completion) returns a completion code on transfer or intermediate transfer completion. The value of the returned completion code is equal to the TCC bit in OPT for the PaRAM entry associated with the channel.

When an interrupt transfer completion code with TCC = *n* is detected by the EDMA3CC, then the corresponding bit is set in the interrupt pending register (IPR.I*n*, if *n* = 0 to 31). Note that once a bit is set in the interrupt pending registers, it remains set; it is your responsibility to clear these bits. The bits set in IPR are cleared by writing a 1 to the corresponding bits in the interrupt clear registers (ICR).

The IPR is shown in Figure 17-73 and described in Table 17-55.

Figure 17-73. Interrupt Pending Register (IPR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
I31	I30	I29	I28	I27	I26	I25	I24	I23	I22	I21	I20	I19	I18	I17	I16
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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0
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 = Read only; -*n* = value after reset

Table 17-55. Interrupt Pending Register (IPR) Field Descriptions

Bit	Field	Value	Description
31-0	I*n*		Interrupt pending for TCC = 0-31.
		0	Interrupt transfer completion code is not detected or was cleared.
		1	Interrupt transfer completion code is detected (I*n* = 1, *n* = EDMA3TC[5:0]).

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.6.5 Interrupt Clear Register (ICR)

The bits in the interrupt pending register (IPR) are cleared by writing a 1 to the corresponding bits in the interrupt clear register (ICR); writes of 0 have no effect. All set bits in IPR must be cleared to allow EDMA3CC to assert additional transfer completion interrupts.

The ICR is shown in Figure 17-74 and described in Table 17-56.

Figure 17-74. Interrupt Clear Register (ICR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
I31	I30	I29	I28	I27	I26	I25	I24	I23	I22	I21	I20	I19	I18	I17	I16
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I15	I14	I13	I12	I11	I10	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

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

Table 17-56. Interrupt Clear Register (ICR) Field Descriptions

Bit	Field	Value	Description
31-0	I*n*		Interrupt clear register for TCC = 0-31.
		0	No effect.
		1	Corresponding bit in the interrupt pending register (IPR) is cleared (I*n* = 0).

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.6.6 Interrupt Evaluate Register (IEVAL)

The interrupt evaluate register (IEVAL) is the only register that physically exists in both the global region and the shadow regions. In other words, the read/write accessibility for the shadow region IEVAL is not affected by the DMA/QDMA region access registers (DRAE*m* and QRAE*m*). IEVAL is needed for robust ISR operations to ensure that interrupts are not missed by the CPU.

The IEVAL is shown in Figure 17-75 and described in Table 17-57.

Figure 17-75. Interrupt Evaluate Register (IEVAL)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		Rsvd	EVAL
R-0		W-0	W-0

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

Table 17-57. Interrupt Evaluate Register (IEVAL) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
0	EVAL		Interrupt evaluate.
		0	No effect.
		1	Causes EDMA3CC completion interrupt to be pulsed, if any enabled (IER*n* = 1) interrupts are still pending (IPR*n* = 1). The EDMA3CC completion region interrupt that is pulsed depends on which IEVAL is being exercised. For example, writing to the EVAL bit in IEVAL0 pulses the region 0 completion interrupt, but writing to the EVAL bit in IEVAL1 pulses the region 1 completion interrupt.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.7 QDMA Channel Registers

The following registers pertain to the 8 QDMA channels. The 8 QDMA channels consist of registers (with the exception of QDMAQNUM) that each have 8 bits and the bit position of each register matches the QDMA channel number.

The QDMA channel registers are accessible via read/writes to the global address range. They are also accessible via read/writes to the shadow address range. The read/write ability to the registers in the shadow region is controlled by the QDMA region access registers (QRAE*m*). These registers are described in Section 17.4.2.3.2 and the details for shadow region/global region usage is explained in Section 17.2.7.

17.4.2.7.1 QDMA Event Register (QER)

The QDMA event register (QER) channel *n* bit is set (E*n* = 1) when the CPU or any EDMA programmer (including EDMA3) performs a write to the trigger word (using the QDMA channel *n* mapping register (QCHMAP*n*)) in the PaRAM entry associated with QDMA channel *n* (which is also programmed using QCHMAP*n*). The E*n* bit is also set when the EDMA3CC performs a link update on a PaRAM address that matches the QCHMAP*n* settings. The QDMA event is latched only if the QDMA event enable register (QEER) channel *n* bit is also enabled (QEER.E*n* = 1). Once a bit is set in QER, then the corresponding QDMA event (auto-trigger) is evaluated by the EDMA3CC logic for an associated transfer request submission to the transfer controllers.

The setting of an event is a higher priority relative to clear operations (via hardware). If set and clear conditions occur concurrently, the set condition wins. If the event was previously set, then the QDMA event missed register (QEMR) would be set because an event is lost. If the event was previously clear, then the event remains set and is prioritized for submission to the event queues.

The set bits in QER are only cleared when the transfer request associated with the corresponding channels has been processed by the EDMA3CC and submitted to the transfer controller. If the E*n* bit is already set and a QDMA event for the same QDMA channel occurs prior to the original being cleared, then the second missed event is latched in QEMR (E*n* =1).

The QER is shown in Figure 17-76 and described in Table 17-58.

Figure 17-76. QDMA Event Register (QER)

31	16
Reserved	
R-0	

15	8	7	6	5	4	3	2	1	0
Reserved		E7	E6	E5	E4	E3	E2	E1	E0
R-0		R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

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

Table 17-58. QDMA Event Register (QER) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	E*n*		QDMA event for channels 0-7.
		0	No effect.
		1	Corresponding QDMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.7.2 QDMA Event Enable Register (QEER)

The EDMA3CC provides the option of selectively enabling/disabling each channel in the QDMA event register (QER) by using the QDMA event enable register (QEER). If any of the event bits in QEER is set to 1 (using the QDMA event enable set register, QEESR), it will enable that corresponding event. Alternatively, if any event bit in QEER is cleared (using the QDMA event enable clear register, QEECR), it will disable the corresponding QDMA channel. The QDMA event register will not latch any event for a QDMA channel, if it is not enabled via QEER.

The QEER is shown in Figure 17-77 and described in Table 17-59.

Figure 17-77. QDMA Event Enable Register (QEER)

31	16
Reserved	
R-0	

15	8	7	6	5	4	3	2	1	0
Reserved		E7	E6	E5	E4	E3	E2	E1	E0
R-0		R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

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

Table 17-59. QDMA Event Enable Register (QEER) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	E*n*		QDMA event enable for channels 0-7.
		0	QDMA channel *n* is not enabled. QDMA event will not be recognized and will not latch in the QDMA event register (QER).
		1	QDMA channel *n* is enabled. QDMA events will be recognized and will get latched in the QDMA event register (QER).

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.7.3 QDMA Event Enable Clear Register (QEECR)

The QDMA event enable register (QEER) cannot be modified by directly writing to the register, in order to ease the software burden when multiple tasks are attempting to simultaneously modify these registers. The QDMA event enable clear register (QEECR) is used to disable events. Writes of 1 to the bits in QEECR clear the corresponding QDMA channel bits in QEER; writes of 0 have no effect.

The QEECR is shown in Figure 17-78 and described in Table 17-60.

Figure 17-78. QDMA Event Enable Clear Register (QEECR)

31	16
Reserved	
R-0	

15	8	7	6	5	4	3	2	1	0
Reserved		E7	E6	E5	E4	E3	E2	E1	E0
R-0		W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

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

Table 17-60. QDMA Event Enable Clear Register (QEECR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	E*n*		QDMA event enable clear for channels 0-7.
		0	No effect.
		1	QDMA event is disabled. Corresponding bit in the QDMA event enable register (QEER) is cleared (E*n* = 0).

17.4.2.7.4 QDMA Event Enable Set Register (QEESR)

The QDMA event enable register (QEER) cannot be modified by directly writing to the register, in order to ease the software burden when multiple tasks are attempting to simultaneously modify these registers. The QDMA event enable set register (QEESR) is used to enable events. Writes of 1 to the bits in QEESR set the corresponding QDMA channel bits in QEER; writes of 0 have no effect.

The QEESR is shown in Figure 17-79 and described in Table 17-61.

Figure 17-79. QDMA Event Enable Set Register (QEESR)

31	16
Reserved	
R-0	

15	8	7	6	5	4	3	2	1	0
Reserved		E7	E6	E5	E4	E3	E2	E1	E0
R-0		W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

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

Table 17-61. QDMA Event Enable Set Register (QEESR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	E*n*		QDMA event enable set for channels 0-7.
		0	No effect.
		1	QDMA event is enabled. Corresponding bit in the QDMA event enable register (QEER) is set (E*n* = 1).

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.7.5 QDMA Secondary Event Register (QSER)

The QDMA secondary event register (QSER) provides information on the state of a QDMA event. If at any time a bit corresponding to a QDMA channel is set in QSER, that implies that the corresponding QDMA event is in the queue. Once a bit corresponding to a QDMA channel is set in QSER, the EDMA3CC does not prioritize additional events on the same QDMA channel. Depending on the condition that lead to the setting of the QSER bits, either the EDMA3CC hardware or the software (using QSECR) needs to clear the QSER bits for the EDMA3CC to evaluate subsequent QDMA events on the channel. Based on whether the associated TR is valid, or it is a null or dummy TR, the implications on the state of QSER and the required user action in order to submit another QDMA transfer might be different.

The QSER is shown in Figure 17-80 and described in Table 17-62.

Figure 17-80. QDMA Secondary Event Register (QSER)

31	16
Reserved	
R-0	

15	8	7	6	5	4	3	2	1	0
Reserved		E7	E6	E5	E4	E3	E2	E1	E0
R-0		R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

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

Table 17-62. QDMA Secondary Event Register (QSER) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	E*n*		QDMA secondary event register for channels 0-7.
		0	QDMA event is not currently stored in the event queue.
		1	QDMA event is currently stored in event queue. EDMA3CC will not prioritize additional events.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.2.7.6 QDMA Secondary Event Clear Register (QSECR)

The QDMA secondary event clear register (QSECR) clears the status of the QDMA secondary event register (QSER) and the QDMA event register (QER). CPU writes of 1 clear the corresponding set bits in QSER and QER. Writes of 0 have no effect. Note that this differs from the secondary event clear register (SECR) operation, which only clears the secondary event register (SER) bits and does not affect the event registers.

The QSECR is shown in Figure 17-81 and described in Table 17-63.

Figure 17-81. QDMA Secondary Event Clear Register (QSECR)

31	16
Reserved	
R-0	

15	8	7	6	5	4	3	2	1	0
Reserved		E7	E6	E5	E4	E3	E2	E1	E0
R-0		W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

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

Table 17-63. QDMA Secondary Event Clear Register (QSECR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	E*n*		QDMA secondary event clear register for channels 0-7.
		0	No effect.
		1	Corresponding bit in the QDMA secondary event register (QSER) and the QDMA event register (QER) is cleared (E*n* = 0).

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.3 EDMA3 Transfer Controller (EDMA3TC) Registers

Table 17-64 lists the memory-mapped registers for the EDMA3 transfer controller (EDMA3TC). See your device-specific data manual for the memory address of these registers. All other register offset addresses not listed in Table 17-64 should be considered as reserved locations and the register contents should not be modified.

Table 17-64. EDMA3 Transfer Controller (EDMA3TC) Registers

Offset	Acronym	Register Description	Section
0h	REVID	Revision Identification Register	Section 17.4.3.1
4h	TCCFG	EDMA3TC Configuration Register	Section 17.4.3.2
100h	TCSTAT	EDMA3TC Channel Status Register	Section 17.4.3.3
120h	ERRSTAT	Error Status Register	Section 17.4.3.4.1
124h	ERREN	Error Enable Register	Section 17.4.3.4.2
128h	ERRCLR	Error Clear Register	Section 17.4.3.4.3
12Ch	ERRDET	Error Details Register	Section 17.4.3.4.4
130h	ERRCMD	Error Interrupt Command Register	Section 17.4.3.4.5
140h	RDRATE	Read Command Rate Register	Section 17.4.3.5
240h	SAOPT	Source Active Options Register	Section 17.4.3.6.1
244h	SASRC	Source Active Source Address Register	Section 17.4.3.6.2
248h	SACNT	Source Active Count Register	Section 17.4.3.6.3
24Ch	SADST	Source Active Destination Address Register	Section 17.4.3.6.4
250h	SABIDX	Source Active B-Index Register	Section 17.4.3.6.5
254h	SAMPPRXY	Source Active Memory Protection Proxy Register	Section 17.4.3.6.6
258h	SACNTRLD	Source Active Count Reload Register	Section 17.4.3.6.7
25Ch	SASRCBREF	Source Active Source Address B-Reference Register	Section 17.4.3.6.8
260h	SADSTBREF	Source Active Destination Address B-Reference Register	Section 17.4.3.6.9
280h	DFCNTRLD	Destination FIFO Set Count Reload Register	Section 17.4.3.6.10
284h	DFSRCBREF	Destination FIFO Set Source Address B-Reference Register	Section 17.4.3.6.11
288h	DFDSTBREF	Destination FIFO Set Destination Address B-Reference Register	Section 17.4.3.6.12
300h	DFOPT0	Destination FIFO Options Register 0	Section 17.4.3.6.13
304h	DFSRC0	Destination FIFO Source Address Register 0	Section 17.4.3.6.14
308h	DFCNT0	Destination FIFO Count Register 0	Section 17.4.3.6.15
30Ch	DFDST0	Destination FIFO Destination Address Register 0	Section 17.4.3.6.16
310h	DFBIDX0	Destination FIFO B-Index Register 0	Section 17.4.3.6.17
314h	DFMPPRXY0	Destination FIFO Memory Protection Proxy Register 0	Section 17.4.3.6.18
340h	DFOPT1	Destination FIFO Options Register 1	Section 17.4.3.6.13
344h	DFSRC1	Destination FIFO Source Address Register 1	Section 17.4.3.6.14
348h	DFCNT1	Destination FIFO Count Register 1	Section 17.4.3.6.15
34Ch	DFDST1	Destination FIFO Destination Address Register 1	Section 17.4.3.6.16
350h	DFBIDX1	Destination FIFO B-Index Register 1	Section 17.4.3.6.17
354h	DFMPPRXY1	Destination FIFO Memory Protection Proxy Register 1	Section 17.4.3.6.18
380h	DFOPT2	Destination FIFO Options Register 2	Section 17.4.3.6.13
384h	DFSRC2	Destination FIFO Source Address Register 2	Section 17.4.3.6.14
388h	DFCNT2	Destination FIFO Count Register 2	Section 17.4.3.6.15
38Ch	DFDST2	Destination FIFO Destination Address Register 2	Section 17.4.3.6.16
390h	DFBIDX2	Destination FIFO B-Index Register 2	Section 17.4.3.6.17
394h	DFMPPRXY2	Destination FIFO Memory Protection Proxy Register 2	Section 17.4.3.6.18
3C0h	DFOPT3	Destination FIFO Options Register 3	Section 17.4.3.6.13
3C4h	DFSRC3	Destination FIFO Source Address Register 3	Section 17.4.3.6.14
3C8h	DFCNT3	Destination FIFO Count Register 3	Section 17.4.3.6.15

Copyright © 2013–2016, Texas Instruments Incorporated

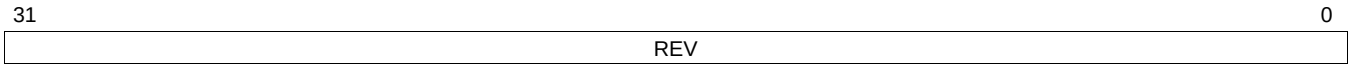
Table 17-64. EDMA3 Transfer Controller (EDMA3TC) Registers (continued)

Offset	Acronym	Register Description	Section
3CCh	DFDST3	Destination FIFO Destination Address Register 3	Section 17.4.3.6.16
3D0h	DFBIDX3	Destination FIFO B-Index Register 3	Section 17.4.3.6.17
3D4h	DFMPPRXY3	Destination FIFO Memory Protection Proxy Register 3	Section 17.4.3.6.18

17.4.3.1 Revision Identification Register (REVID)

The revision identification register (REVID) is a constant register that uniquely identifies the EDMA3TC and specific revision of the EDMA3TC. The REVID is shown in Figure 17-82 and described in Table 17-65.

Figure 17-82. Revision ID Register (REVID)



R-4000 3B00h

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

Table 17-65. Revision ID Register (REVID) Field Descriptions

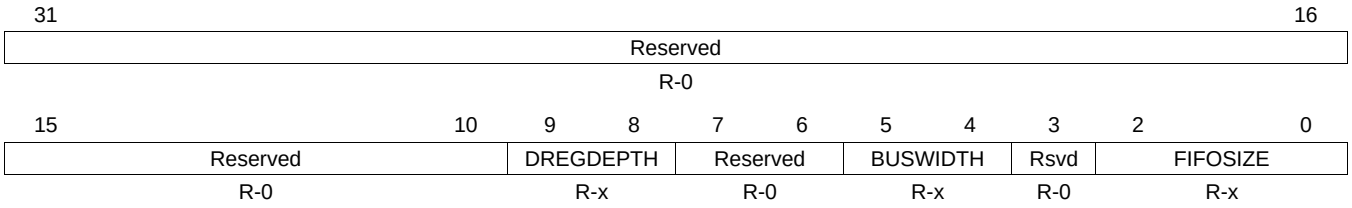
Bit	Field	Value	Description
31-0	REV		Peripheral identifier.
		4000 3B00h	Uniquely identifies the EDMA3TC and the specific revision of the EDMA3TC.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.3.2 EDMA3TC Configuration Register (TCCFG)

The EDMA3TC configuration register (TCCFG) is shown in Figure 17-83 and described in Table 17-66.

Figure 17-83. EDMA3TC Configuration Register (TCCFG)



LEGEND: R = Read only; -*n* = value after reset; -x = value is indeterminate after reset

Table 17-66. EDMA3TC Configuration Register (TCCFG) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9-8	DREGDEPTH	0-3h	Destination register FIFO depth parameterization.
		0	1 entry
		1h	2 entry
		2h	4 entry (for EDMA3TC0 and EDMA3TC1)
		3h	Reserved
7-6	Reserved	0	Reserved
5-4	BUSWIDTH	0-3h	Bus width parameterization.
		0	32-bit
		1h	64-bit (for EDMA3TC0 and EDMA3TC1)
		2h-3h	Reserved
3	Reserved	0	Reserved
2-0	FIFOSIZE	0-7h	FIFO size.
		0	32-byte FIFO
		1h	64-byte FIFO
		2h	128-byte FIFO (for EDMA3TC0 and EDMA3TC1)
		3h	256-byte FIFO
		4h-7h	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.3.3 EDMA3TC Channel Status Register (TCSTAT)

The EDMA3TC channel status register (TCSTAT) is shown in Figure 17-84 and described in Table 17-67.

Figure 17-84. EDMA3TC Channel Status Register (TCSTAT)

31	16
Reserved	
R-0	

15	13	12	11	10	8
Reserved		DFSTRTPTR		Reserved	
R-0		R-0		R-0	

7	6	4	3	2	1	0
Reserved	DSTACTV		Reserved	WSACTV	SRCACTV	PROGBUSY
R-0	R-0		R-0	R-0	R-0	R-0

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

Table 17-67. EDMA3TC Channel Status Register (TCSTAT) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reserved
12-11	DFSTRTPTR	0-3h	Destination FIFO start pointer. The offset to the head entry of the destination register FIFO, in units of *entries*.
10-7	Reserved	0	Reserved
6-4	DSTACTV	0-7h	Destination active state. Specifies the number of transfer requests (TRs) that are resident in the destination register FIFO at a given instant. This bit field can be primarily used for advanced debugging.
		0	Destination FIFO is empty.
		1h	Destination FIFO contains 1 TR.
		2h	Destination FIFO contains 2 TR.
		3h	Destination FIFO contains 3 TR.
		4h	Destination FIFO contains 4 TR. (Full if DSTREGDEPTH == 4)
			If the destination register FIFO is empty, then any TR written to Prog Set immediately transitions to the destination register FIFO. If the destination register FIFO is not empty and not full, then any TR written to Prog Set immediately transitions to the destination register FIFO set if the source active state (SRCACTV) bit is set to idle.
			If the destination register FIFO is full, then TRs cannot transition to the destination register FIFO. The destination register FIFO becomes not full when the TR at the head of the destination register FIFO is completed.
		5h-7h	Reserved
3	Reserved	0	Reserved
2	WSACTV		Write status active.
		0	Write status is not pending. Write status has been received for all previously issued write commands.
		1	Write status is pending. Write status has not been received for all previously issued write commands.
1	SRCACTV		Source active state.
		0	Source active set is idle and is available for programming by the EDMA3CC. Source active register set contains a previously processed transfer request.
		1	Source active set is busy servicing a transfer request.
0	PROGBUSY		Program register set busy.
		0	Program set idle and is available for programming by the EDMA3CC.
		1	Program set busy.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.3.4 Error Registers

17.4.3.4.1 Error Status Register (ERRSTAT)

The error status register (ERRSTAT) is shown in Figure 17-85 and described in Table 17-68.

Figure 17-85. Error Status Register (ERRSTAT)

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		MMRAERR	TRERR	Reserved	BUSERR
R-0		R-0	R-0	R-0	R-0

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

Table 17-68. Error Status Register (ERRSTAT) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	MMRAERR		MMR address error.
		0	MMR address error is not detected.
		1	User attempted to read or write to an invalid address in configuration memory map.
2	TRERR		Transfer request (TR) error event.
		0	Transfer request (TR) error is not detected.
		1	Transfer request (TR) detected that violates constant addressing mode transfer (SAM or DAM is set to 1) alignment rules or has ACNT or BCNT == 0.
1	Reserved	0	Reserved
0	BUSERR		Bus error event.
		0	Bus error is not detected.
		1	EDMA3TC has detected an error at source or destination address. Error information can be read from the error details register (ERRDET).

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.3.4.2 Error Enable Register (ERREN)

The error enable register (ERREN) is shown in Figure 17-86 and described in Table 17-69. When any of the enable bits in ERREN is set, a bit set in the corresponding error status register (ERRSTAT) causes an assertion of the EDMA3TC interrupt.

Figure 17-86. Error Enable Register (ERREN)

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		MMRAERR	TRERR	Reserved	BUSERR
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 17-69. Error Enable Register (ERREN) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	MMRAERR		Interrupt enable for MMR address error (MMRAERR).
		0	MMRAERR is disabled.
		1	MMRAERR is enabled and contributes to the state of EDMA3TC error interrupt generation
2	TRERR		Interrupt enable for transfer request error (TRERR).
		0	TRERR is disabled.
		1	TRERR is enabled and contributes to the state of EDMA3TC error interrupt generation.
1	Reserved	0	Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.
0	BUSERR		Interrupt enable for bus error (BUSERR).
		0	BUSERR is disabled.
		1	BUSERR is enabled and contributes to the state of EDMA3TC error interrupt generation.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.3.4.3 Error Clear Register (ERRCLR)

The error clear register (ERRCLR) is shown in Figure 17-87 and described in Table 17-70.

Figure 17-87. Error Clear Register (ERRCLR)

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		MMRAERR	TRERR	Reserved	BUSERR
R-0		W-0	W-0	R-0	W-0

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

Table 17-70. Error Clear Register (ERRCLR) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	MMRAERR		Interrupt enable clear for the MMR address error (MMRAERR) bit in the error status register (ERRSTAT).
		0	No effect.
		1	Clears the MMRAERR bit in the error status register (ERRSTAT) but does not clear the error details register (ERRDET).
2	TRERR		Interrupt enable clear for the transfer request error (TRERR) bit in the error status register (ERRSTAT).
		0	No effect.
		1	Clears the TRERR bit in the error status register (ERRSTAT) but does not clear the error details register (ERRDET).
1	Reserved	0	Reserved
0	BUSERR		Interrupt clear for the bus error (BUSERR) bit in the error status register (ERRSTAT).
		0	No effect.
		1	Clears the BUSERR bit in the error status register (ERRSTAT) and clears the error details register (ERRDET).

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.3.4.4 Error Details Register (ERRDET)

The error details register (ERRDET) is shown in Figure 17-88 and described in Table 17-71.

Figure 17-88. Error Details Register (ERRDET)

31	18	17	16
Reserved		TCCHEN	TCINTEN
R-0		R-0	R-0

15	14	13	8	7	4	3	0
Reserved		TCC		Reserved		STAT	
R-0		R-0		R-0		R-0	

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

Table 17-71. Error Details Register (ERRDET) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
17	TCCHEN	0-1	Transfer completion chaining enable. Contains the TCCHEN value in the channel options parameter (OPT) programmed by the channel controller for the read or write transaction that resulted in an error.
16	TCINTEN	0-1	Transfer completion interrupt enable. Contains the TCINTEN value in the channel options parameter (OPT) programmed by the channel controller for the read or write transaction that resulted in an error.
15-14	Reserved	0	Reserved
13 - 8	TCC	0-3Fh	Transfer complete code. Contains the TCC value in the channel options parameter (OPT) programmed by the channel controller for the read or write transaction that resulted in an error.
7-4	Reserved	0	Reserved
3-0	STAT	0-Fh	Transaction status. Stores the nonzero status/error code that was detected on the read status or write status bus. If read status and write status are returned on the same cycle, then the EDMA3TC chooses nonzero version. If both are nonzero, then the write status is treated as higher priority.
		0	No error
		1h	Read addressing error
		2h	Read privilege error
		3h	Read timeout error
		4h	Read data error
		5h-6h	Reserved
		7h	Read exclusive operation error
		8h	Reserved
		9h	Write addressing error
		Ah	Write privilege error
		Bh	Write timeout error
		Ch	Write data error
		Dh-Eh	Reserved
		Fh	Write exclusive operation error

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.3.4.5 Error Interrupt Command Register (ERRCMD)

The error interrupt command register (ERRCMD) is shown in Figure 17-89 and described in Table 17-72.

Figure 17-89. Error Interrupt Command Register (ERRCMD)

31	16
Reserved	
R-0	

15	1	0
Reserved		EVAL
R-0		W-0

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

Table 17-72. Error Interrupt Command Register (ERRCMD) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	EVAL		Error evaluate.
		0	No effect.
		1	EDMA3TC error line is pulsed if any of the error status register (ERRSTAT) bits are set to 1.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.3.5 Read Command Rate Register (RDRATE)

The EDMA3 transfer controller issues Read commands at a rate controlled by the Read command rate register (RDRATE). The RDRATE defines the number of idle cycles that the Read controller must wait before issuing subsequent commands. This applies both to commands within a transfer request packet (TRP) and for commands that are issued for different transfer requests (TRs). For instance, if RDRATE is set to 4 cycles between reads, there are 32 inactive cycles between reads.

RDRATE allows flexibility in transfer controller access requests to an endpoint. For an application, RDRATE can be manipulated to slow down the access rate, so that the endpoint may service requests from other masters during the inactive EDMA3TC cycles.

The RDRATE is shown in Figure 17-90 and described in Table 17-73.

NOTE: It is expected that the RDRATE value for a transfer controller is static, as it is decided based on the application requirement. It is not recommended to change this setting on the go.

Figure 17-90. Read Command Rate Register (RDRATE)

31	16
Reserved	
R-0	

15	3	2	0
Reserved		RDRATE	
R-0		R/W-0	

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

Table 17-73. Read Command Rate Register (RDRATE) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	RDRATE	0-7h	Read rate. Controls the number of cycles between Read commands. This is a global setting that applies to all TRs for this EDMA3TC.
		0	Reads issued as fast as possible.
		1h	4 EDMA3TC cycles between reads.
		2h	8 EDMA3TC cycles between reads.
		3h	16 EDMA3TC cycles between reads.
		4h	32 EDMA3TC cycles between reads.
		5h-7h	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.3.6 EDMA3TC Channel Registers

The EDMA3TC channel registers are split into three parts: the programming registers, the source active registers, and the destination FIFO registers. This section describes the registers and their functions. The program register set is programmed by the channel controller and is for internal use. The source active registers and the destination FIFO registers are read-only and are provided to facilitate advanced debug capabilities. The number of destination FIFO register sets depends on the destination FIFO depth. Both TC0 and TC1 have a destination FIFO depth of 4, and there are four sets of destination FIFO registers.

17.4.3.6.1 Source Active Options Register (SAOPT)

The source active options register (SAOPT) is shown in Figure 17-91 and described in Table 17-74.

Figure 17-91. Source Active Options Register (SAOPT)

31	23	22	21	20	19	18	17	16
Reserved		TCCHEN	Rsvd	TCINTEN	Reserved		TCC	
R-0		R/W-0	R-0	R/W-0	R-0		R/W-0	

15	12	11	10	8	7	6	4	3	2	1	0
TCC		Rsvd	FWID		Rsvd	PRI[1]		Reserved		DAM	SAM
R/W-0		R-0	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

(1) On previous architectures, the EDMA3TC priority was controlled by the queue priority register (QUEPRI) in the EDMA3CC memory-map. However for this device, the priority control for the transfer controllers is controlled by the chip-level registers in the System Configuration Module. You should use the chip-level registers and not QUEPRI to configure the TC priority.

Table 17-74. Source Active Options Register (SAOPT) Field Descriptions

Bit	Field	Value	Description
31-23	Reserved	0	Reserved
22	TCCHEN		Transfer complete chaining enable.
		0	Transfer complete chaining is disabled.
		1	Transfer complete chaining is enabled.
21	Reserved	0	Reserved
20	TCINTEN		Transfer complete interrupt enable.
		0	Transfer complete interrupt is disabled.
		1	Transfer complete interrupt is enabled.
19-18	Reserved	0	Reserved
17-12	TCC	0-3Fh	Transfer complete code. This 6-bit code is used to set the relevant bit in CER or IPR of the EDMA3CC.
11	Reserved	0	Reserved
10-8	FWID	0-7h	FIFO width. Applies if either SAM or DAM is set to constant addressing mode.
		0	FIFO width is 8 bits.
		1h	FIFO width is 16 bits.
		2h	FIFO width is 32 bits.
		3h	FIFO width is 64 bits.
		4h	FIFO width is 128 bits.
		5h-7h	Reserved
7	Reserved	0	Reserved
6-4	PRI	0-7h	Transfer priority. Reflects the values programmed in the queue priority register (QUEPRI) in the EDMA3CC.
		0	Priority 0 - Highest priority
		1h-6h	Priority 1 to priority 6
		7h	Priority 7 - Lowest priority
3-2	Reserved	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

Table 17-74. Source Active Options Register (SAOPT) Field Descriptions (continued)

Bit	Field	Value	Description
1	DAM		Destination address mode within an array.
		0	Increment (INCR) mode. Destination addressing within an array increments.
		1	Constant addressing (CONST) mode. Destination addressing within an array wraps around upon reaching FIFO width.
0	SAM		Source address mode within an array.
		0	Increment (INCR) mode. Source addressing within an array increments.
		1	Constant addressing (CONST) mode. Source addressing within an array wraps around upon reaching FIFO width.

17.4.3.6.2 Source Active Source Address Register (SASRC)

The source active source address register (SASRC) is shown in Figure 17-92 and described in Table 17-75.

Figure 17-92. Source Active Source Address Register (SASRC)

31	0
SADDR	
R-0	

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

Table 17-75. Source Active Source Address Register (SASRC) Field Descriptions

Bit	Field	Value	Description
31-0	SADDR	0-FFFF FFFFh	Source address for program register set. EDMA3TC updates value according to source addressing mode (SAM bit in the source active options register, SAOPT) .

17.4.3.6.3 Source Active Count Register (SACNT)

The source active count register (SACNT) is shown in Figure 17-93 and described in Table 17-76.

Figure 17-93. Source Active Count Register (SACNT)

31	16
BCNT	
R-0	
15	0
ACNT	
R-0	

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

Table 17-76. Source Active Count Register (SACNT) Field Descriptions

Bit	Field	Value	Description
31-16	BCNT	0-FFFFh	B dimension count. Number of arrays to be transferred, where each array is ACNT in length. It is decremented after each Read command appropriately. Represents the amount of data remaining to be Read. It should be 0 when transfer request (TR) is complete.
15-0	ACNT	0-FFFFh	A dimension count. Number of bytes to be transferred in first dimension. It is decremented after each Read command appropriately. Represents the amount of data remaining to be Read. It should be 0 when transfer request (TR) is complete.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.3.6.4 Source Active Destination Address Register (SADST)

The source active destination address register (SADST) is shown in Figure 17-94 and described in Table 17-77.

Figure 17-94. Source Active Destination Address Register (SADST)

31	0
DADDR	
R-0	

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

Table 17-77. Source Active Destination Address Register (SADST) Field Descriptions

Bit	Field	Value	Description
31-0	DADDR	0	Always reads as 0

17.4.3.6.5 Source Active B-Index Register (SABIDX)

The source active B-index register (SABIDX) is shown in Figure 17-95 and described in Table 17-78.

Figure 17-95. Source Active B-Index Register (SABIDX)

31	16
DSTBIDX	
R-0	
15	0
SRCBIDX	
R-0	

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

Table 17-78. Source Active B-Index Register (SABIDX) Field Descriptions

Bit	Field	Value	Description
31-16	DSTBIDX	0	B-Index offset between destination arrays. Represents the offset in bytes between the starting address of each destination. Always reads as 0.
15-0	SRCBIDX	0-FFFFh	B-Index offset between source arrays. Represents the offset in bytes between the starting address of each source array.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.3.6.6 Source Active Memory Protection Proxy Register (SAMPPRXY)

The source active memory protection proxy register (SAMPPRXY) is shown in Figure 17-96 and described in Table 17-79.

Figure 17-96. Source Active Memory Protection Proxy Register (SAMPPRXY)

31	16
Reserved	
R-0	

15	9	8	7	4	3	0
Reserved		PRIV	Reserved		PRIVID	
R-0		R-0	R-0		R-0	

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

Table 17-79. Source Active Memory Protection Proxy Register (SAMPPRXY) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	PRIV		Privilege level. The privilege level used by the host to set up the parameter entry in the channel controller. This field is set up when the associated TR is submitted to the EDMA3TC. The privilege ID is used while issuing Read and write command to the target endpoints so that the target endpoints can perform memory protection checks based on the PRIV of the host that set up the DMA transaction.
		0	User-level privilege
		1	Supervisor-level privilege
7-4	Reserved	0	Reserved
3-0	PRIVID	0-Fh	Privilege ID. This contains the privilege ID of the host that set up the parameter entry in the channel controller. This field is set up when the associated TR is submitted to the EDMA3TC. This PRIVID value is used while issuing Read and write commands to the target endpoints so that the target endpoints can perform memory protection checks based on the PRIVID of the host that set up the DMA transaction.
		0	For any other master that sets up the PaRAM entry.
		1	If CPU sets up the PaRAM entry.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.3.6.7 Source Active Count Reload Register (SACNTRLD)

The source active count reload register (SACNTRLD) is shown in Figure 17-97 and described in Table 17-80.

Figure 17-97. Source Active Count Reload Register (SACNTRLD)

31	16
Reserved	
R-0	

15	0
ACNTRLD	
R-0	

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

Table 17-80. Source Active Count Reload Register (SACNTRLD) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	ACNTRLD	0-FFFFh	A-count reload value. Represents the originally programmed value of ACNT. The reload value is used to reinitialize ACNT after each array is serviced.

17.4.3.6.8 Source Active Source Address B-Reference Register (SASRCBREF)

The source active source address B-reference register (SASRCBREF) is shown in Figure 17-98 and described in Table 17-81.

Figure 17-98. Source Active Source Address B-Reference Register (SASRCBREF)

31	0
SADDRBREF	
R-0	

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

Table 17-81. Source Active Source Address B-Reference Register (SASRCBREF) Field Descriptions

Bit	Field	Value	Description
31-0	SADDRBREF	0-FFFF FFFFh	Source address B-reference. Represents the starting address for the array currently being Read.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.3.6.9 Source Active Destination Address B-Reference Register (SADSTBREF)

The source active destination address B-reference register (SADSTBREF) is shown in Figure 17-99 and described in Table 17-82.

Figure 17-99. Source Active Destination Address B-Reference Register (SADSTBREF)

31	0
DADDRBREF	
R-0	

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

Table 17-82. Source Active Destination Address B-Reference Register (SADSTBREF) Field Descriptions

Bit	Field	Value	Description
31-0	DADDRBREF	0	Always reads as 0

17.4.3.6.10 Destination FIFO Set Count Reload Register (DFCNTRLD)

The destination FIFO set count reload register (DFCNTRLD) is shown in Figure 17-100 and described in Table 17-83.

Figure 17-100. Destination FIFO Set Count Reload Register (DFCNTRLD)

31	16
Reserved	
R-0	

15	0
ACNTRLD	
R-0	

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

Table 17-83. Destination FIFO Set Count Reload Register (DFCNTRLD) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	ACNTRLD	0-FFFFh	A-count reload value. Represents the originally programmed value of ACNT. The reload value is used to reinitialize ACNT after each array is serviced.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.3.6.11 Destination FIFO Set Source Address B-Reference Register (DFSRCBREF)

The destination FIFO set source address B-reference register (DFSRCBREF) is shown in Figure 17-101 and described in Table 17-84.

Figure 17-101. Destination FIFO Set Source Address B-Reference Register (DFSRCBREF)

31	0
SADDRBREF	
R-0	

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

Table 17-84. Destination FIFO Set Source Address B-Reference Register (DFSRCBREF) Field Descriptions

Bit	Field	Value	Description
31-0	SADDRBREF	0	Not applicable. Always Read as 0.

17.4.3.6.12 Destination FIFO Set Destination Address B-Reference (DFDSTBREF)

The destination FIFO set destination address B-reference register (DFDSTBREF) is shown in Figure 17-102 and described in Table 17-85.

Figure 17-102. Destination FIFO Set Destination Address B-Reference Register (DFDSTBREF)

31	0
DADDRBREF	
R-0	

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

Table 17-85. Destination FIFO Set Destination Address B-Reference Register (DFDSTBREF) Field Descriptions

Bit	Field	Value	Description
31-0	DADDRBREF	0-FFFF FFFFh	Destination address reference for the destination FIFO register set. Represents the starting address for the array currently being written.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.3.6.13 Destination FIFO Options Register *n* (DFOPT*n*)

The destination FIFO options register *n* (DFOPT*n*) is shown in Figure 17-103 and described in Table 17-86.

Figure 17-103. Destination FIFO Options Register *n* (DFOPT*n*)

31	23	22	21	20	19	18	17	16
Reserved		TCCHEN	Rsvd	TCINTEN	Reserved		TCC	
R-0		R/W-0	R-0	R/W-0	R-0		R/W-0	

15	12	11	10	8	7	6	4	3	2	1	0
TCC		Rsvd	FWID		Rsvd	PRI		Reserved		DAM	SAM
R/W-0		R-0	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 17-86. Destination FIFO Options Register *n* (DFOPT*n*) Field Descriptions

Bit	Field	Value	Description
31-23	Reserved	0	Reserved
22	TCCHEN		Transfer complete chaining enable.
		0	Transfer complete chaining is disabled.
		1	Transfer complete chaining is enabled.
21	Reserved	0	Reserved
20	TCINTEN		Transfer complete interrupt enable.
		0	Transfer complete interrupt is disabled.
		1	Transfer complete interrupt is enabled.
19-18	Reserved	0	Reserved
17-12	TCC	0-3Fh	Transfer complete code. This 6-bit code is used to set the relevant bit in CER or IPR of the EDMA3CC.
11	Reserved	0	Reserved
10-8	FWID	0-7h	FIFO width. Applies if either SAM or DAM is set to constant addressing mode.
		0	FIFO width is 8 bits.
		1h	FIFO width is 16 bits.
		2h	FIFO width is 32 bits.
		3h	FIFO width is 64 bits.
		4h	FIFO width is 128 bits.
		5h-7h	Reserved
7	Reserved	0	Reserved
6-4	PRI	0-7h	Transfer priority.
		0	Priority 0 - Highest priority
		1h-6h	Priority 1 to priority 6
		7h	Priority 7 - Lowest priority
3-2	Reserved	0	Reserved
1	DAM		Destination address mode within an array.
		0	Increment (INCR) mode. Destination addressing within an array increments.
		1	Constant addressing (CONST) mode. Destination addressing within an array wraps around upon reaching FIFO width.
0	SAM		Source address mode within an array.
		0	Increment (INCR) mode. Source addressing within an array increments.
		1	Constant addressing (CONST) mode. Source addressing within an array wraps around upon reaching FIFO width.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.3.6.14 Destination FIFO Source Address Register *n* (DFSRC*n*)

The destination FIFO source address register *n* (DFSRC*n*) is shown in Figure 17-104 and described in Table 17-87.

Figure 17-104. Destination FIFO Source Address Register *n* (DFSRC*n*)

31	0
SADDR	
R-0	

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

Table 17-87. Destination FIFO Source Address Register *n* (DFSRC*n*) Field Descriptions

Bit	Field	Value	Description
31-0	SADDR	0	Always Read as 0.

17.4.3.6.15 Destination FIFO Count Register *n* (DFCNT*n*)

The destination FIFO count register *n* (DFCNT*n*) is shown in Figure 17-105 and described in Table 17-88.

Figure 17-105. Destination FIFO Count Register *n* (DFCNT*n*)

31	16
BCNT	
R-0	
15	0
ACNT	
R-0	

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

Table 17-88. Destination FIFO Count Register *n* (DFCNT*n*) Field Descriptions

Bit	Field	Value	Description
31-16	BCNT	0-FFFFh	B-dimension count. Number of arrays to be transferred, where each array is ACNT in length. Count/count remaining for destination register set. Represents the amount of data remaining to be written.
15-0	ACNT	0-FFFFh	A-dimension count. Number of bytes to be transferred in first dimension count/count remaining for destination register set. Represents the amount of data remaining to be written.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.3.6.16 Destination FIFO Destination Address Register *n* (DFDST*n*)

The destination FIFO destination address register *n* (DFDST*n*) is shown in Figure 17-106 and described in Table 17-89.

Figure 17-106. Destination FIFO Destination Address Register *n* (DFDST*n*)

31	0
DADDR	
R-0	

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

Table 17-89. Destination FIFO Destination Address Register *n* (DFDST*n*) Field Descriptions

Bit	Field	Value	Description
31-0	DADDR	0	Destination address for the destination FIFO register set. When a transfer request (TR) is complete, the final value should be the address of the last write command issued.

17.4.3.6.17 Destination FIFO B-Index Register *n* (DFBIDX*n*)

The destination FIFO B-index register *n* (DFBIDX*n*) is shown in Figure 17-107 and described in Table 17-90.

Figure 17-107. Destination FIFO B-Index Register *n* (DFBIDX*n*)

31	16
DSTBIDX	
R-0	
15	0
SRCBIDX	
R-0	

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

Table 17-90. Destination FIFO B-Index Register *n* (DFBIDX*n*) Field Descriptions

Bit	Field	Value	Description
31-16	DSTBIDX	0-FFFFh	B-Index offset between destination arrays. Represents the offset in bytes between the starting address of each destination.
15-0	SRCBIDX	0	B-Index offset between source arrays. Represents the offset in bytes between the starting address of each source array. Always Read as 0.

Copyright © 2013–2016, Texas Instruments Incorporated

17.4.3.6.18 Destination FIFO Memory Protection Proxy Register *n* (DFMPPRXY*n*)

The destination FIFO memory protection proxy register *n* (DFMPPRXY*n*) is shown in Figure 17-108 and described in Table 17-91.

Figure 17-108. Destination FIFO Memory Protection Proxy Register *n* (DFMPPRXY*n*)

31	16
Reserved	
R-0	

15	9	8	7	4	3	0
Reserved		PRIV	Reserved		PRIVID	
R-0		R-0	R-0		R-0	

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

Table 17-91. Destination FIFO Memory Protection Proxy Register *n* (DFMPPRXY*n*) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	PRIV		Privilege level. This contains the privilege level used by the EDMA programmer to set up the parameter entry in the channel controller. This field is set up when the associated TR is submitted to the EDMA3TC. The privilege ID is used while issuing Read and write command to the target endpoints so that the target endpoints can perform memory protection checks based on the PRIV of the host that set up the DMA transaction.
		0	User-level privilege
		1	Supervisor-level privilege
7-4	Reserved	0	Reserved
3-0	PRIVID	0-Fh	Privilege ID. This contains the Privilege ID of the EDMA programmer that set up the parameter entry in the channel controller. This field is set up when the associated TR is submitted to the EDMA3TC. This PRIVID value is used while issuing Read and write commands to the target endpoints so that the target endpoints can perform memory protection checks based on the PRIVID of the host that set up the DMA transaction.
		0	For any other master that sets up the PaRAM entry
		1	If CPU sets up the PaRAM entry

Copyright © 2013–2016, Texas Instruments Incorporated

17.5 Tips

17.5.1 Debug Checklist

This section lists some tips to keep in mind while debugging applications using the EDMA3. Table 17-92 provides some common issues and their probable causes and resolutions.

Table 17-92. Debug List

Issue	Description/Solution
The transfer associated with the channel does not happen. The channel does not get serviced.	The EDMA3 channel controller (EDMA3CC) may not service a transfer request, even though the associated PaRAM set is programmed appropriately. Check for the following: 1) Verify that events are enabled, that is, if an external/peripheral event is latched in the event register (ER), make sure that the event is enabled in the event enable register (EER). Similarly for QDMA channels, make sure that QDMA events are appropriately enabled in the QDMA event enable register (QEER). 2) Verify that the DMA or QDMA secondary event register (SER) bits corresponding to the particular event or channel are not set.
The secondary event register bits are set, not allowing additional transfers to occur on a channel.	It is possible that a trigger event was received when the parameter set associated with the channel/event was a NULL set for a previous transfer on the channel. This is typical in two cases: 1) QDMA channels: Typically if the parameter set is nonstatic and expected to be terminated by a NULL set (OPT.STATIC = 0, LINK = FFFFh), the parameter set is updated with a NULL set after submission of the last TR. Because QDMA channels are autotriggered, this update caused the generation of an event. An event generated for a NULL set causes an error condition and results in setting the bits corresponding to the QDMA channel in QEMR and QSER. This will disable further prioritization of the channel. 2) DMA channels used in a continuous mode: The peripheral may be set up to continuously generate infinite events (for instance, in case of the McBSP, every time the data shifts out from DXR, it generates an XEVT). The parameter set may be programmed to expect only a finite number of events and to be terminated by a NULL link. After the expected number of events, the parameter set is reloaded with a NULL parameter set. Because the peripheral will generate additional events, an error condition is set in SER.E*n* and EMR.E*n*, preventing further event prioritization. You must ensure that the number of events received is limited to the expected number of events for which the parameter set is programmed, or you must ensure that bits corresponding to a particular channel or event are not set in the secondary event registers (SER/QSER) and the event missed registers (EMR/QEMR) before trying to perform subsequent transfers for the event/channel.
Completion interrupts are not asserted, or no further interrupts are received after the first completion interrupt.	You must ensure the following: 1) The interrupt generation is enabled in the OPT of the associated PaRAM set (TCINTEN = 1 and/or ITCINTEN = 1). 2) The interrupts are enabled in the EDMA3 channel controller (EDMA3CC), via the interrupt enable register (IER). 3) The corresponding interrupts are enabled in the device interrupt controller. 4) The set interrupts are cleared in the interrupt pending register (IPR) before exiting the transfer completion interrupt service routine (ISR). See Section 17.2.9.1.2 for details on writing EDMA3 ISRs. 5) If working with shadow region interrupts, make sure that the DMA region access enable registers (DRAE) are set up properly, because DRAE act as secondary enables for shadow region completion interrupts, along with IER. If working with shadow region interrupts, make sure that the bits corresponding to the transfer completion code (TCC) value are also enabled in DRAE. For instance, if the PaRAM set associated with channel 0 returns a completion code of 31 (OPT.TCC = 31), make sure that DRAE.E31 is also set for a shadow region completion interrupt because the interrupt pending register bit set will be IPR.I31.

Copyright © 2013–2016, Texas Instruments Incorporated

17.5.2 Miscellaneous Programming/Debug Tips

1. For several registers, the setting and clearing of bits needs to be done via separate dedicated registers. For example, the event register (ER) bits can only be cleared by writing a 1 to the corresponding bits in the event clear register (ECR). Similarly, the event enable register (EER) bits can only be set with writes of 1 to the corresponding bits in the event enable set registers (EESR) and can only be cleared with writes of 1 to the corresponding bits in the event enable clear register (EECR).
2. Writes to the shadow region memory maps are governed by region access enable registers (DRAE/QRAE). If the appropriate channels are not enabled in these registers, read/write access to the shadow region memory map is not enabled.
3. When working with shadow region completion interrupts, ensure that the DMA region access enable registers (DRAE) for every region are set in a mutually exclusive way (unless it is a requirement for an application). If there is an overlap in the allocated channels and transfer completion codes (setting of interrupt pending register bits) in the region resource allocation, it results in multiple shadow region completion interrupts. For example, if DRAE0.E0 and DRAE1.E0 are both set, then on completion of a transfer that returns a TCC = 0, they will generate both shadow region 0 and 1 completion interrupts.
4. While programming a non-dummy parameter set, ensure the CCNT is not left to zero.
5. Enable the EDMA3CC error interrupt in the device controller and attach an interrupt service routine (ISR) to ensure that error conditions are not missed in an application and are appropriately addressed with the ISR.
6. Depending on the application, you may want to break large transfers into smaller transfers and use self-chaining to prevent starvation of other events in an event queue.
7. In applications where a large transfer is broken into sets of small transfers using chaining or other methods, you might choose to use the early chaining option to reduce the time between the sets of transfers and increase the throughput. However, keep in mind that with early completion, all data might have not been received at the end point when completion is reported because the EDMA3CC internally signals completion when the TR is submitted to the EDMA3TC, potentially before any data has been transferred.
8. The event queue entries can be observed to determine the last few events if there is a system failure (provided the entries were not bypassed).
9. In order to put the EDMA3CC and EDMA3TC in power-down modes, you should ensure that there is no activity with the EDMA3CC and EDMA3TC. The EDMA3CC status register (CCSTAT) and the EDMA3TC channel status register (TCSTAT) should be used.

Copyright © 2013–2016, Texas Instruments Incorporated

17.6 Setting Up a Transfer

The following list provides a quick guide for the typical steps involved in setting up a transfer.

1. Initiating a DMA/QDMA channel:
   (a) Determine the type of channel (QDMA or DMA) to be used.
   (b) If using a QDMA channel, program the QDMA channel *n* mapping register (QCHMAP*n*) with the parameter set number to which the channel maps and the trigger word.
   (c) If the channel is being used in the context of a shadow region, ensure the DMA region access enable register (DRAE) for the region is properly set up to allow read/write accesses to bits in the event register and interrupt register in the shadow region memory-map. The subsequent steps in this process should be done using the respective shadow region registers. (Shadow region descriptions and usage are provided in Section 17.2.7.1.)
   (d) Determine the type of triggering used.
      (i) If external events are used for triggering (DMA channels), enable the respective event in EER by writing into EESR.
      (ii) If a QDMA channel is used, enable the channel in QEER by writing into QEESR.
   (e) Queue setup.
      (i) If a QDMA channel is used, set up QDMAQNUM to map the channel to the respective event queue.
      (ii) If a DMA channel is used, set up DMAQNUM to map the event to the respective event queue.
2. Parameter set setup: Program the PaRAM set number associated with the channel. Note that if it is a QDMA channel, the PaRAM entry that is configured as trigger word is written last. Alternatively, enable the QDMA channel just before the write to the trigger word.
   See Section 17.3 for parameter set field setups for different types of transfers. See the sections on chaining (Section 17.2.8) and interrupt completion (Section 17.2.9) on how to set up final/intermediate completion chaining and/or interrupts.
3. Interrupt setup:
   (a) If working in the context of a shadow region, ensure the relevant bits in DRAE are set.
   (b) Enable the interrupt in IER by writing into IESR.
   (c) Ensure that the EDMA3CC completion interrupt is enabled properly in the device interrupt controller.
   (d) Set up the interrupt controller properly to receive the expected EDMA3 interrupt.
4. Initiate transfer (this step is highly dependent on the event trigger source):
   (a) If the source is an external event coming from a peripheral, the peripheral will be enabled to start generating relevant EDMA3 events that can be latched to the ER transfer.
   (b) For QDMA events, writes to the trigger word will initiate the transfer.
   (c) Manually-triggered transfers will be initiated by writes to the event set register (ESR).
   (d) Chained-trigger events initiate when a previous transfer returns a transfer completion code equal to the chained channel number.
5. Wait for completion:
   (a) If the interrupts are enabled as mentioned in step 3, then the EDMA3CC generates a completion interrupt to the CPU whenever transfer completion results in setting the corresponding bits in the interrupt pending register (IPR). The set bits must be cleared in IPR by writing to the corresponding bit in ICR.
   (b) If polling for completion (interrupts not enabled in the device controller), then the application code can wait on the expected bits to be set in IPR. Again, the set bits in IPR must be manually cleared by writing to ICR before the next set of transfers is performed for the same transfer completion code values.

Copyright © 2013–2016, Texas Instruments Incorporated

External Memory Interface A (EMIFA)

This chapter describes the external memory interface A (EMIFA).

The EMIFA SDRAM interface is not supported on all devices, see your device-specific data manual to see if the EMIFA SDRAM is supported on your device.

Topic **Page**

18.1 Introduction **697**
18.2 Architecture **697**
18.3 Example Configuration **738**
18.4 Registers **760**

Copyright © 2013–2016, Texas Instruments Incorporated

18.1 Introduction

18.1.1 Purpose of the Peripheral

EMIFA memory controller is complaint with the JESD21-C SDR SDRAM memories utilizing 16-bit data bus of EMIFA memory controller. The purpose of this EMIFA is to provide a means for the CPU to connect to a variety of external devices including:

- Single data rate (SDR) SDRAM
- Asynchronous devices including NOR Flash, NAND Flash, and SRAM

The most common use for the EMIFA is to interface with both a flash device and an SDRAM device simultaneously. Section 18.3 contains an example of operating the EMIFA in this configuration.

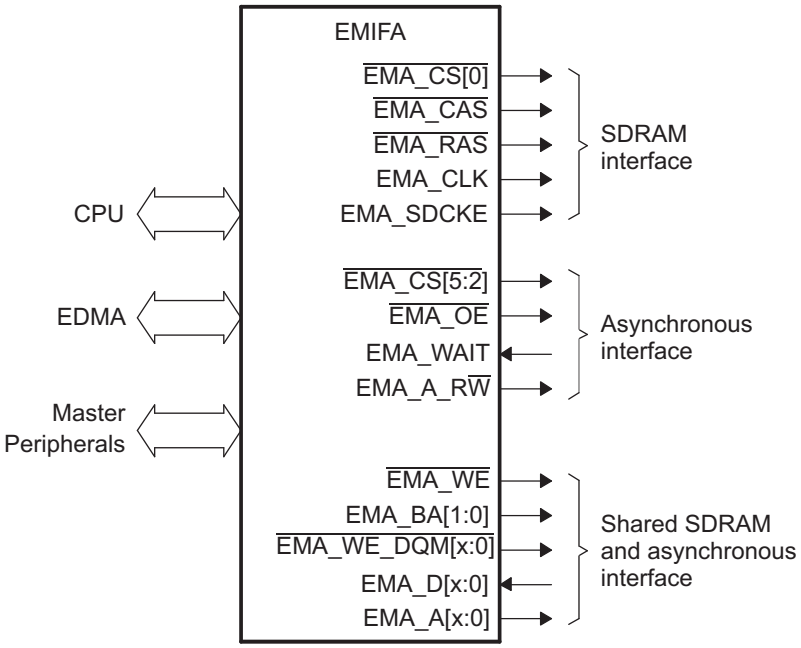
18.1.2 Features

The EMIFA includes many features to enhance the ease and flexibility of connecting to external SDR SDRAM and asynchronous devices. For details on features of EMIFA, see your device-specific data manual.

18.1.3 Functional Block Diagram

Figure 18-1 illustrates the connections between the EMIFA and its internal requesters, along with the external EMIFA pins. Section 18.2.2 contains a description of the entities internal to the SoC that can send requests to the EMIFA, along with their prioritization. Section 18.2.3 describes the EMIFA external pins and summarizes their purpose when interfacing with SDRAM and asynchronous devices.

Figure 18-1. EMIFA Functional Block Diagram



18.2 Architecture

This section provides details about the architecture and operation of the EMIFA. Both, SDRAM and asynchronous interface are covered, along with other system-related issues such as clock control and pin multiplexing.

The EMIFA SDRAM interface is not supported on all devices, see your device-specific data manual to see if the EMIFA SDRAM is supported on your device.

Copyright © 2013–2016, Texas Instruments Incorporated

18.2.1 Clock Control

The EMIFA clock is output on the EMA_CLK pin and should be used when interfacing to external memories. The EMIFA clock (EMA_CLK) does not run during device reset. When the RESET pin is released and after the PLL controller releases the device from reset, EMA_CLK begins to oscillate at a frequency determined by the PLL controller.

For details on clock generation and control, see the *Device Clocking* chapter.

18.2.2 EMIFA Requests

Different sources within the SoC can make requests to the EMIFA. These requests consist of accesses to SDRAM memory, asynchronous memory, and EMIFA registers. Because the EMIFA can process only one request at a time, a high performance crossbar switch exists within the SoC to provide prioritized requests from the different sources to the EMIFA. The sources are:

1. CPU
2. EDMA
3. Other master peripherals

If a request is submitted from two or more sources simultaneously, the crossbar switch will forward the highest priority request to the EMIFA first. Upon completion of a request, the crossbar switch again evaluates the pending requests and forwards the highest priority pending request to the EMIFA.

When the EMIFA receives a request, it may or may not be immediately processed. In some cases, the EMIFA will perform one or more auto refresh cycles before processing the request. For details on the EMIFA's internal arbitration between performing requests and performing auto refresh cycles, see Section 18.2.12.

18.2.3 Pin Descriptions

This section describes the function of each of the EMIFA pins.

Table 18-1. EMIFA Pins Used to Access Both SDRAM and Asynchronous Memories

Pins(s)	I/O	Description
EMA_D[x:0]	I/O	**EMIFA data bus.** The number of available data bus pins varies among devices, see your device-specific data manual for details.
EMA_ A[x:0]	O	**EMIFA address bus.** The number of available address pins varies among devices, see your device-specific data manual for details. When interfacing to an SDRAM device, these pins are primarily used to provide the row and column address to the SDRAM. The mapping from the internal program address to the external values placed on these pins can be found in Section 18.2.4.11. EMA_A[10] is also used during the PRE command to select which banks to deactivate. When interfacing to an asynchronous device, these pins are used in conjunction with the EMA_BA pins to form the address that is sent to the device. The mapping from the internal program address to the external values placed on these pins can be found in Section 18.2.5.1.
EMA_BA[1:0]	O	**EMIFA bank address.** When interfacing to an SDRAM device, these pins are used to provide the bank address inputs to the SDRAM. The mapping from the internal program address to the external values placed on these pins can be found in Section 18.2.4.11. When interfacing to an asynchronous device, these pins are used in conjunction with the EMA_A pins to form the address that is sent to the device. The mapping from the internal program address to the external values placed on these pins can be found in Section 18.2.5.1.
EMA_WE_DQM[x:0]	O	**Active-low byte enables.** When interfacing to SDRAM, these pins are connected to the DQM pins of the SDRAM to individually enable/disable each of the bytes in a data access. When interfacing to an asynchronous device, these pins are connected to byte enables. See Section 18.2.5 for details.
EMA_WE	O	**Active-low write enable.** When interfacing to SDRAM, this pin is connected to the WE pin of the SDRAM and is used to send commands to the device. When interfacing to an asynchronous device, this pin provides a signal which is active-low during the strobe period of an asynchronous write access cycle.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 18-2. EMIFA Pins Specific to SDRAM

Pin(s)	I/O	Description
EMA_CS[0]	O	**Active-low chip enable pin for SDRAM devices.** This pin is connected to the chip-select pin of the attached SDRAM device and is used for enabling/disabling commands. By default, the EMIFA keeps this SDRAM chip select active, even if the EMIFA is not interfaced with an SDRAM device. This pin is deactivated when accessing the asynchronous memory bank and is reactivated on completion of the asynchronous assess.
EMA_RAS	O	**Active-low row address strobe pin.** This pin is connected to the RAS pin of the attached SDRAM device and is used for sending commands to the device.
EMA_CAS	O	**Active-low column address strobe pin.** This pin is connected to the CAS pin of the attached SDRAM device and is used for sending commands to the device.
EMA_SDCKE	O	**Clock enable pin.** This pin is connected to the CKE pin of the attached SDRAM device and is used for issuing the SELF REFRESH command which places the device in self refresh mode. See Section 18.2.4.7 for details.
EMA_CLK	O	**SDRAM clock pin.** This pin is connected to the CLK pin of the attached SDRAM device. See Section 18.2.1 for details on the clock signal.

Table 18-3. EMIFA Pins Specific to Asynchronous Memory

Pin(s)	I/O	Description
EMA_CS[5:2]	O	**Active-low chip enable pins for asynchronous devices.** These pins are meant to be connected to the chip-select pins of the attached asynchronous device. These pins are active only during accesses to the asynchronous memory.
EMA_WAIT	I	**Wait input with programmable polarity / NAND Flash ready input.** Not all devices support both EMA_WAIT[1] and EMA_WAIT[0], see your device-specific data manual to determine support on each device. A connected asynchronous device can extend the strobe period of an access cycle by asserting the EMA_WAIT input to the EMIFA as described in Section 18.2.5.7. To enable this functionality, the EW bit in the asynchronous *n* configuration register (CE*n*CFG) must be set to 1. The WP0 and WP1 bits in the asynchronous wait cycle configuration register (AWCC) must be configured to define the polarity of the EMA_WAIT pin. The CS*n*_WAIT bit in AWCC must also be configured to determine which EMA_WAIT[*n*] signal is used for memory accesses. When the CS2NAND/CS3NAND/CS4NAND/CS5NAND bit in the NAND Flash control register (NANDFCR) is set, this pin instead functions as a NAND Flash ready input.
EMA_OE	O	**Active-low pin enable for asynchronous devices.** This pin provides a signal which is active-low during the strobe period of an asynchronous read access cycle.
EMA_A_RW	O	**EMIFA asynchronous read/write control.** This pin stays high during reads and stays low during writes (same duration as CS).

Copyright © 2013–2016, Texas Instruments Incorporated

18.2.4 SDRAM Controller and Interface

The EMIFA can gluelessly interface to most standard SDR SDRAM devices and supports such features as self refresh mode and prioritized refresh. In addition, it provides flexibility through programmable parameters such as the refresh rate, CAS latency, and many SDRAM timing parameters. The following sections include details on how to interface and properly configure the EMIFA to perform read and write operations to externally connected SDR SDRAM devices. Also, Section 18.3 provides a detailed example of interfacing the EMIFA to a common SDRAM device.

18.2.4.1 SDRAM Commands

The EMIFA supports the SDRAM commands described in Table 18-4. The truth table for the SDRAM commands is shown in Table 18-5 and an example timing waveform of the PRE command is shown in Figure 18-2. EMA_A[10] is pulled low in this example to deactivate only the bank specified by the EMA_BA pins.

Table 18-4. EMIFA SDRAM Commands

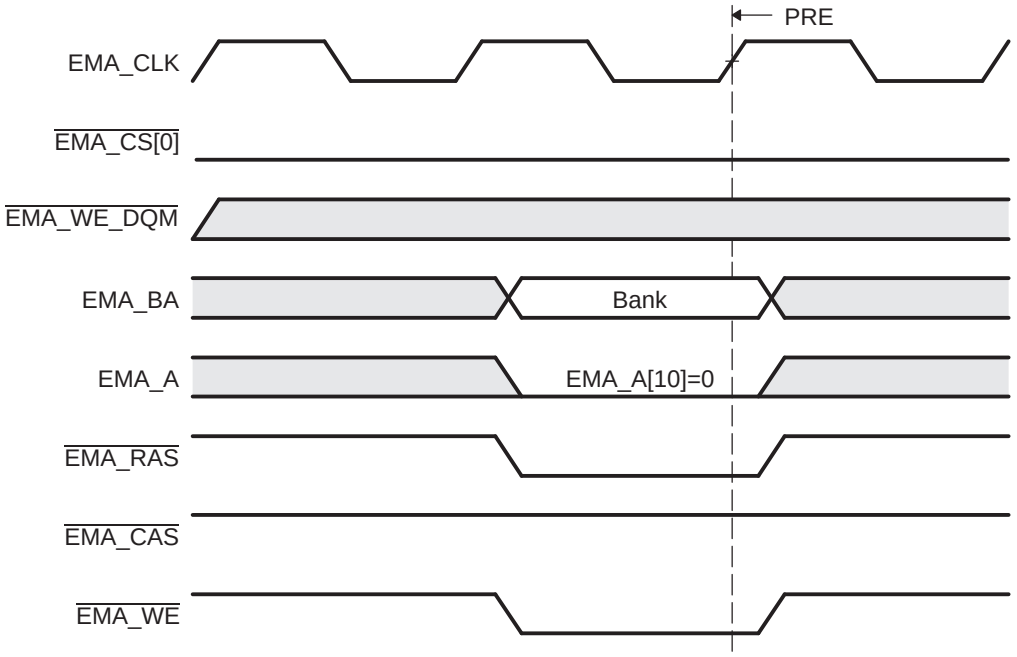
Command	Function
PRE	**Precharge.** Depending on the value of EMA_A[10], the PRE command either deactivates the open row in all banks (EMA_A[10] = 1) or only the bank specified by the EMA_BA[1:0] pins (EMA_A[10] = 0).
ACTV	**Activate**. The ACTV command activates the selected row in a particular bank for the current access.
READ	**Read.** The READ command outputs the starting column address and signals the SDRAM to begin the burst read operation. Address EMA_A[10] is always pulled low to avoid auto precharge. This allows for better bank interleaving performance.
WRT	**Write.** The WRT command outputs the starting column address and signals the SDRAM to begin the burst write operation. Address EMA_A[10] is always pulled low to avoid auto precharge. This allows for better bank interleaving performance.
BT	**Burst terminate**. The BT command is used to truncate the current read or write burst request.
LMR	**Load mode register**. The LMR command sets the mode register of the attached SDRAM devices and is only issued during the SDRAM initialization sequence described in Section 18.2.4.4.
REFR	**Auto refresh**. The REFR command signals the SDRAM to perform an auto refresh according to its internal address.
SLFR	**Self refresh**. The self refresh command places the SDRAM into self refresh mode, during which it provides its own clock signal and auto refresh cycles.
NOP	**No operation**. The NOP command is issued during all cycles in which one of the above commands is not issued.

Table 18-5. Truth Table for SDRAM Commands

SDRAM Pins:	CKE	$\overline{CS}$	$\overline{RAS}$	$\overline{CAS}$	$\overline{WE}$	BA[1:0]	A[12:11]	A[10]	A[9:0]
EMIFA Pins:	EMA_SDCKE	$\overline{EMA_CS[0]}$	$\overline{EMA_RAS}$	$\overline{EMA_CAS}$	$\overline{EMA_WE}$	EMA_BA[1:0]	EMA_A[12:11]	EMA_A[10]	EMA_A[9:0]
PRE	H	L	L	H	L	Bank/X	X	L/H	X
ACTV	H	L	L	H	H	Bank	Row	Row	Row
READ	H	L	H	L	H	Bank	Column	L	Column
WRT	H	L	H	L	L	Bank	Column	L	Column
BT	H	L	H	H	L	X	X	X	X
LMR	H	L	L	L	L	X	Mode	Mode	Mode
REFR	H	L	L	L	H	X	X	X	X
SLFR	L	L	L	L	H	X	X	X	X
NOP	H	L	H	H	H	X	X	X	X

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 18-2. Timing Waveform of SDRAM PRE Command



18.2.4.2 Interfacing to SDRAM

The EMIFA supports a glueless interface to SDRAM devices with the following characteristics:

- Pre-charge bit is A[10]
- The number of column address bits is 8, 9, 10, or 11. See your device-specific data manual for the number of column address bits supported on your device.
- The number of row address bits is 13, 14, 15, or 16. See your device-specific data manual for the number of row address bits supported on your device.
- The number of internal banks is 1, 2, or 4. See your device-specific data manual for the number of internal banks supported on your device.

Figure 18-3 shows an interface between the EMIFA and a 2M × 16 × 4 bank SDRAM device, and Figure 18-4 shows an interface between the EMIFA and a 512K × 16 × 2 bank SDRAM device. For devices supporting 16-bit interface, refer to Table 18-6 for list of commonly-supported SDRAM devices and the required connections for the address pins.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 18-3. EMIFA to 2M × 16 × 4 bank SDRAM Interface

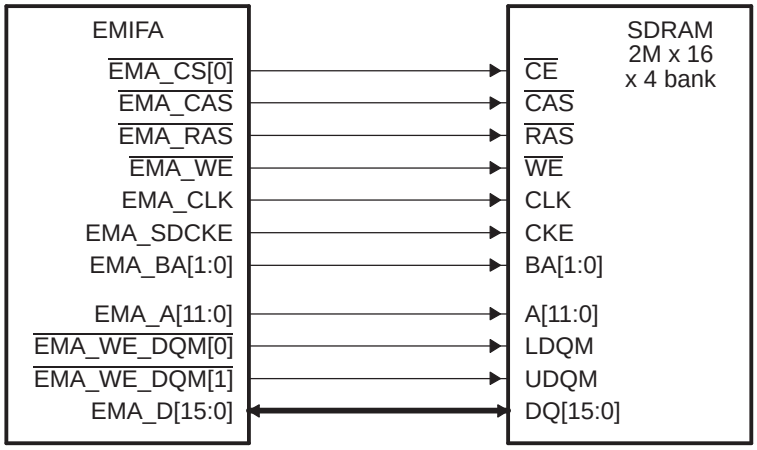


Figure 18-4. EMIFA to 512K × 16 × 2 bank SDRAM Interface

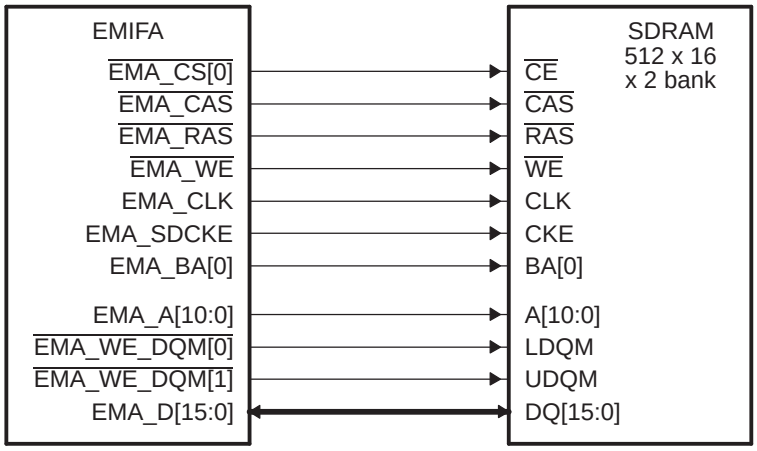


Table 18-6. 16-bit EMIFA Address Pin Connections

SDRAM Size	Width	Banks	Device	Address Pins
16M bits	×16	2	SDRAM	A[10:0]
			EMIFA	EMA_A[10:0]
64M bits	×16	4	SDRAM	A[11:0]
			EMIFA	EMA_A[11:0]
128M bits	×16	4	SDRAM	A[11:0]
			EMIFA	EMA_A[11:0]
256M bits	x16	4	SDRAM	A[12:0]
			EMIFA	EMA_A[12:0]
512M bits	x16	4	SDRAM	A[12:0]
			EMIFA	EMA_A[12:0]

Copyright © 2013–2016, Texas Instruments Incorporated

18.2.4.3 SDRAM Configuration Registers

The operation of the EMIFA's SDRAM interface is controlled by programming the appropriate configuration registers. This section describes the purpose and function of each configuration register, but Section 18.4 should be referred for a more detailed description of each register, including the default registers values and bit-field positions. The following tables list the four such configuration registers, along with a description of each of their programmable fields.

NOTE: Writing to any of the fields: NM, CL, IBANK, and PAGESIZE in the SDRAM configuration register (SDCR) causes the EMIFA to abandon whatever it is currently doing and trigger the SDRAM initialization procedure described in Section 18.2.4.4.

Table 18-7. Description of the SDRAM Configuration Register (SDCR)

Parameter	Description
SR	This bit controls entering and exiting of the Self-Refresh mode. The field should be written using a byte-write to the upper byte of SDCR to avoid triggering the SDRAM initialization sequence.
PD	This bit controls entering and exiting of the Power down mode. The field should be written using a byte-write to the upper byte of SDCR to avoid triggering the SDRAM initialization sequence. If both SR and PD bits are set, the EMIFA will go into Self Refresh.
PDWR	Perform refreshes during Power Down. Writing a 1 to this bit will cause the EMIFA to exit the power down state and issue an AUTO REFRESH command every time Refresh May level is set. The field should be written using a byte-write to the upper byte of SDCR to avoid triggering the SDRAM initialization sequence. This bit should be set along with PD when entering power-down mode.
NM	**Narrow Mode.** This bit defines the width of the data bus between the EMIFA and the attached SDRAM device. When set to 1, the data bus is set to 16-bits. When set to 0, the data bus is set to 32-bits. This bit must always be set to 1.
CL	**CAS latency.** This field defines the number of clock cycles between when an SDRAM issues a READ command and when the first piece of data appears on the bus. The value in this field is sent to the attached SDRAM device via the LOAD MODE REGISTER command during the SDRAM initialization procedure as described in Section 18.2.4.4. Only, values of 2h (CAS latency = 2) and 3h (CAS latency = 3) are supported and should be written to this field. A 1 must be simultaneously written to the BIT11_9LOCK bit field of SDCR in order to write to the CL bit field.
IBANK	**Number of Internal SDRAM Banks.** This field defines the number of banks inside the attached SDRAM devices in the following way: • When IBANK = 0, 1 internal bank is used • When IBANK = 1h, 2 internal banks are used • When IBANK = 2h, 4 internal banks are used This field value affects the mapping of logical addresses to SDRAM row, column, and bank addresses. See Section 18.2.4.11 for details.
PAGESIZE	**Page Size.** This field defines the internal page size of the attached SDRAM devices in the following way: • When PAGESIZE = 0, 256-word pages are used • When PAGESIZE = 1h, 512-word pages are used • When PAGESIZE = 2h, 1024-word pages are used • When PAGESIZE = 3h, 2048-word pages are used This field value affects the mapping of logical addresses to SDRAM row, column, and bank addresses. See Section 18.2.4.11 for details.

Table 18-8. Description of the SDRAM Refresh Control Register (SDRCR)

Parameter	Description
RR	**Refresh Rate**. This field controls the rate at which attached SDRAM devices will be refreshed. The following equation can be used to determine the required value of RR for an SDRAM device: • RR = f_{EMA_CLK} / (Required SDRAM Refresh Rate) More information about the operation of the SDRAM refresh controller can be found in Section 18.2.4.6.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 18-9. Description of the SDRAM Timing Register (SDTIMR)

Parameter	Description
T_RFC T_RP T_RCD T_WR T_RAS T_RC T_RRD	**SDRAM Timing Parameters.** These fields configure the EMIFA to comply with the AC timing requirements of the attached SDRAM devices. This allows the EMIFA to avoid violating SDRAM timing constraints and to more efficiently schedule its operations. More details about each of these parameters can be found in the register description in Section 18.4.6. These parameters should be set to satisfy the corresponding timing requirements found in the SDRAM's datasheet.

Table 18-10. Description of the SDRAM Self Refresh Exit Timing Register (SDSRETR)

Parameter	Description
T_XS	**Self Refresh Exit Parameter.** The T_XS field of this register informs the EMIFA about the minimum number of EMA_CLK cycles required between exiting Self Refresh and issuing any command. This parameter should be set to satisfy the t_{XSR} value for the attached SDRAM device.

18.2.4.4 SDRAM Auto-Initialization Sequence

The EMIFA automatically performs an SDRAM initialization sequence, regardless of whether it is interfaced to an SDRAM device, when either of the following two events occur:

- The EMIFA comes out of reset. No memory accesses to the SDRAM and Asynchronous interfaces are performed until this auto-initialization is complete.
- A write is performed to any of the three least significant bytes of the SDRAM configuration register (SDCR)

An SDRAM initialization sequence consists of the following steps:

1. If the initialization sequence is activated by a write to SDCR, and if any of the SDRAM banks are open, the EMIFA issues a PRE command with EMA_A[10] held high to indicate all banks. This is done so that the maximum ACTV to PRE timing for an SDRAM is not violated.
2. The EMIFA drives EMA_SDCKE high and begins continuously issuing NOP commands until eight SDRAM refresh intervals have elapsed. An SDRAM refresh interval is equal to the value of the RR field of SDRAM refresh control register (SDRCR), divided by the frequency of EMA_CLK (RR/f_{EMA_CLK}). This step is used to avoid violating the Power-up constraint of most SDRAM devices that requires 200 μs (sometimes 100 μs) between receiving stable Vdd and CLK and the issuing of a PRE command. Depending on the frequency of EMA_CLK, this step may or may not be sufficient to avoid violating the SDRAM constraint. See Section 18.2.4.5 for more information.
3. After the refresh intervals have elapsed, the EMIFA issues a PRE command with EMA_A[10] held high to indicate all banks.
4. The EMIFA issues eight AUTO REFRESH commands.
5. The EMIFA issues the LMR command with the EMA_A[9:0] pins set as described in Table 18-11.
6. Finally, the EMIFA performs a refresh cycle, which consists of the following steps:

   (a) Issuing a PRE command with EMA_A[10] held high if any banks are open

   (b) Issuing an REF command

Copyright © 2013–2016, Texas Instruments Incorporated

Table 18-11. SDRAM LOAD MODE REGISTER Command

EMA_A[9:7]	EMA_A[6:4]	EMA_A[3]	EMA_A[2:0]
0 (Write bursts are of the programmed burst length in EMA_A[2:0])	These bits control the CAS latency of the SDRAM and are set according to CL field in the SDRAM configuration register (SDCR) as follows: • If CL = 2, EMA_A[6:4] = 2h (CAS latency = 2) • If CL = 3, EMA_A[6:4] = 3h (CAS latency = 3)	0 (Sequential Burst Type. Interleaved Burst Type not supported)	These bits control the burst length of the SDRAM and are set according to the NM field in the SDRAM configuration register (SDCR) as follows: • If NM = 0, EMA_A[2:0] = 2h (Burst Length = 4) • If NM = 1, EMA_A[2:0] = 3h (Burst Length = 8)

18.2.4.5 SDRAM Configuration Procedure

There are two different SDRAM configuration procedures. Although EMIFA automatically performs the SDRAM initialization sequence described in Section 18.2.4.4 when coming out of reset, it is recommended to follow one of the procedures listed below before performing any EMIFA memory requests. Procedure A should be followed if it is determined that the SDRAM Power-up constraint was not violated during the SDRAM Auto-Initialization Sequence detailed in Section 18.2.4.4 on coming out of Reset. The SDRAM Power-up constraint specifies that 200 μs (sometimes 100 μs) should exits between receiving stable Vdd and CLK and the issuing of a PRE command. Procedure B should be followed if the SDRAM Power-up constraint was violated. The 200 μs (100 μs) SDRAM Power-up constraint will be violated if the frequency of EMA_CLK is greater than 50 MHz (100 MHz for 100 μs SDRAM power-up constraint) during SDRAM Auto-Initialization Sequence. Procedure B should be followed if there is any doubt that the Power-up constraint was met.

Procedure A — Following is the procedure to be followed if the SDRAM Power-up constraint was NOT violated:

1. Place the SDRAM into Self-Refresh Mode by setting the SR bit of SDCR to 1. A byte-write to the upper byte of SDCR should be used to avoid restarting the SDRAM Auto-Initialization Sequence described in Section 18.2.4.4. The SDRAM should be placed into Self-Refresh mode when changing the frequency of EMA_CLK to avoid incurring the 200 μs Power-up constraint again.
2. Program the CPU's PLL Controller to provide the desired EMA_CLK clock frequency. Refer to the device Data Manual for details on programming the PLL Controller. The frequency of the memory clock must meet the timing requirements in the SDRAM manufacturer's documentation and the timing limitations shown in the electrical specifications of the device Data Manual.
3. Remove the SDRAM from Self-Refresh Mode by clearing the SR bit of SDCR to 0. A byte-write to the upper byte of SDCR should be used to avoid restarting the SDRAM Auto-Initialization Sequence described in Section 18.2.4.4.
4. Program SDTIMR and SDSRETR to satisfy the timing requirements for the attached SDRAM device. The timing parameters should be taken from the SDRAM datasheet.
5. Program the RR field of SDRCR to match that of the attached device's refresh interval. See Section 18.2.4.6.1 details on determining the appropriate value.
6. Program SDCR to match the characteristics of the attached SDRAM device. This will cause the auto-initialization sequence in Section 18.2.4.4 to be re-run. This second initialization generally takes much less time due to the increased frequency of EMA_CLK.

Procedure B — Following is the procedure to be followed if the SDRAM Power-up constraint was violated:

1. Program the CPU's PLL Controller to provide the desired EMA_CLK clock frequency. Refer to the device Data Manual for details on programming the PLL Controller. The frequency of the memory clock must meet the timing requirements in the SDRAM manufacturer's documentation and the timing limitations shown in the electrical specifications of the device Data Manual.
2. Program SDTIMR and SDSRETR to satisfy the timing requirements for the attached SDRAM device. The timing parameters should be taken from the SDRAM datasheet.

Copyright © 2013–2016, Texas Instruments Incorporated

3. Program the RR field of SDRCR such that the following equation is satisfied: (RR × 8)/(f_{EMA_CLK}) > 200 μs (sometimes 100 μs). For example, an EMA_CLK frequency of 100 MHz would require setting RR to 2501 (9C5h) or higher to meet a 200 μs constraint.
4. Program SDCR to match the characteristics of the attached SDRAM device. This will cause the auto-initialization sequence in Section 18.2.4.4 to be re-run with the new value of RR.
5. Perform a read from the SDRAM to assure that step 5 of this procedure will occur after the initialization process has completed. Alternatively, wait for 200 μs instead of performing a read.
6. Finally, program the RR field to match that of the attached device's refresh interval. See Section 18.2.4.6.1 details on determining the appropriate value.

After following the above procedure, the EMIFA is ready to perform accesses to the attached SDRAM device. See Section 18.3 for an example of configuring the SDRAM interface.

18.2.4.6 EMIFA Refresh Controller

An SDRAM device requires that each of its rows be refreshed at a minimum required rate. The EMIFA can meet this constraint by performing auto refresh cycles at or above this required rate. An auto refresh cycle consists of issuing a PRE command to all banks of the SDRAM device followed by issuing a REFR command. To inform the EMIFA of the required rate for performing auto refresh cycles, the RR field of the SDRAM refresh control register (SDRCR) must be programmed. The EMIFA will use this value along with two internal counters to automatically perform auto refresh cycles at the required rate. The auto refresh cycles cannot be disabled, even if the EMIFA is not interfaced with an SDRAM. The remainder of this section details the EMIFA's refresh scheme and provides an example for determining the appropriate value to place in the RR field of SDRCR.

The two counters used to perform auto-refresh cycles are a 13-bit refresh interval counter and a 4-bit refresh backlog counter. At reset and upon writing to the RR field, the refresh interval counter is loaded with the value from RR field and begins decrementing, by one, each EMIFA clock cycle. When the refresh interval counter reaches zero, the following actions occur:

- The refresh interval counter is reloaded with the value from the RR field and restarts decrementing.
- The 4-bit refresh backlog counter increments unless it has already reached its maximum value.

The refresh backlog counter records the number of auto refresh cycles that the EMIFA currently has outstanding. This counter is decremented by one each time an auto refresh cycle is performed and incremented by one each time the refresh interval counter expires. The refresh backlog counter saturates at the values of 0000b and 1111b. The EMIFA uses the refresh backlog counter to determine the urgency with which an auto refresh cycle should be performed. The four levels of urgency are described in Table 18-12. This refresh scheme allows the required refreshes to be performed with minimal impact on access requests.

Table 18-12. Refresh Urgency Levels

Urgency Level	Refresh Backlog Counter Range	Action Taken
Refresh May	1-3	An auto-refresh cycle is performed only if the EMIFA has no requests pending and none of the SDRAM banks are open.
Refresh Release	4-7	An auto-refresh cycle is performed if the EMIFA has no requests pending, regardless of whether any SDRAM banks are open.
Refresh Need	8-11	An auto-refresh cycle is performed at the completion of the current access unless there are read requests pending.
Refresh Must	12-15	Multiple auto-refresh cycles are performed at the completion of the current access until the Refresh Release urgency level is reached. At that point, the EMIFA can begin servicing any new read or write requests.

Copyright © 2013–2016, Texas Instruments Incorporated

18.2.4.6.1 Determining the Appropriate Value for the RR Field

The value that should be programmed into the RR field of SDRCR can be calculated by using the frequency of the EMA_CLK signal (f_{EMA_CLK}) and the required refresh rate of the SDRAM ($f_{Refresh}$). The following formula can be used:

$RR = f_{EMA_CLK} / f_{Refresh}$

The SDRAM datasheet often communicates the required SDRAM Refresh Rate in terms of the number of REFR commands required in a given time interval. The required SDRAM Refresh Rate in the formula above can therefore be calculated by dividing the number of required cycles per time interval (n_{cycles}) by the time interval given in the datasheet ($t_{Refresh\ Period}$) :

$f_{Refresh} = n_{cycles} / t_{Refresh\ Period}$

Combining these formulas, the value that should be programmed into the RR field can be computed as:

$RR = f_{EMA_CLK} \times t_{Refresh\ Period} / n_{cycles}$

The following example illustrates calculating the value of RR. Given that:

- f_{EMA_CLK} = 100 MHz (frequency of the EMIFA clock)
- $t_{Refresh\ Period}$ = 64 ms (required refresh interval of the SDRAM)
- n_{cycles} = 8192 (number of cycles in a refresh interval for the SDRAM)

RR can be calculated as:

RR = 100 MHz × 64 ms/8192

RR = 781.25

RR = 782 cycles = 30Eh cycles

18.2.4.7 Self-Refresh Mode

The EMIFA can be programmed to enter the self-refresh state by setting the SR bit of SDCR to 1. This will cause the EMIFA to issue the SLFR command after completing any outstanding SDRAM access requests and clearing the refresh backlog counter by performing one or more auto refresh cycles. This places the attached SDRAM device into self-refresh mode in which it consumes a minimal amount of power while performing its own refresh cycles. The SR bit should be set and cleared using a byte-write to the upper byte of the SDRAM configuration register (SDCR) to avoid triggering the SDRAM initialization sequence.

While in the self-refresh state, the EMIFA continues to service asynchronous bank requests and register accesses as normal, with one caveat. The EMIFA will not park the data bus following a read to asynchronous memory while in the self-refresh state. Instead, the EMIFA tri-states the data bus. Therefore, it is not recommended to perform asynchronous read operations while the EMIFA is in the self-refresh state, in order to prevent floating inputs on the data bus. More information about data bus parking can be found in Section 18.2.6.

The EMIFA will exit from the self-refresh state if either of the following events occur:

- The SR bit of SDCR is cleared to 0.
- An SDRAM accesses is requested.

The EMIFA exits from the self-refresh state by driving EMA_SDCKE high and performing an auto refresh cycle.

The attached SDRAM device should also be placed into Self-Refresh Mode when changing the frequency of EMA_CLK using the PLL Controller. If the frequency of EMA_CLK changes while the SDRAM is not in Self-Refresh Mode, Procedure B in Section 18.2.4.5 should be followed to reinitialize the device.

Copyright © 2013–2016, Texas Instruments Incorporated

18.2.4.8 Power Down Mode

To support low-power modes, the EMIFA can be requested to issue a POWER DOWN command to the SDRAM by setting the PD bit in the SDRAM configuration register (SDCR). When this bit is set, the EMIFA will continue normal operation until all outstanding memory access requests have been serviced and the SDRAM refresh backlog (if there is one) has been cleared. At this point the EMIFA will enter the power-down state. Upon entering this state, the EMIFA will issue a POWER DOWN command (same as a NOP command but driving EMA_SDCKE low on the same cycle). The EMIFA then maintains EMA_SDCKE low until it exits the power-down state.

Since the EMIFA services the refresh backlog before it enters the power-down state, all internal banks of the SDRAM are closed (precharged) prior to issuing the POWER DOWN command. Therefore, the EMIFA only supports Precharge Power Down. The EMIFA does not support Active Power Down, where internal banks of the SDRAM are open (active) before the POWER DOWN command is issued.

During the power-down state, the EMIFA services the SDRAM, asynchronous memory, and register accesses as normal, returning to the power-down state upon completion.

The PDWR bit in SDCR indicates whether the EMIFA should perform refreshes in power-down state. If the PDWR bit is set, the EMIFA exits the power-down state every time the Refresh Must level is set, performs AUTO REFRESH commands to the SDRAM, and returns back to the power-down state. This evenly distributes the refreshes to the SDRAM in power-down state. If the PDWR bit is not set, the EMIFA does not perform any refreshes to the SDRAM. Therefore, the data integrity of the SDRAM is not assured upon power down exit if the PDWR bit is not set.

If the PD bit is cleared while in the power-down state, the EMIFA will come out of the power-down state. The EMIFA:

- Drives EMA_SDCKE high.
- Enters its idle state.

Copyright © 2013–2016, Texas Instruments Incorporated

18.2.4.9 SDRAM Read Operation

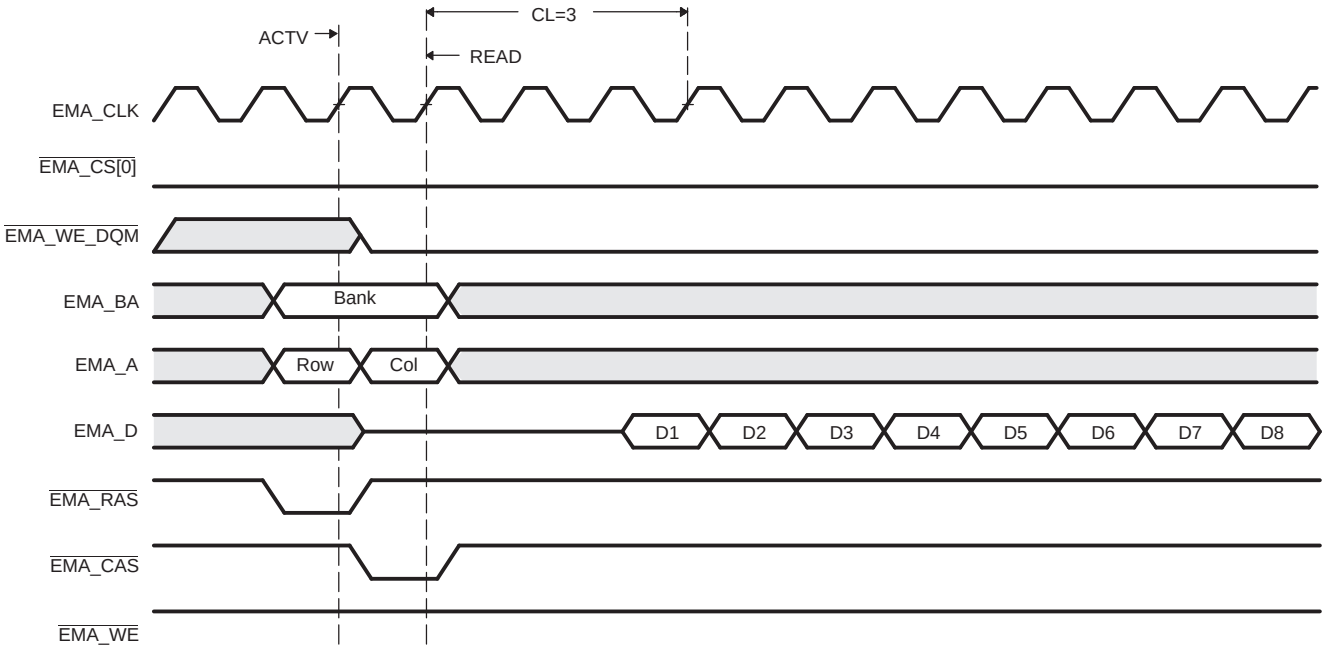
When the EMIFA receives a read request to SDRAM from one of the requesters listed in Section 18.2.2, it performs one or more read access cycles. A read access cycle begins with the issuing of the ACTV command to select the desired bank and row of the SDRAM device. After the row has been opened, the EMIFA proceeds to issue a READ command while specifying the desired bank and column address. EMA_A[10] is held low during the READ command to avoid auto-precharging. The READ command signals the SDRAM device to start bursting data from the specified address while the EMIFA issues NOP commands. Following a READ command, the CL field of the SDRAM configuration register (SDCR) defines how many delay cycles will be present before the read data appears on the data bus. This is referred to as the CAS latency.

Figure 18-5 shows the signal waveforms for a basic SDRAM read operation in which a burst of data is read from a single page. When the EMIFA SDRAM interface is configured to 16 bit by setting the NM bit of the SDRAM configuration register (SDCR) to 1, a burst size of eight is used. Figure 18-5 shows a burst size of eight.

The EMIFA will truncate a series of bursting data if the remaining addresses of the burst are not required to complete the request. The EMIFA can truncate the burst in three ways:

- By issuing another READ to the same page in the same bank.
- By issuing a PRE command in order to prepare for accessing a different page of the same bank.
- By issuing a BT command in order to prepare for accessing a page in a different bank.

Figure 18-5. Timing Waveform for Basic SDRAM Read Operation



Several other pins are also active during a read access. The EMA_WE_DQM[1:0] pins are driven low during the READ commands and are kept low during the NOP commands that correspond to the burst request. The state of the other EMIFA pins during each command can be found in Table 18-5.

The EMIFA schedules its commands based on the timing information that is provided to it in the SDRAM timing register (SDTIMR). The values for the timing parameters in this register should be chosen to satisfy the timing requirements listed in the SDRAM datasheet. The EMIFA uses this timing information to avoid violating any timing constraints related to issuing commands. This is commonly accomplished by inserting NOP commands between various commands during an access. Refer to the register description of SDTIMR in Section 18.4.6 for more details on the various timing parameters.

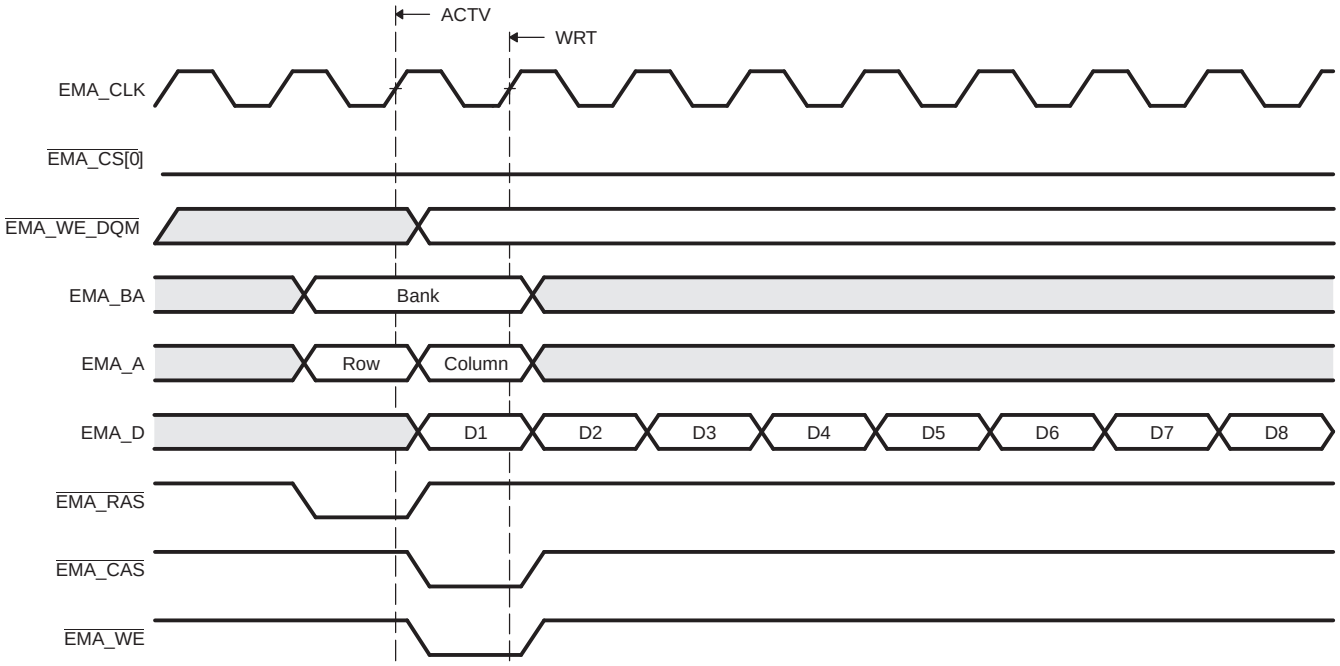
Copyright © 2013–2016, Texas Instruments Incorporated

18.2.4.10 SDRAM Write Operations

When the EMIFA receives a write request to SDRAM from one of the requesters listed in Section 18.2.2, it performs one or more write-access cycles. A write-access cycle begins with the issuing of the ACTV command to select the desired bank and row of the SDRAM device. After the row has been opened, the EMIFA proceeds to issue a WRT command while specifying the desired bank and column address. EMA_A[10] is held low during the WRT command to avoid auto-precharging. The WRT command signals the SDRAM device to start writing a burst of data to the specified address while the EMIFA issues NOP commands. The associated write data will be placed on the data bus in the cycle concurrent with the WRT command and with subsequent burst continuation NOP commands.

Figure 18-6 shows the signal waveforms for a basic SDRAM write operation in which a burst of data is read from a single page. When the EMIFA SDRAM interface is configured to 16-bit by setting the NM bit of the SDRAM configuration register (SDCR) to 1, a burst size of eight is used. Figure 18-6 shows a burst size of eight.

Figure 18-6. Timing Waveform for Basic SDRAM Write Operation



The EMIFA will truncate a series of bursting data if the remaining addresses of the burst are not part of the write request. The EMIFA can truncate the burst in three ways:

- By issuing another WRT to the same page
- By issuing a PRE command in order to prepare for accessing a different page of the same bank
- By issuing a BT command in order to prepare for accessing a page in a different bank

Several other pins are also active during a write access. The EMA_WE_DQM[1:0] pins are driven to select which bytes of the data word will be written to the SDRAM device. They are also used to mask out entire undesired data words during a burst access. The state of the other EMIFA pins during each command can be found in Table 18-5.

The EMIFA schedules its commands based on the timing information that is provided to it in the SDRAM timing register (SDTIMR). The values for the timing parameters in this register should be chosen to satisfy the timing requirements listed in the SDRAM datasheet. The EMIFA uses this timing information to avoid violating any timing constraints related to issuing commands. This is commonly accomplished by inserting NOP commands during various cycles of an access. Refer to the register description of SDTIMR in Section 18.4.6 for more details on the various timing parameters.

Copyright © 2013–2016, Texas Instruments Incorporated

18.2.4.11 Mapping from Logical Address to EMIFA Pins

When the EMIFA receives an SDRAM access request, it must convert the address of the access into the appropriate signals to send to the SDRAM device. The details of this address mapping are shown in Table 18-13 for 16-bit operation. Using the settings of the IBANK and PAGESIZE fields of the SDRAM configuration register (SDCR), the EMIFA determines which bits of the logical address are mapped to the SDRAM row, column, and bank addresses.

As the logical address is incremented by one halfword (16-bit operation), the column address is likewise incremented by one until a page boundary is reached. When the logical address increments across a page boundary, the EMIFA moves into the same page in the next bank of the attached device by incrementing the bank address EMA_BA and resetting the column address. The page in the previous bank is left open until it is necessary to close it. This method of traversal through the SDRAM banks helps maximize the number of open banks inside of the SDRAM and results in an efficient use of the device. There is no limitation on the number of banks that can be open at one time, but only one page within a bank can be open at a time.

The EMIFA uses the $\overline{\text{EMA_WE_DQM}}$ pins during a WRT command to mask out selected bytes or entire words. The $\overline{\text{EMA_WE_DQM}}$ pins are always low during a READ command.

Table 18-13. Mapping from Logical Address to EMIFA Pins for 16-bit SDRAM

		Logical Address													
IBANK	PAGESIZE	31:27	26	25	24	23	22	21:14	13	12	11	10	9	8:1	0
0	0	-						Row Address						Col Address	$\overline{\text{EMA_WE_DQM}}$[0]
1	0	-						Row Address					EMA_BA[0]	Col Address	$\overline{\text{EMA_WE_DQM}}$[0]
2	0	-						Row Address				EMA_BA[1:0]		Col Address	$\overline{\text{EMA_WE_DQM}}$[0]
0	1	-						Row Address					Column Address		$\overline{\text{EMA_WE_DQM}}$[0]
1	1	-						Row Address				EMA_BA[0]	Column Address		$\overline{\text{EMA_WE_DQM}}$[0]
2	1	-						Row Address			EMA_BA[1:0]		Column Address		$\overline{\text{EMA_WE_DQM}}$[0]
0	2	-						Row Address				Column Address			$\overline{\text{EMA_WE_DQM}}$[0]
1	2	-						Row Address			EMA_BA[0]	Column Address			$\overline{\text{EMA_WE_DQM}}$[0]
2	2	-						Row Address		EMA_BA[1:0]		Column Address			$\overline{\text{EMA_WE_DQM}}$[0]
0	3	-						Row Address			Column Address				$\overline{\text{EMA_WE_DQM}}$[0]
1	3	-						Row Address		EMA_BA[0]	Column Address				$\overline{\text{EMA_WE_DQM}}$[0]
2	3	-						Row Address		EMA_BA[1:0]	Column Address				$\overline{\text{EMA_WE_DQM}}$[0]

NOTE: The upper bit of the Row Address is used only when addressing 256-Mbit and 512-Mbit SDRAM memories.

Copyright © 2013–2016, Texas Instruments Incorporated

18.2.5 Asynchronous Controller and Interface

The EMIFA easily interfaces to a variety of asynchronous devices including NOR Flash, NAND Flash, and SRAM. It can be operated in two major modes (see Table 18-14):

- Normal Mode
- Select Strobe Mode

Table 18-14. Normal Mode vs. Select Strobe Mode

Mode	Function of $\overline{\text{EMA_WE_DQM}}$ pins	Operation of $\overline{\text{EMA_CS[5:2]}}$
Normal Mode	Byte enables	Active during the entire asynchronous access cycle
Select Strobe Mode	Byte enables	Active only during the strobe period of an access cycle

The first mode of operation is Normal Mode, in which the $\overline{\text{EMA_WE_DQM}}$ pins of the EMIFA function as byte enables. In this mode, the $\overline{\text{EMA_CS[5:2]}}$ pins behaves as typical chip select signals, remaining active for the duration of the asynchronous access. See Section 18.2.5.1 for an example interface with multiple 8-bit devices.

The second mode of operation is Select Strobe Mode, in which the $\overline{\text{EMA_CS[5:2]}}$ pins act as a strobe, active only during the strobe period of an access. In this mode, the $\overline{\text{EMA_WE_DQM}}$ pins of the EMIFA function as standard byte enables for reads and writes. A summary of the differences between the two modes of operation are shown in Table 18-14. Refer to Section 18.2.5.4 for the details of asynchronous operations in Normal Mode, and to Section 18.2.5.5 for the details of asynchronous operations in Select Strobe Mode. The EMIFA hardware defaults to Normal Mode, but can be manually switched to Select Strobe Mode by setting the SS bit in the asynchronous *m* (m = 1, 2, 3, or 4) configuration register (CE*n*CFG) (*n* = 2, 3, 4, or 5). Throughout the chapter, *m* can hold the values 1, 2, 3 or 4; and *n* can hold the values 2, 3, 4, or 5.

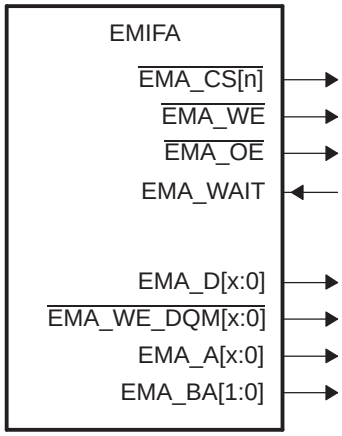
In both Normal Mode and Select Strobe Mode, the EMIFA can be configured to operate in a sub-mode called NAND Flash Mode. In NAND Flash Mode, the EMIFA is able to calculate an error correction code (ECC) for transfers up to 518 bytes.

The EMIFA also provides configurable cycle timing parameters and an Extended Wait Mode that allows the connected device to extend the strobe period of an access cycle. The following sections describe the features related to interfacing with external asynchronous devices.

18.2.5.1 Interfacing to Asynchronous Memory

Figure 18-7 shows the EMIFA's external pins used in interfacing with an asynchronous device. In $\overline{\text{EMA_CS[n]}}$, n = 2, 3, 4, or 5.

Figure 18-7. EMIFA Asynchronous Interface

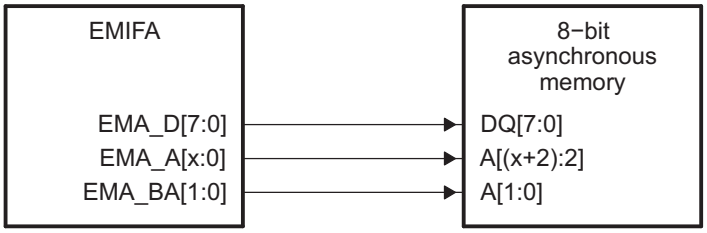


Copyright © 2013–2016, Texas Instruments Incorporated

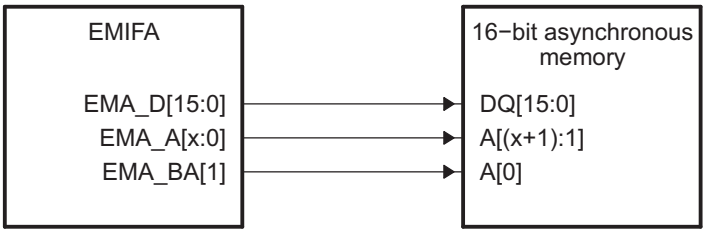
Of special note is the connection between the EMIFA and the external device's address bus. The EMIFA address pin EMA_A[0] always provides the least significant bit of a 32-bit word address. Therefore, when interfacing to a 16-bit or 8-bit asynchronous device, the EMA_BA[1] and EMA_BA[0] pins provide the least-significant bits of the halfword or byte address, respectively. Additionally, when the EMIFA interfaces to a 16-bit asynchronous device, the EMA_BA[0] pin can serve as the upper address line EMA_A[22]. Note that the width of the address bus varies with devices; therefore, see your device-specific data manual for the EMA_A bus width supported. Figure 18-8 and Figure 18-9 show the mapping between the EMIFA and the connected device's data and address pins for various programmed data bus widths. The data bus width may be configured in the asynchronous *n* configuration register (CE*n*CFG).

Figure 18-9 shows a common interface between the EMIFA and external asynchronous memory. Figure 18-9 shows an interface between the EMIFA and an external memory with byte enables. The EMIFA should be operated in either Normal Mode or Select Strobe Mode when using this interface, so that the $\overline{\text{EMA_WE_DQM}}$ signals operate as byte enables.

Figure 18-8. EMIFA to 8-bit/16-bit Memory Interface

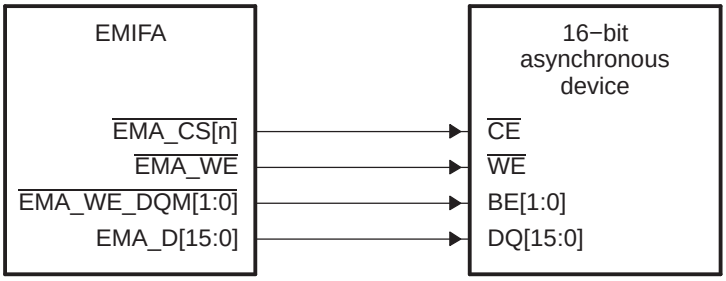


a) EMIF to 8-bit memory interface



b) EMIF to 16-bit memory interface

Figure 18-9. Common Asynchronous Interface



Copyright © 2013–2016, Texas Instruments Incorporated

18.2.5.2 Accessing Larger Asynchronous Memories

The device has a limited number of dedicated EMIFA address pins, enough to interface directly to an SDRAM. If a device such as an asynchronous flash needs to be attached to the EMIFA, then GPIO pins may be used to control the flash device's upper address lines. This is sufficient to boot from the flash. Normally, code stored in flash is copied into SDRAM or internal memory before executing because these memories have much faster access times. For details on which device pins are GPIO capable, see your device-specific data manual.

The ROM bootloader can load a secondary bootloader from an attached asynchronous device. The ROM bootloader assumes that any GPIO pins used to control the upper address lines of the boot flash will be pulled to 0 after reset. This means that normally the GPIO pins selected for this function will be either spare or used as outputs only by the application, and therefore can be pulled to 0 at reset with an external pulldown resistor. The GPIO pins chosen should be tri-stated by default on device reset. For details on which GPIO-capable pins are tri-stated on device reset, see your device-specific data manual.

When booting from flash, the ROM bootloader copies a board-specific secondary bootloader from the lower portion of the flash, so it does not need to manipulate the upper address lines. Only the secondary bootloader, which is board-specific and is stored in the external flash, needs to know which GPIO pins have been assigned to the function of upper address lines. Therefore, the secondary bootloader can perform the task of configuring the selected pins as GPIO and loading the remainder of the code from the upper flash memory.

18.2.5.3 Configuring the EMIFA for Asynchronous Accesses

The operation of the EMIFA's asynchronous interface can be configured by programming the appropriate register fields. The reset value and bit position for each register field can be found in Section 18.4, but the Boot ROM documentation should be consulted to determine if the fields are programmed during boot. The following tables list the register fields that can be programmed and describe the purpose of each field. These registers can be programmed prior to accessing the external memory, and the transfer following a write to these registers will use the new configuration.

Table 18-15. Description of the Asynchronous *m* Configuration Register (CE*n*CFG)

Parameter	Description
SS	**Select Strobe mode.** This bit selects the EMIFA's mode of operation in the following way: • SS = 0 selects Normal Mode – $\overline{\text{EMA_WE_DQM}}$ pins function as byte enables – $\overline{\text{EMA_CS}}$[5:2] active for duration of access • SS = 1 selects Select Strobe Mode – $\overline{\text{EMA_WE_DQM}}$ pins function as byte enables – $\overline{\text{EMA_CS}}$[5:2] acts as a strobe.
EW	**Extended Wait Mode enable.** • EW = 0 disables Extended Wait Mode • EW = 1 enables Extended Wait Mode When set to 1, the EMIFA enables its Extended Wait Mode in which the strobe width of an access cycle can be extended in response to the assertion of the EMA_WAIT pin(1). The WP*n* bit in the asynchronous wait cycle configuration register (AWCC) controls to polarity of EMA_WAIT pin. Extended Wait Mode should not be used while in NAND Flash Mode. See Section 18.2.5.7 for more details on this mode of operation.
W_SETUP/R_SETUP	**Read/Write setup widths.** These fields define the number (n) of EMIFA clock cycles of setup time for the address pins (EMA_A and EMA_BA), byte enables ($\overline{\text{EMA_WE_DQM}}$), and asynchronous chip enable ($\overline{\text{EMA_CS}}$[5:2]) before the read strobe pin ($\overline{\text{EMA_OE}}$) or write strobe pin ($\overline{\text{EMA_WE}}$) falls. This value should be encoded as n - 1, where n is the number of EMIFA clock cycles. For example, when W_SETUP = 2, then write setup width = 3 EMA_CLK cycles. For writes, the W_SETUP field also defines the setup time for the data pins (EMA_D). Refer to the datasheet of the external asynchronous device to determine the appropriate setting for this field.

(1) The EMA_WAIT pin is not available on all devices; therefore, this field is reserved on those devices.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 18-15. Description of the Asynchronous *m* Configuration Register (CE*n*CFG) (continued)

Parameter	Description
W_STROBE/R_STROBE	**Read/Write strobe widths.** These fields define the number (n) of EMIFA clock cycles between the falling and rising edge of the read strobe pin (EMA_OE) or write strobe pin (EMA_WE). This value should be encoded as n - 1, where n is the number of EMIFA clock cycles. For example, when W_SETUP = 2, then write setup width = 3 EMA_CLK cycles. If Extended Wait Mode is enabled by setting the EW field in the asynchronous *n* configuration register (CE*n*CFG), these fields must be set to a value greater than zero. Refer to the datasheet of the external asynchronous device to determine the appropriate setting for this field.
W_HOLD/R_HOLD	**Read/Write hold widths.**[2] These fields define the number (n) of EMIFA clock cycles of hold time for the address pins (EMA_A and EMA_BA), byte enables (EMA_WE_DQM), and asynchronous chip enable (EMA_CS[5:2]) after the read strobe pin (EMA_OE) or write strobe pin (EMA_WE) rises. This value should be encoded as n - 1, where n is the number of EMIFA clock cycles. For example, when W_SETUP = 2, then write setup width = 3 EMA_CLK cycles.For writes, the W_HOLD field also defines the hold time for the data pins (EMA_D). Refer to the datasheet of the external asynchronous device to determine the appropriate setting for this field.
TA	**Minimum turnaround time.** This field defines the minimum number of EMIFA clock cycles between asynchronous reads and writes, minus one cycle. The purpose of this feature is to avoid contention on the bus. The value written to this field also determines the number of cycles that will be inserted between asynchronous accesses and SDRAM accesses. Refer to the datasheet of the external asynchronous device to determine the appropriate setting for this field. If more turnaround cycles are required than can be programmed into the TA field, additional cycles can be added to the R_HOLD field to compensate.
ASIZE	**Asynchronous Device Bus Width.** This field determines the data bus width of the asynchronous interface in the following way: • ASIZE = 0 selects an 8-bit bus • ASIZE = 1 selects a 16-bit bus The configuration of ASIZE determines the function of the EMA_A and EMA_BA pins as described in Section 18.2.5.1. This field also determines the number of external accesses required to fulfill a request generated by one of the sources mentioned in Section 18.2.2. For example, a request for a 32-bit word would require four external access when ASIZE = 0. Refer to the datasheet of the external asynchronous device to determine the appropriate setting for this field.

(2) When using a 16 bit NAND device with ECC calculation support enabled via the EMIFA controller , the EMIFA ECC engine requires the read data to be stable for 2 clock cycles to calculate ECC. The ECC latches 8 bits of data at a time, therefore it requiring additional 2 clock cycles to complete the calculation for 16-bit data. For reliable 16-Bit NAND operations R_HOLD must be set to 1 to ensure 2 clock cycles for ECC calculation.

Table 18-16. Description of the Asynchronous Wait Cycle Configuration Register (AWCC)[1]

Parameter	Description
WP*n*	**EMA_WAIT Polarity.** • WP*n* = 0 selects active-low polarity • WP*n* = 1 selects active-high polarity When set to 1, the EMIFA will wait if the EMA_WAIT pin is high. When cleared to 0, the EMIFA will wait if the EMA_WAIT pin is low. The EMIFA must have the Extended Wait Mode enabled for the EMA_WAIT pin to affect the width of the strobe period. The polarity of the EMA_WAIT signal is not programmable in NAND Flash Mode.

(1) The EMA_WAIT pin is not available on all devices; therefore, this register is reserved on those devices.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 18-16. Description of the Asynchronous Wait Cycle Configuration Register (AWCC)[(1)] (continued)

Parameter	Description
MAX_EXT_WAIT	**Maximum Extended Wait Cycles.** This field configures the number of EMIFA clock cycles the EMIFA will wait for the EMA_WAIT pin to be deactivated during the strobe period of an access cycle. The maximum number of EMIFA clock cycles it will wait is determined by the following formula: Maximum Extended Wait Cycles = (MAX_EXT_WAIT + 1) × 16 If the EMA_WAIT pin is not deactivated within the time specified by this field, the EMIFA resumes the access cycle, registering whatever data is on the bus and proceeding to the hold period of the access cycle. This situation is referred to as an Asynchronous Timeout. An Asynchronous Timeout generates an interrupt, if it has been enabled in the EMIFA interrupt mask set register (INTMSKSET). Refer to Section 18.2.8.1 for more information about the EMIFA interrupts. Extended Wait Mode should not be used while in NAND Flash Mode.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 18-17. Description of the EMIFA Interrupt Mask Set Register (INTMSKSET)

Parameter	Description
WR_MASK_SET	**Wait Rise Mask Set.** Writing a 1 enables an interrupt to be generated when a rising edge on EMA_WAIT[1] occurs while in NAND Flash Mode
AT_MASK_SET	**Asynchronous Timeout Mask Set.** Writing a 1 to this bit enables an interrupt to be generated when an Asynchronous Timeout occurs.

(1) The EMA_WAIT pin is not available on all devices; therefore, this field is reserved on those devices.

Table 18-18. Description of the EMIFA Interrupt Mast Clear Register (INTMSKCLR)

Parameter	Description
WR_MASK_CLR	**Wait Rise Mask Clear.** Writing a 1 to this bit disables the interrupt, clearing the WR_MASK_SET bit in the EMIFA interrupt mask set register (INTMSKSET).
AT_MASK_CLR	**Asynchronous Timeout Mask Clear.** Writing a 1 to this bit prevents an interrupt from being generated when an Asynchronous Timeout occurs.

18.2.5.4 Read and Write Operations in Normal Mode

Normal Mode is the asynchronous interface's default mode of operation. It is selected when the SS bit in the asynchronous *n* configuration register (CE*n*CFG) is cleared to 0. In this mode, the $\overline{\text{EMA_WE_DQM}}$ pins operate as byte enables. Section 18.2.5.4.1 and Section 18.2.5.4.2 explain the details of read and write operations while in Normal Mode.

18.2.5.4.1 Asynchronous Read Operations (Normal Mode)

NOTE: During the entirety of an asynchronous read operation, the $\overline{\text{EMA_WE}}$ pin is driven high.

An asynchronous read is performed when any of the requesters mentioned in Section 18.2.2 request a read from the attached asynchronous memory. After the request is received, a read operation is initiated once it becomes the EMIFA's highest priority task, according to the priority scheme detailed in Section 18.2.12. In the event that the read request cannot be serviced by a single access cycle to the external device, multiple access cycles will be performed by the EMIFA until the entire request is fulfilled. The details of an asynchronous read operation in Normal Mode are described in Table 18-19. Also, Figure 18-10 shows an example timing diagram of a basic read operation.

Table 18-19. Asynchronous Read Operation in Normal Mode

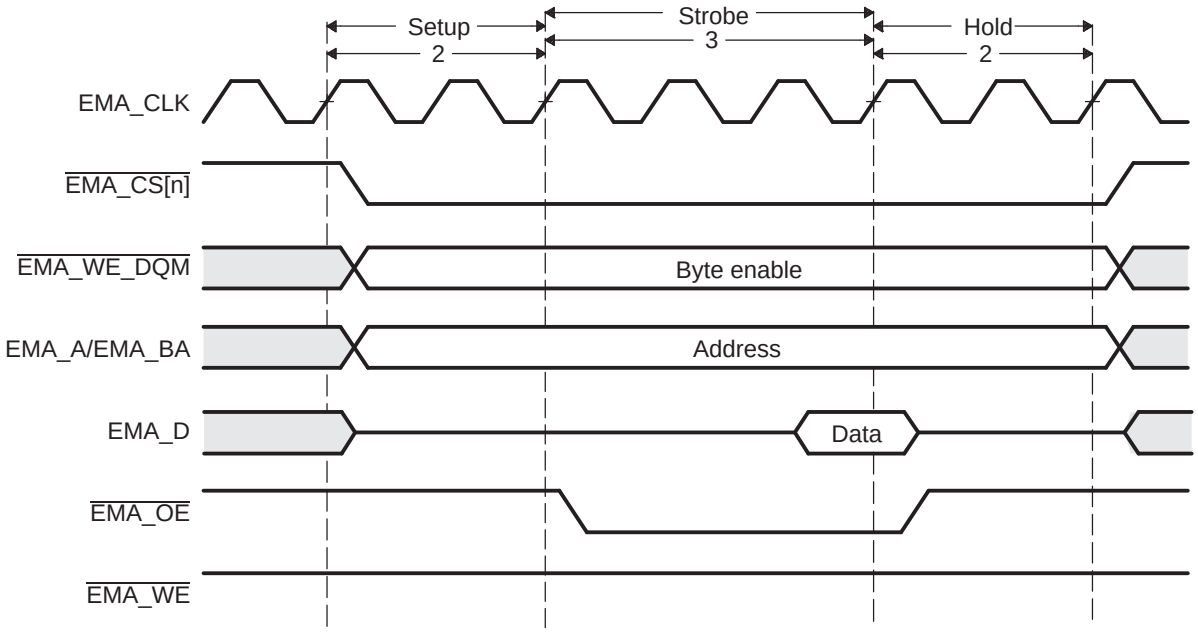
Time Interval	Pin Activity in Normal Mode
Turnaround period	Once the read operation becomes the highest priority task for the EMIFA, the EMIFA waits for the programmed number of turn-around cycles before proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the asynchronous *n* configuration register (CE*n*CFG). There are two exceptions to this rule: • If the current read operation was directly proceeded by another read operation to the same chip select, no turnaround cycles are inserted. After the EMIFA has waited for the turnaround cycles to complete, it again checks to make sure that the read operation is still its highest priority task. If so, the EMIFA proceeds to the setup period of the operation. If it is no longer the highest priority task, the EMIFA terminates the operation.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 18-19. Asynchronous Read Operation in Normal Mode (continued)

Time Interval	Pin Activity in Normal Mode
Start of the setup period	The following actions occur at the start of the setup period: • The setup, strobe, and hold values are set according to the R_SETUP, R_STROBE, and R_HOLD values in CE*n*CFG. • The address pins EMA_A and EMA_BA become valid and carry the values described in Section 18.2.5.1. • $\overline{\text{EMA_CS}}$[5:2] falls to enable the external device (if not already low from a previous operation)
Strobe period	The following actions occur during the strobe period of a read operation: 1. $\overline{\text{EMA_OE}}$ falls at the start of the strobe period 2. On the rising edge of the clock which is concurrent with the end of the strobe period: • $\overline{\text{EMA_OE}}$ rises • The data on the EMA_D bus is sampled by the EMIFA. In Figure 18-10, EMA_WAIT is inactive. If EMA_WAIT is instead activated, the strobe period can be extended by the external device to give it more time to provide the data. Section 18.2.5.7 contains more details on using the EMA_WAIT pin.
End of the hold period	At the end of the hold period: • The address pins EMA_A and EMA_BA become invalid • $\overline{\text{EMA_CS}}$[5:2] rises (if no more operations are required to complete the current request) EMIFA may be required to issue additional read operations to a device with a small data bus width in order to complete an entire word access. In this case, the EMIFA immediately re-enters the setup period to begin another operation without incurring the turn-round cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIFA returns to its previous state unless another asynchronous request has been submitted and is currently the highest priority task. If this is the case, the EMIFA instead enters directly into the turnaround period for the pending read or write operation.

Figure 18-10. Timing Waveform of an Asynchronous Read Cycle in Normal Mode



Copyright © 2013–2016, Texas Instruments Incorporated

18.2.5.4.2 Asynchronous Write Operations (Normal Mode)

NOTE: During the entirety of an asynchronous write operation, the EMA_OE pin is driven high.

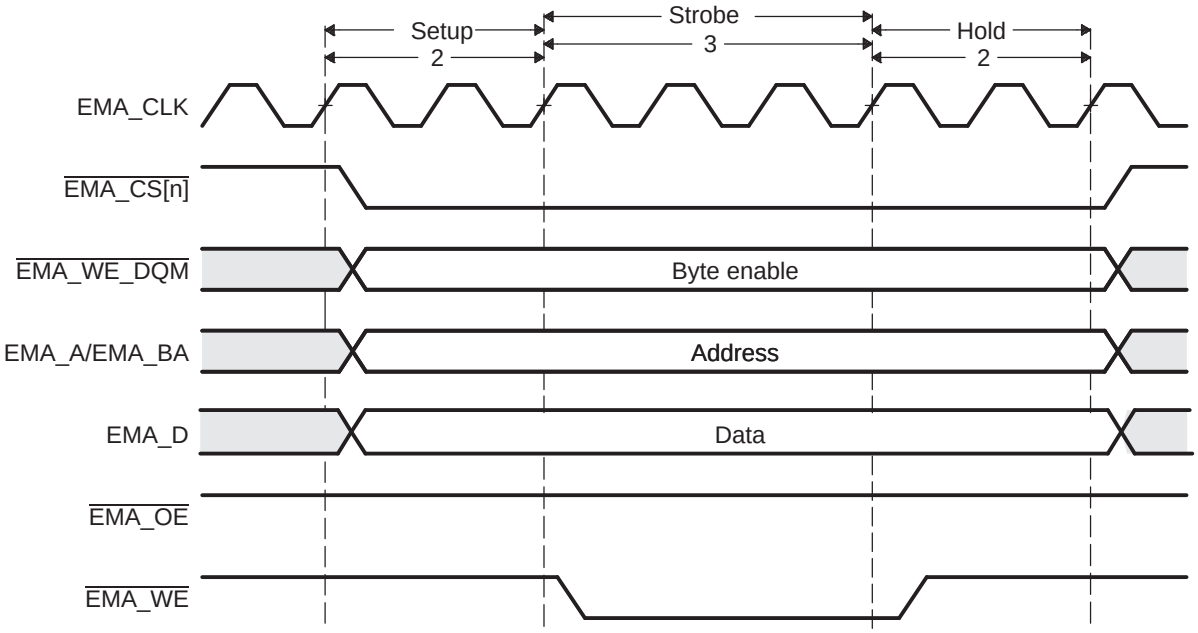
An asynchronous write is performed when any of the requesters mentioned in Section 18.2.2 request a write to memory in the asynchronous bank of the EMIFA. After the request is received, a write operation is initiated once it becomes the EMIFA's highest priority task, according to the priority scheme detailed in Section 18.2.12. In the event that the write request cannot be serviced by a single access cycle to the external device, multiple access cycles will be performed by the EMIFA until the entire request is fulfilled. The details of an asynchronous write operation in Normal Mode are described in Table 18-20. Also, Figure 18-11 shows an example timing diagram of a basic write operation.

Table 18-20. Asynchronous Write Operation in Normal Mode

Time Interval	Pin Activity in Normal Mode
Turnaround period	Once the write operation becomes the highest priority task for the EMIFA, the EMIFA waits for the programmed number of turn-around cycles before proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the asynchronous *n* configuration register (CE*n*CFG). There are two exceptions to this rule: • If the current write operation was directly proceeded by another write operation to the same chip select, no turn-around cycles are inserted. After the EMIFA has waited for the turn-around cycles to complete, it again checks to make sure that the write operation is still its highest priority task. If so, the EMIFA proceeds to the setup period of the operation. If it is no longer the highest priority task, the EMIFA terminates the operation.
Start of the setup period	The following actions occur at the start of the setup period: • The setup, strobe, and hold values are set according to the W_SETUP, W_STROBE, and W_HOLD values in CE*n*CFG. • The address pins EMA_A and EMA_BA and the data pins EMA_D become valid. The EMA_A and EMA_BA pins carry the values described in Section 18.2.5.1. • EMA_CS[5:2] falls to enable the external device (if not already low from a previous operation).
Strobe period	The following actions occur at the start of the strobe period of a write operation: 1. EMA_WE falls 2. The EMA_WE_DQM pins become valid as byte enables. The following actions occur on the rising edge of the clock which is concurrent with the end of the strobe period: 1. EMA_WE rises 2. The EMA_WE_DQM pins deactivate In Figure 18-11, EMA_WAIT is inactive. If EMA_WAIT is instead activated, the strobe period can be extended by the external device to give it more time to accept the data. Section 18.2.5.7 contains more details on using the EMA_WAIT pin.
End of the hold period	At the end of the hold period: • The address pins EMA_A and EMA_BA become invalid • The data pins become invalid • EMA_CS[n] (n = 2, 3, 4, or 5) rises (if no more operations are required to complete the current request) The EMIFA may be required to issue additional write operations to a device with a small data bus width in order to complete an entire word access. In this case, the EMIFA immediately re-enters the setup period to begin another operation without incurring the turnaround cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIFA returns to its previous state unless another asynchronous request has been submitted and is currently the highest priority task. If this is the case, the EMIFA instead enters directly into the turnaround period for the pending read or write operation.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 18-11. Timing Waveform of an Asynchronous Write Cycle in Normal Mode



Copyright © 2013–2016, Texas Instruments Incorporated

18.2.5.5 Read and Write Operation in Select Strobe Mode

Select Strobe Mode is the EMIFA's second mode of operation. It is selected when the SS bit of the asynchronous *n* configuration register (CE*n*CFG) is set to 1. In this mode, the EMA_WE_DQM pins operate as byte enables and the EMA_CS[n] (n = 2, 3, 4, or 5) pin is only active during the strobe period of an access cycle. Section 18.2.5.4.1 and Section 18.2.5.4.2 explain the details of read and write operations while in Select Strobe Mode.

18.2.5.5.1 Asynchronous Read Operations (Select Strobe Mode)

NOTE: During the entirety of an asynchronous read operation, the EMA_WE pin is driven high.

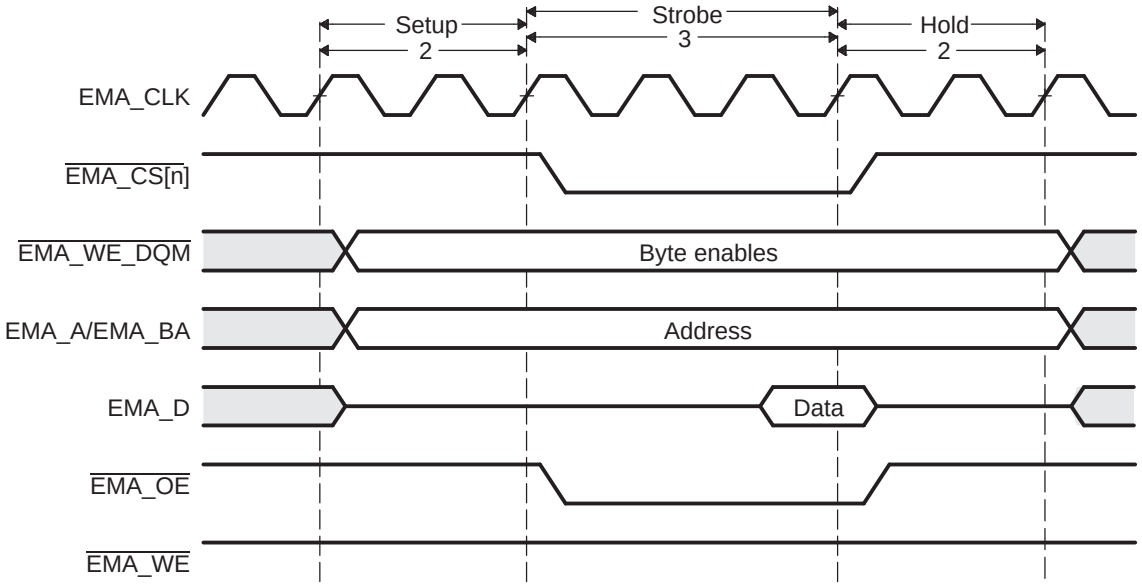
An asynchronous read is performed when any of the requesters mentioned in Section 18.2.2 request a read from the attached asynchronous memory. After the request is received, a read operation is initiated once it becomes the EMIFA's highest priority task, according to the priority scheme detailed in Section 18.2.12. In the event that the read request cannot be serviced by a single access cycle to the external device, multiple access cycles will be performed by the EMIFA until the entire request is fulfilled. The details of an asynchronous read operation in Select Strobe Mode are described in Table 18-21. Also, Figure 18-12 shows an example timing diagram of a basic read operation.

Table 18-21. Asynchronous Read Operation in Select Strobe Mode

Time Interval	Pin Activity in Select Strobe Mode
Turnaround period	Once the read operation becomes the highest priority task for the EMIFA, the EMIFA waits for the programmed number of turn-around cycles before proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the asynchronous *n* configuration register (CE*n*CFG). There are two exceptions to this rule: • If the current read operation was directly proceeded by another read operation to the same chip select, no turn-around cycles are inserted. After the EMIFA has waited for the turn-around cycles to complete, it again checks to make sure that the read operation is still its highest priority task. If so, the EMIFA proceeds to the setup period of the operation. If it is no longer the highest priority task, the EMIFA terminates the operation.
Start of the setup period	The following actions occur at the start of the setup period: • The setup, strobe, and hold values are set according to the R_SETUP, R_STROBE, and R_HOLD values in CE*n*CFG. • The address pins EMA_A and EMA_BA become valid and carry the values described in Section 18.2.5.1. • The EMA_WE_DQM pins become valid as byte enables.
Strobe period	The following actions occur during the strobe period of a read operation: 1. EMA_CS[n] (n = 2, 3, 4, or 5) and EMA_OE fall at the start of the strobe period 2. On the rising edge of the clock which is concurrent with the end of the strobe period: • EMA_CS[n] (n = 2, 3, 4, or 5) and EMA_OE rise • The data on the EMA_D bus is sampled by the EMIFA. In Figure 18-12, EMA_WAIT is inactive. If EMA_WAIT is instead activated, the strobe period can be extended by the external device to give it more time to provide the data. Section 18.2.5.7 contains more details on using the EMA_WAIT pin.
End of the hold period	At the end of the hold period: • The address pins EMA_A and EMA_BA become invalid • The EMA_EMA_WE_DQM pins become invalid The EMIFA may be required to issue additional read operations to a device with a small data bus width in order to complete an entire word access. In this case, the EMIFA immediately re-enters the setup period to begin another operation without incurring the turnaround cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIFA returns to its previous state unless another asynchronous request has been submitted and is currently the highest priority task. If this is the case, the EMIFA instead enters directly into the turnaround period for the pending read or write operation.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 18-12. Timing Waveform of an Asynchronous Read Cycle in Select Strobe Mode



Copyright © 2013–2016, Texas Instruments Incorporated

18.2.5.5.2 Asynchronous Write Operations (Select Strobe Mode)

NOTE: During the entirety of an asynchronous write operation, the $\overline{\text{EMA_OE}}$ pin is driven high.

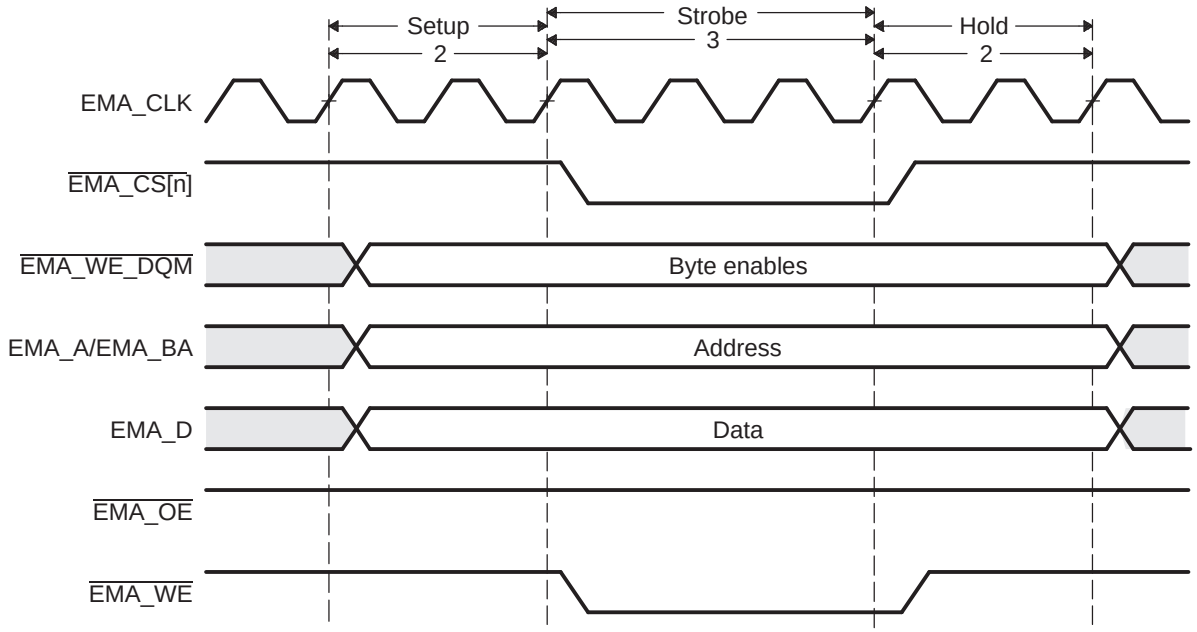
An asynchronous write is performed when any of the requesters mentioned in Section 18.2.2 request a write to memory in the asynchronous bank of the EMIFA. After the request is received, a write operation is initiated once it becomes the EMIFA's highest priority task, according to the priority scheme detailed in Section 18.2.12. In the event that the write request cannot be serviced by a single access cycle to the external device, multiple access cycles will be performed by the EMIFA until the entire request is fulfilled. The details of an asynchronous write operation in Select Strobe Mode are described in Table 18-22. Also, Figure 18-13 shows an example timing diagram of a basic write operation.

Table 18-22. Asynchronous Write Operation in Select Strobe Mode

Time Interval	Pin Activity in Select Strobe Mode
Turnaround period	Once the write operation becomes the highest priority task for the EMIFA, the EMIFA waits for the programmed number of turn-around cycles before proceeding to the setup period of the operation. The number of wait cycles is taken directly from the TA field of the asynchronous *n* configuration register (CE*n*CFG). There are two exceptions to this rule: • If the current write operation was directly proceeded by another write operation to the same chip select, no turn-around cycles are inserted. After the EMIFA has waited for the turnaround cycles to complete, it again checks to make sure that the write operation is still its highest priority task. If so, the EMIFA proceeds to the setup period of the operation. If it is no longer the highest priority task, the EMIFA terminates the operation.
Start of the setup period	The following actions occur at the start of the setup period: • The setup, strobe, and hold values are set according to the W_SETUP, W_STROBE, and W_HOLD values in CE*n*CFG. • The address pins EMA_A and EMA_BA and the data pins EMA_D become valid. The EMA_A and EMA_BA pins carry the values described in Section 18.2.5.1. • The $\overline{\text{EMA_WE_DQM}}$ pins become active as byte enables.
Strobe period	The following actions occur at the start of the strobe period of a write operation: • $\overline{\text{EMA_CS[n]}}$ (n = 2, 3, 4, or 5) and $\overline{\text{EMA_WE}}$ fall The following actions occur on the rising edge of the clock which is concurrent with the end of the strobe period: • $\overline{\text{EMA_CS[n]}}$ (n = 2, 3, 4, or 5) and $\overline{\text{EMA_WE}}$ rise In Figure 18-13, EMA_WAIT is inactive. If EMA_WAIT is instead activated, the strobe period can be extended by the external device to give it more time to accept the data. Section 18.2.5.7 contains more details on using the EMA_WAIT pin.
End of the hold period	At the end of the hold period: • The address pins EMA_A and EMA_BA become invalid • The data pins become invalid • The $\overline{\text{EMA_WE_DQM}}$ pins become invalid The EMIFA may be required to issue additional write operations to a device with a small data bus width in order to complete an entire word access. In this case, the EMIFA immediately re-enters the setup period to begin another operation without incurring the turnaround cycle delay. The setup, strobe, and hold values are not updated in this case. If the entire word access has been completed, the EMIFA returns to its previous state unless another asynchronous request has been submitted and is currently the highest priority task. If this is the case, the EMIFA instead enters directly into the turn-around period for the pending read or write operation.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 18-13. Timing Waveform of an Asynchronous Write Cycle in Select Strobe Mode



Copyright © 2013–2016, Texas Instruments Incorporated

18.2.5.6 NAND Flash Mode

NAND Flash Mode is a submode of both Normal Mode and Select Strobe Mode. Chip select $\overline{\text{EMA_CS}}$[n] (n = 2, 3, 4, or 5) may be placed in NAND Flash mode by setting the CS*n*NAND (*n* = 2, 3, 4, or 5) bit in the NAND Flash control register (NANDFCR). Table 18-23 displays the bit fields present in NANDFCR and briefly describes their use.

When a chip select space is configured to operate in NAND Flash mode, the EMIFA hardware can calculate the error correction code (ECC) for each 518 byte data transfer to that chip select space. The EMIFA hardware will not generate the NAND access cycle, which includes the command, address, and data phases, necessary to complete a transfer to NAND Flash. All NAND Flash operations can be divided into single asynchronous cycles, and with the help of software the EMIFA can execute a complete NAND access cycle.

Table 18-23. Description of the NAND Flash Control Register (NANDFCR)

Parameter	Description
CS5ECC	**NAND Flash ECC state for $\overline{\text{EMA_CS}}$[5].** • Set to 1 to start an ECC calculation for $\overline{\text{EMA_CS}}$[5] • Cleared to 0 when NAND Flash 4 ECC register (NANDF4ECC) is read.
CS5NAND	**NAND Flash mode for $\overline{\text{EMA_CS}}$[5].** • Set to 1 to enable NAND Flash mode for $\overline{\text{EMA_CS}}$[5]
CS4ECC	**NAND Flash ECC state for $\overline{\text{EMA_CS}}$[4].** • Set to 1 to start an ECC calculation for $\overline{\text{EMA_CS}}$[4] • Cleared to 0 when NAND Flash 3 ECC register (NANDF3ECC) is read.
CS4NAND	**NAND Flash mode for $\overline{\text{EMA_CS}}$[4].** • Set to 1 to enable NAND Flash mode for $\overline{\text{EMA_CS}}$[4]
CS3ECC	**NAND Flash ECC state for $\overline{\text{EMA_CS}}$[3].** • Set to 1 to start an ECC calculation for $\overline{\text{EMA_CS}}$[3] • Cleared to 0 when NAND Flash 2ECC register (NANDF2ECC) is read.
CS3NAND	**NAND Flash mode for $\overline{\text{EMA_CS}}$[3].** • Set to 1 to enable NAND Flash mode for $\overline{\text{EMA_CS}}$[3]
CS2ECC	**NAND Flash ECC state for $\overline{\text{EMA_CS}}$[2].** • Set to 1 to start an ECC calculation for $\overline{\text{EMA_CS}}$[2] • Cleared to 0 when NAND Flash 1 ECC register (NANDF1ECC) is read.
CS2NAND	**NAND Flash mode for $\overline{\text{EMA_CS}}$[2].** • Set to 1 to enable NAND Flash mode for $\overline{\text{EMA_CS}}$[2]

18.2.5.6.1 Configuring for NAND Flash Mode

Similar to the asynchronous accesses previously described, the EMIFA's memory-mapped registers must be programmed appropriately to interface to a NAND Flash device. In addition to the fields listed in Table 18-15, the CS*n*NAND (*n* = 2, 3, 4, or 5) bit of the NAND Flash control register (NANDFCR) should be set to 1 to enter NAND Flash Mode. Note that the EW bit of CE*n*CFG should be cleared to avoid enabling the wait feature while in NAND Flash Mode.

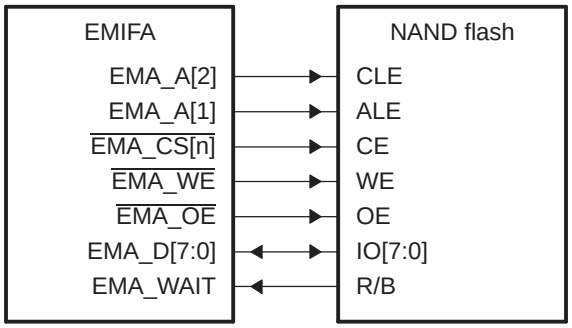
18.2.5.6.2 Connecting to NAND Flash

Figure 18-14 shows the EMIFA external pins used to interface with a NAND Flash device. EMIFA address lines are used to drive the NAND Flash device's command latch enable (CLE) and address latch enable (ALE) signals. Any EMIFA address lines may be used to drive the CLE and ALE signals of the NAND Flash.

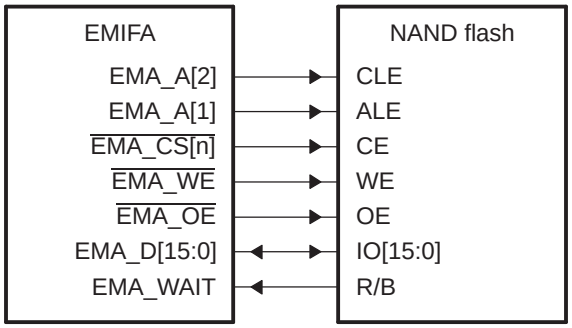
NOTE: The EMIFA will not control the NAND Flash device's write protect pin. The write protect pin must be controlled outside of the EMIFA.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 18-14. EMIFA to NAND Flash Interface



a) Connection to 8−bit NAND device



b) Connection to 16−bit NAND device

18.2.5.6.3 Driving CLE and ALE

As stated in Section 18.2.5.1, the EMIFA always drives the least significant bit of a 32-bit word address on EMA_A[0]. This functionality must be considered when attempting to drive the offset lines connected to CLE and ALE to the appropriate state.

For example, if using EMA_A[2] and EMA_A[1] to connect to CLE and ALE, respectively, the following offsets should be added to EMIFA base address:

- 0000 0000h to drive CLE and ALE low
- 0000 0010h to drive CLE high and ALE low
- 0000 0008h to drive CLE low and ALE high

18.2.5.6.4 NAND Read and Program Operations

A NAND Flash access cycle is composed of a command, address, and data phase. The EMIFA will not automatically generate these three phases to complete a NAND access with one transfer request. To complete a NAND access cycle, multiple single asynchronous access cycles must be completed by the EMIFA. Software must be used to request the appropriate asynchronous accesses to complete a NAND Flash access cycle. This software must be developed to the specification of the chosen NAND Flash device.

Since NAND operations are divided into single asynchronous access cycles, the chip select signal will not remain activated for the duration of the NAND operation. Instead, the chip select signal will deactivate between each asynchronous access cycle. For this reason, the EMIFA does not support NAND Flash devices that require the chip select signal to remain low during the t_R time for a read. See Section 18.2.5.6.8 for workaround.

Copyright © 2013–2016, Texas Instruments Incorporated

Care must be taken when performing a NAND read or write operation via the EDMA controller. See Section 18.2.5.6.5 for more details.

NOTE: The EMIFA does not support NAND Flash devices that require the chip select signal to remain low during the t_R time for a read. See Section 18.2.5.6.8 for workaround.

18.2.5.6.5 NAND Data Read and Write via EDMA Controller

When performing NAND accesses, the EDMA controller is most efficiently used for the data phase of the access. The command and address phases of the NAND access require only a few words of data to be transferred and therefore do not take advantage of the EDMA controller's ability to transfer larger quantities of data with a single request. In this section we will focus on using the EDMA controller for the data phase of a NAND access.

There are two conditions that require care to be taken when performing NAND reads and writes via the EDMA controller. These are:

- The address lines used to drive CLE and ALE signals must be driven low
- The EMIFA does not support constant addressing mode

Since the EMIFA does not support a constant addressing mode, when programming the EDMA, a linear incrementing address mode must be used. When using a linear incrementing address mode, if the CLE and ALE are driven by EMA_A[2] and EMA_A[1], respectively, care must be taken not to increase the address into a range that drives CLE and/or ALE high. To prevent the address from incrementing into a range that drives CLE and/or ALE high, the EDMA ACNT, BCNT, SIDX, DIDX, and synchronization type must be programmed appropriately. Following is an example configuration of EDMA controller when EMA_A[2] is connected to CLE and EMA_A[1] is connected to ALE.

EDMA setup for a NAND Flash data read:

- ACNT ≤ 8 bytes (this can also be set to less than or equal to the external data bus width)
- BCNT = transfer size in bytes/ACNT
- SIDX (source index) = 0
- DIDX (destination index) = ACNT
- AB synchronized

EDMA setup for a NAND Flash data write:

- ACNT ≤ 8 bytes (this can also be set to less than or equal to the external data bus width)
- BCNT = transfer size in bytes/ACNT
- SIDX (source index) = ACNT
- DIDX (destination index) = 0
- AB synchronized

18.2.5.6.6 ECC Generation

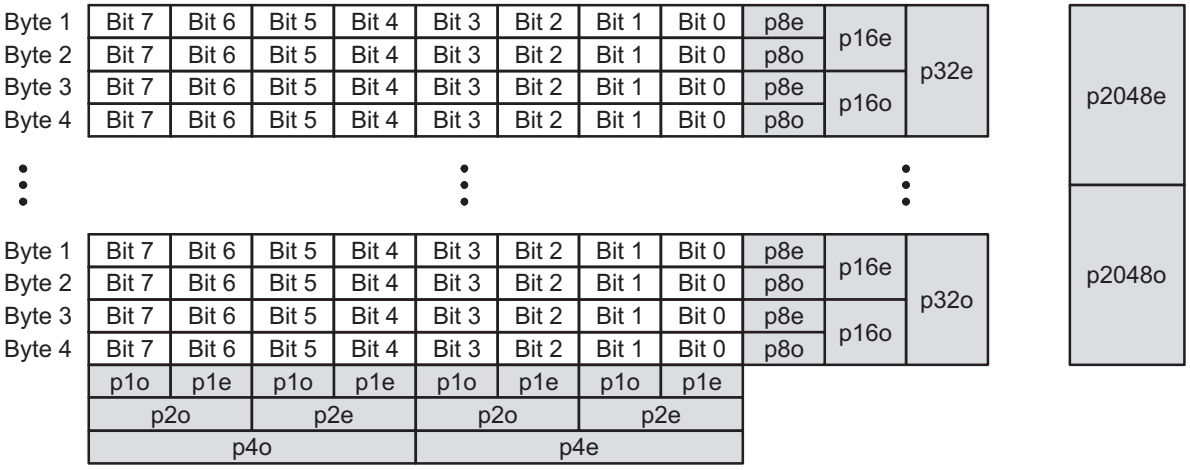
18.2.5.6.6.1 1-Bit ECC

If the CS*n*NAND (*n* = 2, 3, 4, or 5) bit in the NAND Flash control register (NANDFCR) is set to 1, the EMIFA supports 1-bit ECC calculation for up to 512 bytes for the corresponding chip select. To perform the ECC calculation, the CS*n*ECC (*n* = 2, 3, 4, or 5) bit in NANDFCR must be set to 1. It is the responsibility of the software to start the ECC calculation by writing to the CS*n*ECC (*n* = 2, 3, 4, or 5) bit prior to issuing a write or read to NAND Flash. It is also the responsibility of the software to read the calculated ECC from the NAND Flash *m* ECC register (NANDF*m*ECC) (*m* = 1, 2, 3, or 4) once the transfer to NAND Flash has completed. If the software writes or reads more than 512 bytes, the ECC will be incorrect. Reading the NAND*m*ECC (*m* = 1, 2, 3, or 4) clears the CS*n*ECC (*n* = 2, 3, 4, or 5) bit in NANDFCR. The NANDF*m*ECC (*m* = 1, 2, 3, or 4) is cleared upon writing a 1 to the CS*n*ECC (*n* = 2, 3, 4, or 5) bit. Figure 18-15 shows the algorithm used to calculate the ECC value for an 8-bit NAND Flash.

Copyright © 2013–2016, Texas Instruments Incorporated

For an 8-bit NAND Flash p1o through p4e are column parities and p8e through p2048o are row parities. Similarly, the algorithm can be extended to a 16-bit NAND Flash. For a 16-bit NAND Flash p1o through p8e are column parities and p16e through p2048o are row parities. The software must ignore the unwanted parity bits if ECC is desired for less than 512 bytes of data. For example. p2048e and p2048o are not required for ECC on 256 bytes of data. Similarly, p1024e, p1024o, p2048e, and p2048o are not required for ECC on 128 bytes of data.

Figure 18-15. ECC Value for 8-Bit NAND Flash



18.2.5.6.6.2 4-Bit ECC

The EMIFA supports 4-bit ECC on 8-bit/16-bit NAND Flash. In NAND mode, if the NAND Flash 4-bit ECC start bit (4BITECC_START) in the in the NAND Flash control register (NANDFCR) is set, the EMIFA calculates 4-bit ECC for the selected chip select. Only one chip select can be selected for the 4-bit ECC calculation at one time. The selection of the chip select is done by programming the 4-bit ECC CS select bit field (4BITECCSEL) in the NAND Flash control register (NANDFCR). The calculated parity (for writes) and syndrome (for reads) can be read from the NAND Flash 4-bit ECC 1-4 registers (NAND4BITECC[4:1]). The 4-bit ECC start bit (4BITECC_START) is cleared upon reading any of the NAND Flash 4-bit ECC 1-4 registers (NAND4BITECC[4:1]). The NAND Flash 4-bit ECC 1-4 registers are cleared upon writing one to the 4-bit ECC start bit (4BITECC_START).

The 4-bit ECC algorithm works on a 10-bit data bus, but only the lower eight bits of the data bus actually contain data. When the EMIFA is used in 16-bit mode, the lower and upper 8-bits of the 16-bit data read from the data bus are fed into the ECC engine one at a time, in that order. In all cases, since only 8-bits of data are fed to the ECC engine, the upper two bits of the 10-bit data bus that feeds the ECC engine are always zero. However, the parity and the syndrome value read from the NAND Flash 4-bit ECC 1-4 registers (NAND4BITECC[4:1]) are 10 bits wide. It is the responsibility of software to convert 10-bit parity values to 8 bits before writing to the spare location of the NAND Flash after a write operation. Similarly, it is the responsibility of the software to convert the 8-bit parity values read from the spare location of the NAND Flash after a read operation, to 10 bits before writing the NAND Flash 4-bit ECC load register (NAND4BITECCLOAD).

The 4-bit ECC employed in the EMIFA interface is a Reed-Solomon error correcting code. The symbol size is ten bits (two bits are always zero and eight bits contain data as described above). With eight 10-bit parity words, up to four symbols can be corrected per block read. Though the ECC operation is called 4-bit, it is important to note that correction can actually happen on up to four 10-bit symbols. Only the lower eight bits of each 10-bit symbol actually contain data (see above), so correction can happen on up to four bytes. When bit errors are randomly distributed through the block of data read from the NAND, those errors are not likely to fall into the same bytes of data, so 4-bits of correction is an apt description. Technically speaking, however, more than four bits of error can be corrected if multiple bit errors are confined to four or fewer bytes of the data. If bit errors fall into more than four bytes, the ECC engine will report that there are too many errors to correct.

Copyright © 2013–2016, Texas Instruments Incorporated

At the end of the syndrome calculation after read, the error address and the error value can be calculated by setting the address and error value calculation start bit (4BITECC_ADD_CALC_START) in the NAND Flash control register (NANDFCR). The end of address calculation is flagged by the 4-bit ECC correction state field (ECC_STATE) in the NAND Flash status register (NANDFSR). The number of errors can be read from the 4-bit number of errors field (ECC_ERRNUM) in the NAND Flash status register (NANDFSR). The error address value can be read from the NAND Flash error address 1-2 registers (NANDERRADD[2:1]). The error value can be read from the NAND Flash error value 1-2 registers (NANDERRVAL[2:1]). The address and error value start bit (4BITECC_ADD_CALC_START) is cleared upon reading any of the NAND Flash error address 1-2 registers (NANDERRADD[2:1]) or the NAND Flash error value 1-2 registers (NANDERRVAL[2:1]). The EMIFA registers the syndrome value internally before the error address and error value calculation. Therefore, a new read operation can be performed simultaneously with the error address calculation.

The EMIFA supports 4-bit ECC calculation up to 518 bytes. The software needs to follow the following procedure for 4-bit ECC calculation:

For writes:

1. Set the 4BITECC_START bit in the NAND Flash control register (NANDFCR) to 1.
2. Write 518 bytes of data to the NAND Flash.
3. Read the parity from the NAND Flash 4-Bit ECC 1-4 registers (NAND4BITECC[4:1]).
4. Convert the 10-bit parity values to 8-bits. All 10-bit parity values can be concatenated together with ECC value 1 (4BITECCVAL1) as LSB and ECC value 8 (4BITECCVAL8) as MSB. Then the concatenated value can be broken down into ten 8-bit values.
5. Store the parity to spare location in the NAND Flash.

For reads:

1. Set the 4BITECC_START bit in the NAND Flash control register (NANDFCR) to 1.
2. Read 518 bytes of data from the NAND Flash.
3. Clear the 4BITECC_START bit in NANDFCR by reading any of the NAND Flash 4-bit ECC registers.
4. Read the parity stored in the spare location in the NAND Flash.
5. Convert the 8-bit parity values to 10-bits. Reverse of the conversion that was done during writes.
6. Write the parity values in the NAND Flash 4-bit ECC load register (NAND4BITECCLOAD). Write each parity value one at a time starting from 4BITECCVAL8 down to 4BITECCVAL1.
7. Perform a dummy read to the NAND Flash status register (NANDFSR). This is only required to ensure time for syndrome calculation after writing the ECC values in step 6.
8. Read the syndrome from the NAND Flash 4-bit ECC 1-4 registers (NAND4BITECC[4:1]). A syndrome value of 0 means no bit errors. If the syndrome is non-zero, continue with step 9.
9. Set the 4BITECC_ADD_CALC_START bit in the NAND Flash control register (NANDFCR) to 1.
10. Perform a dummy read to any EMIFA registers except the NAND Flash error address 1-2 registers (NANDERRADD[2:1]) or the NAND Flash error value 1-2 registers (NANDERRVAL[2:1]).
11. Start another read from NAND, if required (a new thread from step 1).
12. Wait for the 4-bit ECC correction state field (ECC_STATE) in the NAND Flash status register (NANDFSR) to be equal to 1, 2h, or 3h.
13. The number of errors can be read from the 4-bit number of errors field (ECC_ERRNUM) in the NAND Flash status register (NANDFSR).
14. Read the error address from the NAND Flash error address 1-2 registers (NANDERRADD[2:1]). Address for the error word is equal to (total_words_read + 7 - address_value). For 518 bytes, the address will be equal to (525 - address_value).
15. Read the error value from the NAND Flash error value 1-2 registers (NANDERRVAL[2:1]). Errors can be corrected by XORing the error word with the error value from the NAND Flash error value 1-2 registers (NANDERRVAL[2:1]).

Copyright © 2013–2016, Texas Instruments Incorporated

18.2.5.6.7 NAND Flash Status Register (NANDFSR)

The NAND Flash status register (NANDFSR) indicates the raw status of the EMA_WAIT pin while in NAND Flash Mode. The EMA_WAIT pin should be connected to the NAND Flash device's R/$\overline{B}$ signal, so that it indicates whether or not the NAND Flash device is busy. During a read, the R/$\overline{B}$ signal will transition and remain low while the NAND Flash retrieves the data requested. Once the R/$\overline{B}$ signal transitions high, the requested data is ready and should be read by the EMIFA. During a write/program operation, the R/$\overline{B}$ signal transitions and remains low while the NAND Flash is programming the Flash with the data it has received from the EMIFA. Once the R/$\overline{B}$ signal transitions high, the data has been written to the Flash and the next phase of the transaction may be performed. From this explanation, you can see that the NAND Flash status register is useful to the software for indicating the status of the NAND Flash device and determining when to proceed to the next phase of a NAND Flash operation.

When a rising edge occurs on the EMA_WAIT pin, the EMIFA sets the WR (Wait Rise) bit in the EMIFA interrupt raw register (INTRAW). Therefore, the EMIFA Wait Rise interrupt may be used to indicate the status of the NAND Flash device. The WP*n* bit in the asynchronous wait cycle configuration register (AWCC) does not affect the NAND Flash status register (NANDFSR) or the WR bit in INTRAW. See Section 18.2.8 for more a detailed description of the wait rise interrupt.

18.2.5.6.8 Interfacing to a Non-CE Don't Care NAND Flash

As explained in Section 18.2.5.6.4, the EMIFA does not support NAND Flash devices that require the chip select signal to remain low during the t_R time for a read. One way to work around this limitation is to use a GPIO pin to drive the $\overline{CE}$ signal of the NAND Flash device. If this work around is implemented, software will configure the selected GPIO to be low, then begin the NAND Flash operation, starting with the command phase. Once the NAND Flash operation has completed the software can then configure the selected GPIO to be high.

18.2.5.7 Extended Wait Mode and the EMA_WAIT Pin

The EMIFA supports the Extend Wait Mode. This is a mode in which the external asynchronous device may assert control over the length of the strobe period. The Extended Wait Mode can be entered by setting the EW bit in the asynchronous *n* configuration register (CE*n*CFG) (*n* = 2, 3, 4, or 5). When this bit is set, the EMIFA monitors the EMA_WAIT pin to determine if the attached device wishes to extend the strobe period of the current access cycle beyond the programmed number of clock cycles.

When the EMIFA detects that the EMA_WAIT pin has been asserted, it will begin inserting extra strobe cycles into the operation until the EMA_WAIT pin is deactivated by the external device. The EMIFA will then return to the last cycle of the programmed strobe period and the operation will proceed as usual from this point. Please refer to the device data manual for details on the timing requirements of the EMA_WAIT signal.

The EMA_WAIT pin cannot be used to extend the strobe period indefinitely. The programmable MAX_EXT_WAIT field in the asynchronous wait cycle configuration register (AWCC) determines the maximum number of EMA_CLK cycles the strobe period may be extended beyond the programmed length. When the counter expires, the EMIFA proceeds to the hold period of the operation regardless of the state of the EMA_WAIT pin. The EMIFA can also generate an interrupt upon expiration of this counter. See Section 18.2.8.1 for details on enabling this interrupt.

For the EMIFA to function properly in the Extended Wait mode, the WP*n* bit of AWCC must be programmed to match the polarity of the EMA_WAIT pin. In its reset state of 1, the EMIFA will insert wait cycles when the EMA_WAIT pin is sampled high. When set to 0, the EMIFA will insert wait cycles only when EMA_WAIT is sampled low. This programmability allows for a glueless connection to larger variety of asynchronous devices.

Finally, a restriction is placed on the strobe period timing parameters when operating in Extended Wait mode. Specifically, the sum of the W_SETUP and W_STROBE fields must be greater than 4, and the sum of the R_SETUP and R_STROBE fields must be greater than 4 for the EMIFA to recognize the EMA_WAIT pin has been asserted. The W_SETUP, W_STROBE, R_SETUP, and R_STROBE fields are in CE*n*CFG.

Copyright © 2013–2016, Texas Instruments Incorporated

18.2.6 Data Bus Parking

The EMIFA always drives the data bus to the previous write data value when it is idle. This feature is called data bus parking. Only when the EMIFA issues a read command to the external memory does it stop driving the data bus. The data bus is released (tri-stated) when the chip enable (EMA_CS[n]) is asserted by EMIFA for the read access. After the read operation is completed, the data bus is driven again by the bus parking feature at the end of the turnaround time. At all other times that the EMIF is enabled but not actively transferring data, the bus parking feature drives the data bus to the last written value.

The one exception to this behavior occurs after performing an asynchronous read operation while the EMIFA is in the self-refresh state. In this situation, the read operation is not followed by the EMIFA parking the data bus. Instead, the EMIFA tri-states the data bus. Therefore, it is not recommended to perform asynchronous read operations while the EMIFA is in the self-refresh state, in order to prevent floating inputs on the data bus. External pull-ups, such as 10kΩ resistors, should be placed on the 16 EMIFA data bus pins (which do not have internal pull-ups) if it is required to perform reads in this situation. The precise resistor value should be chosen so that the worst case combined off-state leakage currents do not cause the voltage levels on the associated pins to drop below the high-level input voltage requirement.

For information about the self-refresh state, see Section 18.2.4.7.

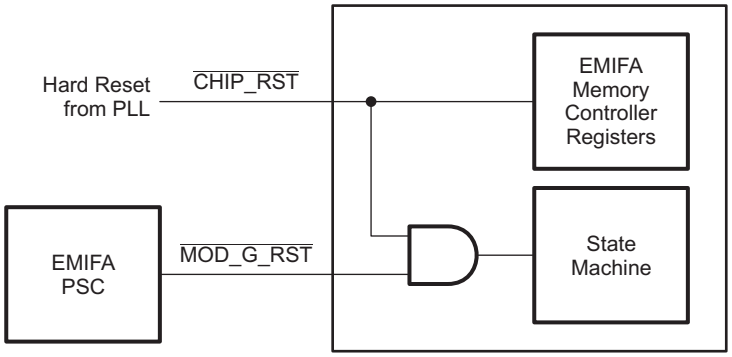
18.2.7 Reset and Initialization Considerations

The EMIFA memory controller has two reset signals, CHIP_RST and MOD_G_RST. The CHIP_RST is a module-level reset that resets both the state machine as well as the EMIFA memory controller's memory-mapped registers. The MOD_G_RST resets the state machine only. If the EMIFA memory controller is reset independently of other peripherals, the user's software should not perform memory, as well as register accesses, while CHIP_RST or MOD_G_RST are asserted. If memory or register accesses are performed while the EMIFA memory controller is in the reset state, other masters may hang. Following the rising edge of CHIP_RST or MOD_G_RST, the EMIFA memory controller immediately begins its initialization sequence. Command and data stored in the EMIFA memory controller FIFOs are lost. Table 18-24 describes the different methods for asserting each reset signal. Figure 18-16 shows the EMIFA memory controller reset diagram.

Table 18-24. Reset Sources

Reset Signal	Reset Source
CHIP_RST	Hardware/ Device Reset
MOD_G_RST	Power and Sleep Controller

Figure 18-16. EMIFA Reset Block Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

The EMIFA and its registers are reset when any of the following events occur:

1. The $\overline{\text{RESET}}$ pin on the device is asserted
2. An emulator reset is initiated through the Code Composer Studio™ integrated development environment

In the first case, the EMIFA will exit the reset state when $\overline{\text{RESET}}$ is released and after the PLL controller releases the entire device from reset. In the second case, the EMIFA will exit the reset state immediately after the emulator reset is complete.

In both cases, the EMIFA automatically begins running the SDRAM initialization sequence described in Section 18.2.4.4 after coming out of reset. Even though the initialization procedure is automatic, a special procedure, found in Section 18.2.4.5 must still be followed.

18.2.8 Interrupt Support

The EMIFA supports a single interrupt to the CPU. Section 18.2.8.1 details the generation and internal masking of EMIFA interrupts, and Section 18.2.8.2 describes how the EMIFA interrupts are sent to the CPU.

18.2.8.1 Interrupt Events

There are three conditions that may cause the EMIFA to generate an interrupt to the CPU. These conditions are:

- A rising edge on the EMA_WAIT signal (wait rise interrupt)
- An asynchronous time out
- Usage of unsupported addressing mode (line trap interrupt)

The wait rise interrupt occurs when a rising edge is detected on EMA_WAIT signal. This interrupt generation is not affected by the WP*n* bit in the asynchronous wait cycle configuration register (AWCC). The asynchronous time out interrupt condition occurs when the attached asynchronous device fails to deassert the EMA_WAIT pin within the number of cycles defined by the MAX_EXT_WAIT bit in AWCC (this happens only in extended wait mode). EMIFA supports only linear incrementing and cache line wrap addressing modes . If an access request for an unsupported addressing mode is received, the EMIFA will set the LT bit in the EMIFA interrupt raw register (INTRAW) and treat the request as a linear incrementing request.

Only when the interrupt is enabled by setting the appropriate bit (WR_MASK_SET/AT_MASK_SET/LT_MASK_SET) in the EMIFA interrupt mask set register (INTMSKSET) to 1, will the interrupt be sent to the CPU. Once enabled, the interrupt may be disabled by writing a 1 to the corresponding bit in the EMIFA interrupt mask clear register (INTMSKCLR). The bit fields in both the INTMSKSET and INTMSKCLR may be used to indicate whether the interrupt is enabled. When the interrupt is enabled, the corresponding bit field in both the INTMSKSET and INTMSKCLR will have a value of 1; when the interrupt is disabled, the corresponding bit field will have a value of 0.

The EMIFA interrupt raw register (INTRAW) and the EMIFA interrupt mask register (INTMSK) indicate the status of each interrupt. The appropriate bit (WR/AT/LT) in INTRAW is set when the interrupt condition occurs, whether or not the interrupt has been enabled. However, the appropriate bit (WR_MASKED/AT_MASKED/LT_MASKED) in INTMSK is set only when the interrupt condition occurs and the interrupt is enabled. Writing a 1 to the bit in INTRAW clears the INTRAW bit as well as the corresponding bit in INTMSK. Table 18-25 contains a brief summary of the interrupt status and control bit fields. See Section 18.4 for complete details on the register fields.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 18-25. Interrupt Monitor and Control Bit Fields

Register Name	Bit Name	Description
EMIFA interrupt raw register (INTRAW)	WR	This bit is set when an rising edge on the EMA_WAIT signal occurs. Writing a 1 clears the WR bit as well as the WR_MASKED bit in INTMSK.
	AT	This bit is set when an asynchronous timeout occurs. Writing a 1 clears the AT bit as well as the AT_MASKED bit in INTMSK.
	LT	This bit is set when an unsupported addressing mode is used. Writing a 1 clears LT bit as well as the LT_MASKED bit in INTMSK.
EMIFA interrupt mask register (INTMSK)	WR_MASKED	This bit is set only when a rising edge on the EMA_WAIT signal occurs and the interrupt has been enabled by writing a 1 to the WR_MASK_SET bit in INTMSKSET.
	AT_MASKED	This bit is set only when an asynchronous timeout occurs and the interrupt has been enabled by writing a 1 to the AT_MASK_SET bit in INTMSKSET.
	LT_MASKED	This bit is set only when line trap interrupt occurs and the interrupt has been enabled by writing a 1 to the LT_MASK_SET bit in INTMSKSET.
EMIFA interrupt mask set register (INTMSKSET)	WR_MASK_SET	Writing a 1 to this bit enables the wait rise interrupt.
	AT_MASK_SET	Writing a 1 to this bit enables the asynchronous timeout interrupt.
	LT_MASK_SET	Writing a 1 to this bit enables the line trap interrupt.
EMIFA interrupt mask clear register (INTMSKCLR)	WR_MASK_CLR	Writing a 1 to this bit disables the wait rise interrupt.
	AT_MASK_CLR	Writing a 1 to this bit disables the asynchronous timeout interrupt.
	LT_MASK_CLR	Writing a 1 to this bit disables the line trap interrupt.

18.2.8.2 Interrupt Multiplexing

For details on EMIFA interrupt multiplexing, see your device-specific data manual.

18.2.8.3 Interrupt Processing

For details on EMIFA interrupt processing, see the *ARM Interrupt Controller (AINTC)* chapter.

18.2.9 EDMA Event Support

EMIFA memory controller is a DMA slave peripheral and therefore does not generate DMA events. Data read and write requests may be made directly, by masters and the DMA.

18.2.10 Pin Multiplexing

For details on EMIFA pin multiplexing, see your device-specific data manual.

18.2.11 Memory Map

For information describing the device memory-map, see your device-specific data manual.

Copyright © 2013–2016, Texas Instruments Incorporated

18.2.12 Priority and Arbitration

Section 18.2.2 describes the external prioritization and arbitration among requests from different sources within the SoC. The result of this external arbitration is that only one request is presented to the EMIFA at a time. Once the EMIFA completes a request, the external arbiter then provides the EMIFA with the next pending request.

Internally, the EMIFA undertakes memory device transactions according to a strict priority scheme. The highest priority events are:

- A device reset.
- A write to any of the three least significant bytes of the SDRAM configuration register (SDCR).

Either of these events will cause the EMIFA to immediately commence its initialization sequence as described in Section 18.2.4.4.

Once the EMIFA has completed its initialization sequence, it performs memory transactions according to the following priority scheme (highest priority listed first):

1. If the EMIFA's backlog refresh counter is at the Refresh Must urgency level, the EMIFA performs multiple SDRAM auto refresh cycles until the Refresh Release urgency level is reached.
2. If an SDRAM or asynchronous read has been requested, the EMIFA performs a read operation.
3. If the EMIFA's backlog refresh counter is at the Refresh Need urgency level, the EMIFA performs an SDRAM auto refresh cycle.
4. If an SDRAM or asynchronous write has been requested, the EMIFA performs a write operation.
5. If the EMIFA's backlog refresh counter is at the Refresh May or Refresh Release urgency level, the EMIFA performs an SDRAM auto refresh cycle.
6. If the value of the SR bit in SDCR has been set to 1, the EMIFA will enter the self-refresh state as described in Section 18.2.4.7.

After taking one of the actions listed above, the EMIFA then returns to the top of the priority list to determine its next action.

Because the EMIFA does not issue auto-refresh cycles when in the self-refresh state, the above priority scheme does not apply when in this state. See Section 18.2.4.7 for details on the operation of the EMIFA when in the self-refresh state.

Copyright © 2013–2016, Texas Instruments Incorporated

18.2.13 System Considerations

This section describes various system considerations to keep in mind when operating the EMIFA.

18.2.13.1 Asynchronous Request Times

In a system that interfaces to both SDRAM and asynchronous memory, the asynchronous requests must not take longer than the smaller of the following two values:

- t_{RAS} (typically 120 μs) - to avoid violating the maximum time allowed between issuing an ACTV and PRE command to the SDRAM.
- $t_{Refresh\ Rate}$ × 11 (typically 15.7 μs × 11 = 172.7 μs) - to avoid refresh violations on the SDRAM.
  The length of an asynchronous request is controlled by multiple factors, the primary factor being the number of access cycles required to complete the request. For example, an asynchronous request for 4 bytes will require four access cycles using an 8-bit data bus and only two access cycle using a 16-bit data bus. The maximum request size that the EMIFA can be sent is 16 words, therefore the maximum number of access cycles per memory request is 64 when the EMIFA is configured with an 8-bit data bus. The length of the individual access cycles that make up the asynchronous request is determined by the programmed setup, strobe, hold, and turnaround values, but can also be extended with the assertion of the EMA_WAIT input signal up to a programmed maximum limit. It is up to the user to make sure that an entire asynchronous request does not exceed the timing values listed above when also interfacing to an SDRAM device. This can be done by limiting the asynchronous timing parameters.

18.2.13.2 Cache Fill Requests

The CPU can run code from either internal or external memory. When running code from external memory, the CPU's program cache is periodically filled with eight words (32-bytes) through a dedicated port to the EMIFA. Two system level concerns arise when filling the program cache from the EMIFA.

First, the program cache fills have the possibility of being locked out from accessing the EMIFA by a stream of higher priority requests. Therefore, care should be taken when issuing persistent requests to the EMIFA from a source such which is a high priority requester.

Second, requests to the EMIFA from the other sources risk missing their deadlines while a program cache fill from the EMIFA is in progress. This is because all other EMIFA accesses are held pending while the program cache is filled. The worst-case scenario that can arise is when a requester submits a request immediately after a program cache fill request has begun. The system should be analyzed to make sure that this worst-case request delay is acceptable.

Copyright © 2013–2016, Texas Instruments Incorporated

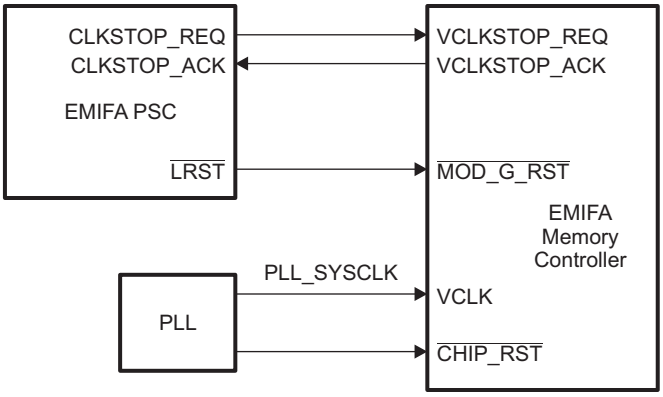
18.2.14 Power Management

Power dissipation from the EMIFA memory controller may be managed by following methods:

- Self-refresh mode
- Power-down mode
- Gating input clocks to the module off

Gating input clocks off to the EMIFA memory controller achieves higher power savings when compared to the power savings of self-refresh or power down mode. The input clocks are turned off outside of the EMIFA memory controller through the use of the Power and Sleep Controller (PSC) and the PLL controller. Figure 18-17 shows the connections between the EMIFA memory controller, PSC, and PLL. Before gating clocks off, the EMIFA memory controller must place the SDR SDRAM memory in self-refresh mode. If the external memory requires a continuous clock, the clock provided by the PLL must not be turned off because this may result in data corruption. See the following subsections for the proper procedures to follow when stopping the EMIFA memory controller clocks.

Figure 18-17. EMIFA PSC Block Diagram



18.2.14.1 Power Management Using Self-Refresh Mode

The EMIFA can be placed into a self-refresh state in order to place the attached SDRAM devices into self-refresh mode, which consumes less power for most SDRAM devices. In this state, the attached SDRAM device uses an internal clock to perform its own auto refresh cycles. This maintains the validity of the data in the SDRAM without the need for any external commands. Refer to Section 18.2.4.7 for more details on placing the EMIFA into the self-refresh state.

18.2.14.2 Power Management Using Power Down Mode

In case of power down, to lower the power consumption, EMIFA drives EMA_SDCKE low. EMA_SDCKE goes high when there is a need to send refresh (REFR) commands, after which EMA_SDCKE is again driven low. EMA_SDCKE remains low until any request arrives. Refer to Section 18.2.4.8 for more details on placing EMIFA in power down mode.

Copyright © 2013–2016, Texas Instruments Incorporated

18.2.14.3 Power Management Using Clock Stop

The LPSC of the memory controller can be programmed to be in one of the following states:

- Enable
- Auto Sleep
- Auto Wake
- Sync Reset

After the EMIFA clock is enabled, by default it is in the enable state. EMIFA can be put to auto sleep state, when the clock is to be gated off. Auto Wake brings back EMIFA to the enable state from the auto sleep state.

18.2.14.3.1 Auto Sleep and Auto Wake

To achieve maximum power savings EMIFA core clock should be gated off. EMIFA memory controller can make use of auto sleep and auto wake to achieve clock gating. Following describes the procedure to be followed to put EMIFA memory controller in auto sleep state:

- EMIFA should be put to self-refresh mode before stopping the clock. Refer to Section 18.2.4.7 for details on self-refresh mode. The EMIFA memory controller will complete any outstanding accesses and backlogged refresh cycles and then place the EMIFA memory in self-refresh mode.
- Then, program the LPSC of EMIFA for auto sleep, to gate off the clocks.

Register and memory access requests are honored while EMIFA is in auto sleep state. When EMIFA sees a request while it is in auto sleep state, it automatically returns to enable state, processes the request, and returns back to auto sleep state until further requests come.

On frequent requests, EMIFA switches between auto sleep and enable states. To bring EMIFA back to the enable state, auto wake can be used. Following procedure is followed for performing auto wake.

- Program the LPSC of EMIFA for auto wake.
- Bring EMIFA out of self-refresh. Refer to Section 18.2.4.7 for details on self-refresh mode.

After auto wake, EMIFA is in enable state and clocks run continuously.

18.2.14.3.2 Sync Reset and Enable

Sync reset of EMIFA through the LPSC does not reset the EMIFA registers or memory. Thus EMIFA LPSC sync reset behavior is similar to EMIFA LPSC auto sleep, except that register or memory requests are not honored by EMIFA. Following is the procedure to put EMIFA in sync reset state:

- EMIFA should be put to self-refresh mode before stopping the clock. Refer to Section 18.2.4.7 for details on self-refresh mode. The EMIFA memory controller will complete any outstanding accesses and backlogged refresh cycles and then place the EMIFA memory in self-refresh mode.
- Then, program the LPSC of EMIFA to Sync-Reset state.

On sync reset, requests to EMIFA are not honored. To bring EMIFA back to the enable state, use the following enable procedure:

- Program the LPSC of EMIFA to enter enable state.
- Bring EMIFA out of self-refresh. Refer to Section 18.2.4.7 for details on self-refresh mode.

Now EMIFA memory controller is in the enable state and continues with normal operation.

18.2.15 Emulation Considerations

EMIFA memory controller will remain fully functional during emulation halts, to allow emulation access to external memory.

Copyright © 2013–2016, Texas Instruments Incorporated

18.3 Example Configuration

This section presents an example of interfacing the EMIFA to both an SDR SDRAM device and an asynchronous flash device.

18.3.1 Hardware Interface

Figure 18-18 shows the hardware interface between the EMIFA, a Samsung K4S641632H-TC(L)70 64Mb SDRAM device, and two SHARP LH28F800BJE-PTTL90 8Mb Flash memory. The connection between the EMIFA and the SDRAM is straightforward, but the connection between the EMIFA and the flash deserves a detailed look.

The address inputs for the flash are provided by three sources. The A[12:0] address inputs are provided by a combination of the EMA_A and EMA_BA pins according to Section 18.2.5.1. The upper address inputs A[18:13] are provided by GPIO pins. The six GPIO pins are connected to the upper address bits of the flash memory and attached to pulldown resistors so that their value is 0 after reset and before configuring the pins as GPIO. This is necessary if the ROM bootloader is copying the secondary bootloader from the flash. More details on using GPIO pins as upper address pins can be found in Section 18.2.5.2. RD/$\overline{BY}$ signal from one flash is connected to EMA_WAIT pin of EMIFA. A GPIO pin can be made use of to receive the RD/$\overline{BY}$ signal coming from the second flash, as shown in Figure 18-18

Finally, this example configuration connects the $\overline{EMA_WE}$ pin to the $\overline{WE}$ input of the flash and operates the EMIFA in Normal Mode.

18.3.2 Software Configuration

The following sections describe how to interface the EMIFA to SDRAM, Asynchronous SRAM (ASRAM), or a NAND Flash device.

18.3.2.1 Configuring the SDRAM Interface

This section describes how to configure the EMIFA to interface with the Samsung K4S641632H-TC(L)70 SDRAM with a clock frequency of f_{EMA_CLK} = 100 MHz. Procedure A described in Section 18.2.4.5 is followed which assumes that the SDRAM power-up timing constraint were met during the SDRAM Auto-Initialization sequence after Reset.

18.3.2.1.1 PLL Programming for the EMIFA to K4S641632H-TC(L)70 Interface

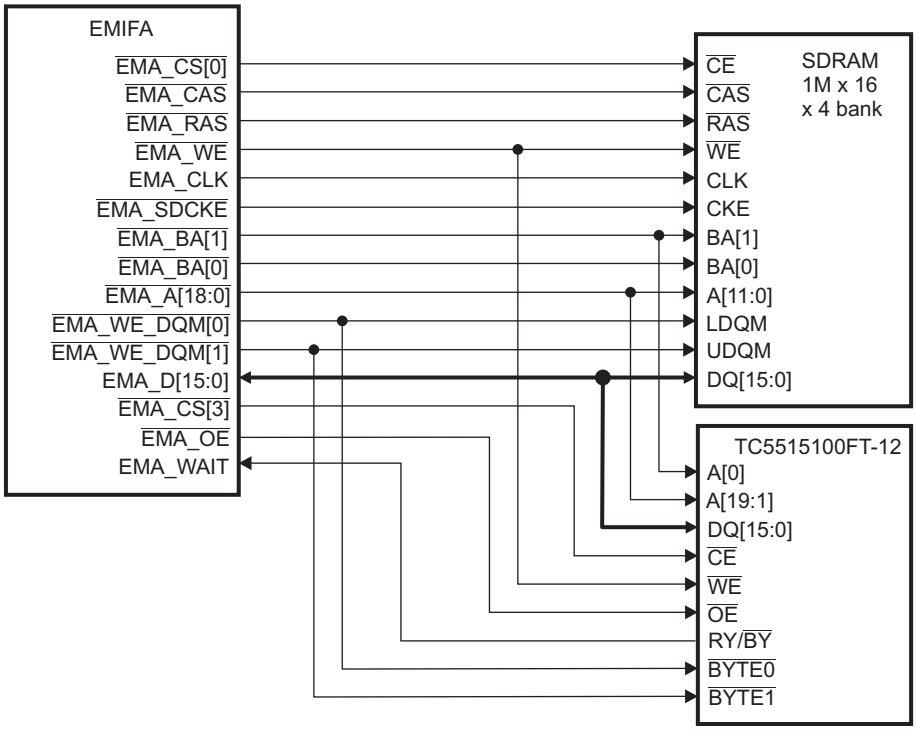
The device PLL Controller should first be programmed to select the desired EMA_CLK frequency. Before doing this, the SDRAM should be placed in Self-Refresh Mode by setting the SR bit in the SDRAM configuration register (SDCR). The SR bit should be set using a byte-write to the upper byte of the SDCR to avoid triggering the SDRAM Initialization Sequence. The EMA_CLK frequency can now be adjusted to the desired value by programming the appropriate SYSCLK domain of the PLL Controller. Once the PLL has been reprogrammed, remove the SDRAM from Self-Refresh by clearing the SR bit in SDCR, again with a byte-write.

Table 18-26. SR Field Value For the EMIFA to K4S641632H-TC(L)70 Interface

Field	Value	Purpose
SR	1 then 0	To place the EMIFA into the self refresh state

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 18-18. Example Configuration Interface



Copyright © 2013–2016, Texas Instruments Incorporated

18.3.2.1.2 SDRAM Timing Register (SDTIMR) Settings for the EMIFA to K4S641632H-TC(L)70 Interface

The fields of the SDRAM timing register (SDTIMR) should be programmed first as described in Table 18-27 to satisfy the required timing parameters for the K4S641632H-TC(L)70. Based on these calculations, a value of 6111 4610h should be written to SDTIMR. Figure 18-19 shows a graphical description of how SDTIMR should be programmed.

Table 18-27. SDTIMR Field Calculations for the EMIFA to K4S641632H-TC(L)70 Interface

Field Name	Formula	Value from K4S641632H-TC(L)70 Datasheet	Value Calculated for Field
T_RFC	T_RFC >= (t_{RFC} × f_{EMA_CLK}) - 1	t_{RC} = 68 ns (min)(1)	6
T_RP	T_RP >= (t_{RP} × f_{EMA_CLK}) - 1	t_{RP} = 20 ns (min)	1
T_RCD	T_RCD >= (t_{RCD} × f_{EMA_CLK}) - 1	t_{RCD} = 20 ns (min)	1
T_WR	T_WR >= (t_{WR} × f_{EMA_CLK}) - 1	t_{RDL} = 2 CLK = 20 ns (min)(2)	1
T_RAS	T_RAS >= (t_{RAS} × f_{EMA_CLK}) - 1	t_{RAS} = 49 ns (min)	4
T_RC	T_RC >= (t_{RC} × f_{EMA_CLK}) - 1	t_{RC} = 68 ns (min)	6
T_RRD	T_RRD >= (t_{RRD} × f_{EMA_CLK}) - 1	t_{RRD} = 14 ns (min)	1

(1) The Samsung datasheet does not specify a t_{RFC} value. Instead, Samsung specifies t_{RC} as the minimum auto refresh period.
(2) The Samsung datasheet does not specify a t_{WR} value. Instead, Samsung specifies t_{RDL} as last data in to row precharge minimum delay.

Figure 18-19. SDRAM Timing Register (SDTIMR)

31 27	26 24	23	22 20	19	18 16
0 0110	001	0	001	0	001
T_RFC	T_RP	Rsvd	T_RCD	Rsvd	T_WR

15 12	11 8	7	6 4	3 0
0100	0110	0	001	0000
T_RAS	T_RC	Rsvd	T_RRD	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

18.3.2.1.3 SDRAM Self Refresh Exit Timing Register (SDSRETR) Settings for the EMIFA to K4S641632H-TC(L)70 Interface

The SDRAM self refresh exit timing register (SDSRETR) should be programmed second to satisfy the t_{XSR} timing requirement from the K4S641632H-TC(L)70 datasheet. Table 18-28 shows the calculation of the proper value to program into the T_XS field of this register. Based on this calculation, a value of 6h should be written to SDSRETR. Figure 18-20 shows how SDSRETR should be programmed.

Table 18-28. RR Calculation for the EMIFA to K4S641632H-TC(L)70 Interface

Field Name	Formula	Value from K4S641632H-TC(L)70 Datasheet	Value Calculated for Field
T_XS	T_XS >= (t_{XSR} × f_{EMA_CLK}) - 1	t_{RC} = 68 ns (min)(1)	6

(1) The Samsung datasheet does not specify a t_{XSR} value. Instead, Samsung specifies t_{RC} as the minimum required time after CKE going high to complete self refresh exit.

Figure 18-20. SDRAM Self Refresh Exit Timing Register (SDSRETR)

31	16
0000 0000 0000 0000	
Reserved	

15	5	4	0
000 0000 0000		0 0110	
Reserved		T_XS	

18.3.2.1.4 SDRAM Refresh Control Register (SDRCR) Settings for the EMIFA to K4S641632H-TC(L)70 Interface

The SDRAM refresh control register (SDRCR) should next be programmed to satisfy the required refresh rate of the K4S641632H-TC(L)70. Table 18-29 shows the calculation of the proper value to program into the RR field of this register. Based on this calculation, a value of 61Ah should be written to SDRCR. Figure 18-21 shows how SDRCR should be programmed.

Table 18-29. RR Calculation for the EMIFA to K4S641632H-TC(L)70 Interface

Field Name	Formula	Values	Value Calculated for Field
RR	RR ≤ f_{EMA_CLK} × $t_{Refresh\ Period}$ / n_{cycles}	From SDRAM datasheet: $t_{Refresh\ Period}$ = 64 ms; n_{cycles} = 4096 EMIFA clock rate: f_{EMA_CLK} = 100 MHz	RR = 1562 cycles = 61Ah cycles

Figure 18-21. SDRAM Refresh Control Register (SDRCR)

31	19	18	16
0 0000 0000 0000		000	
Reserved		Reserved	

15	13	12	0
000		0 0110 0001 1010 (61Ah)	
Reserved		RR	

Copyright © 2013–2016, Texas Instruments Incorporated

18.3.2.1.5 SDRAM Configuration Register (SDCR) Settings for the EMIFA to K4S641632H-TC(L)70 Interface

Finally, the fields of the SDRAM configuration register (SDCR) should be programmed as described in Table 18-30 to properly interface with the K4S641632H-TC(L)70 device. Based on these settings, a value of 4720h should be written to SDCR. Figure 18-22 shows how SDCR should be programmed. The EMIFA is now ready to perform read and write accesses to the SDRAM.

Table 18-30. SDCR Field Values For the EMIFA to K4S641632H-TC(L)70 Interface

Field	Value	Purpose
SR	0	To avoid placing the EMIFA into the self refresh state
NM	1	To configure the EMIFA for a 16-bit data bus
CL	011b	To select a CAS latency of 3
BIT11_9LOCK	1	To allow the CL field to be written
IBANK	010b	To select 4 internal SDRAM banks
PAGESIZE	0	To select a page size of 256 words

Figure 18-22. SDRAM Configuration Register (SDCR)

31	30	29	28				24
0	0	0	0 0000				
SR	Reserved	Reserved	Reserved				

23					18	17	16
00 0000						0	0
Reserved						Reserved	Reserved

15	14	13	12	11		9	8
0	1	0	0	011			1
Reserved	NM	Reserved	Reserved	CL			BIT11_9LOCK

7	6		4	3	2		0
0	010			0	000		
Reserved	IBANK			Reserved	PAGESIZE		

Copyright © 2013–2016, Texas Instruments Incorporated

18.3.2.2 Interfacing to Asynchronous SRAM (ASRAM)

The following example describes how to interface the EMIFA to the Toshiba TC55V16100FT-12 device.

18.3.2.2.1 Meeting AC Timing Requirements for ASRAM

When configuring the EMIFA to interface to ASRAM, you must consider the AC timing requirements of the ASRAM as well as the AC timing requirements of the EMIFA. These can be found in the data sheet for each respective device. The read and write asynchronous cycles are programmed separately in the asynchronous configuration register (CE*n*CFG).

For a read access, Table 18-31 to Table 18-33 list the AC timing specifications that must be considered.

Table 18-31. EMIFA Input Timing Requirements

Parameter	Description
t_{SU}	Data Setup time, data valid before $\overline{\text{EMA_OE}}$ high
t_H	Data Hold time, data valid after $\overline{\text{EMA_OE}}$ high

Table 18-32. ASRAM Output Timing Characteristics

Parameter	Description
t_{ACC}	Address Access time
t_{OH}	Output data Hold time for address change
t_{COD}	Output Disable time from chip enable

Table 18-33. ASRAM Input Timing Requirement for a Read

Parameter	Description
t_{RC}	Read Cycle time

Figure 18-23 shows an asynchronous read access and describes how the EMIFA and ASRAM AC timing requirements work together to define the values for R_SETUP, R_STROBE, and R_HOLD.

From Figure 18-23, the following equations may be derived. t_{cyc} is the period at which the EMIFA operates. The R_SETUP, R_STROBE, and R_HOLD fields are programmed in terms of EMIFA cycles where as the data sheet specifications are typically given in nanoseconds. This explains the presence of t_{cyc} in the denominator of the following equations. A minus 1 is included in the equations because each field in CE*n*CFG is programmed in terms of EMIFA clock cycles, minus 1 cycle. For example, R_SETUP is equal to R_SETUP width in EMIFA clock cycles minus 1 cycle.

$$\text{R_SETUP} + \text{R_STROBE} \geq \frac{t_{ACC}(m) + t_{SU}}{t_{cyc}} - 2$$

$$\text{R_SETUP} + \text{R_STROBE} + \text{R_HOLD} \geq \frac{t_{RC}(m)}{t_{cyc}} - 3$$

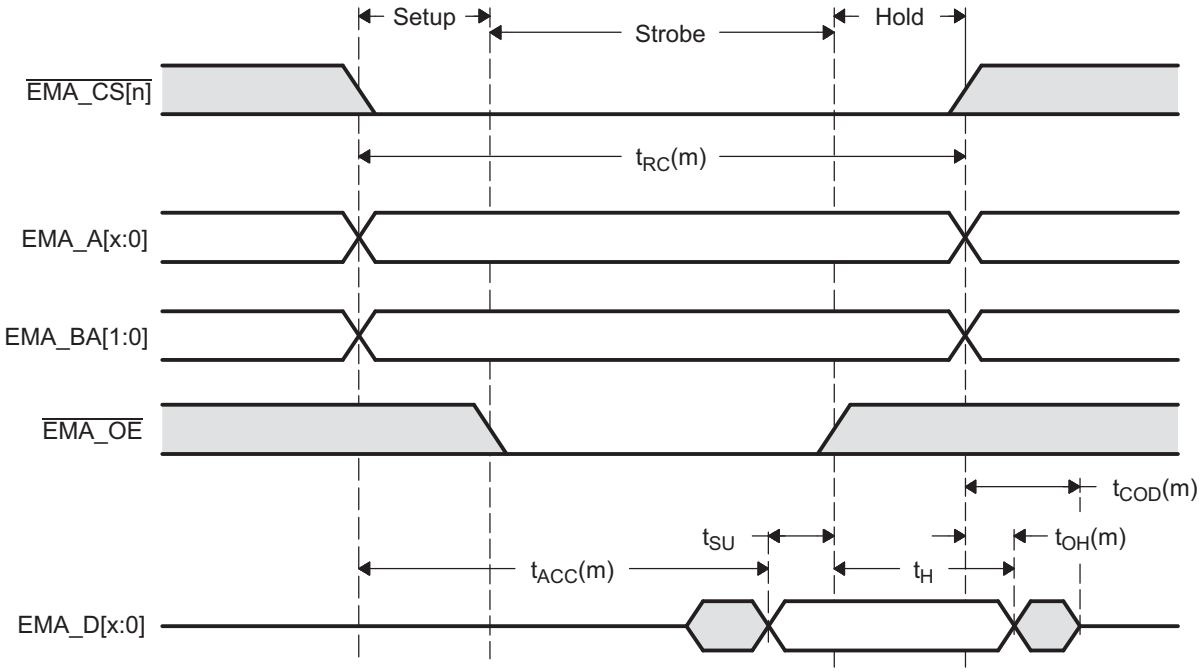
$$\text{R_HOLD} \geq \frac{\left(t_H - t_{OH}(m)\right)}{t_{cyc}} - 1$$

Copyright © 2013–2016, Texas Instruments Incorporated

The EMIFA offers an additional parameter, TA, that defines the turnaround time between read and write cycles. This parameter protects against the situation when the output turn-off time of the memory is longer than the time it takes to start the next write cycle. If this is the case, the EMIFA will drive data at the same time as the memory, causing contention on the bus. By examining Figure 18-23, the equation for TA can be derived as:

$$TA \geq \frac{t_{COD}(m)}{t_{cyc}} - 1$$

Figure 18-23. Timing Waveform of an ASRAM Read



For a write access, Table 18-34 lists the AC timing specifications that must be satisfied.

Table 18-34. ASRAM Input Timing Requirements for a Write

Parameter	Description
t_{WP}	Write Pulse width
t_{AW}	Address valid to end of Write
t_{DS}	Data Setup time
t_{WR}	Write Recovery time
t_{DH}	Data Hold time
t_{WC}	Write Cycle time

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 18-24 shows an asynchronous write access and describes how the EMIFA and ASRAM AC timing requirements work together to define values for W_SETUP, W_STROBE, and W_HOLD.

From Figure 18-24, the following equations may be derived. t_{cyc} is the period at which the EMIFA operates. The W_SETUP, W_STROBE, and W_HOLD fields are programmed in terms of EMIFA cycles where as the data sheet specifications are typically given is nano seconds. This is explains the presence of t_{cyc} in the denominator of the following equations. A minus 1 is included in the equations because each field in CE*n*CFG is programmed in terms of EMIFA clock cycles, minus 1 cycle. For example, W_SETUP is equal to W_SETUP width in EMIFA clock cycles minus 1 cycle.≥

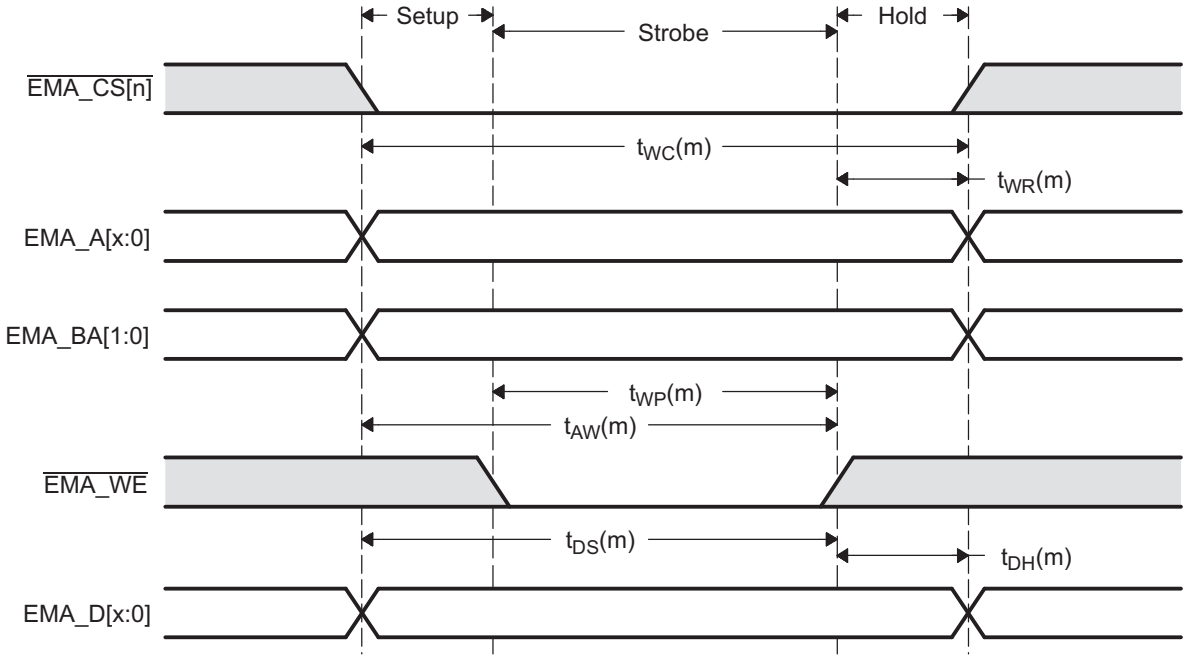
$$\text{W_STROBE} \geq \frac{t_{WP}(m)}{t_{cyc}} - 1$$

$$\text{W_SETUP} + \text{W_STROBE} \geq \max\left(\frac{t_{AW}(m)}{t_{cyc}}, \frac{t_{DS}(m)}{t_{cyc}}\right) - 2$$

$$\text{W_HOLD} \geq \max\left(\frac{t_{WR}(m)}{t_{cyc}}, \frac{t_{DH}(m)}{t_{cyc}}\right) - 1$$

$$\text{W_SETUP} + \text{W_STROBE} + \text{W_HOLD} \geq \frac{t_{WC}(m)}{t_{cyc}} - 3$$

Figure 18-24. Timing Waveform of an ASRAM Write



Copyright © 2013–2016, Texas Instruments Incorporated

18.3.2.2.2 Taking Into Account PCB Delays

The equations described in Section 18.3.2.2.1 are for the ideal case, when board design does not contribute delays. Board characteristics, such as impedance, loading, length, number of nodes, etc., affect how the device driver behaves. Signals driven by the EMIFA will be delayed when they reach the ASRAM and conversely. Table 18-35 lists the delays shown in Figure 18-25 and Figure 18-26 due to PCB affects. The PCB delays are board specific and must be estimated or determined though the use of IBIS modeling. The signals denoted (ASRAM) are the signals seen at the ASRAM. For example, $\overline{\text{EMA_CS}}$ represents the signal at the EMIFA and $\overline{\text{EMA_CS}}$ (ASRAM) represents the delayed signal seen at the ASRAM.

Table 18-35. ASRAM Timing Requirements With PCB Delays

Parameter	Description
	Read Access
t_{EM_CS}	Delay on $\overline{\text{EMA_CS}}$ from EMIFA to ASRAM. $\overline{\text{EMA_CS}}$ is driven by EMIF.
t_{EM_A}	Delay on EMA_A from EMIFA to ASRAM. EMA_A is driven by EMIF.
t_{EM_OE}	Delay on $\overline{\text{EMA_OE}}$ from EMIFA to ASRAM. $\overline{\text{EMA_OE}}$ is driven by EMIF.
t_{EM_D}	Delay on EMA_D from ASRAM to EMIFA. EMA_D is driven by ASRAM.
	Write Access
t_{EM_CS}	Delay on $\overline{\text{EMA_CS}}$ from EMIFA to ASRAM. $\overline{\text{EMA_CS}}$ is driven by EMIF.
t_{EM_A}	Delay on EMA_A from EMIFA to ASRAM. EMA_A is driven by EMIF.
t_{EM_WE}	Delay on $\overline{\text{EMA_WE}}$ from EMIFA to ASRAM. $\overline{\text{EMA_WE}}$ is driven by EMIF.
t_{EM_D}	Delay on EMA_D from EMIFA to ASRAM. EMA_D is driven by EMIF.

From Figure 18-25, the following equations may be derived. t_{cyc} is the period at which the EMIFA operates. The R_SETUP, R_STROBE, and R_HOLD fields are programmed in terms of EMIFA cycles where as the data sheet specifications are typically given in nano seconds. This is explains the presence of t_{cyc} in the denominator of the following equations. A minus 1 is included in the equations because each field in CE*n*CFG is programmed in terms of EMIFA clock cycles, minus 1 cycle. For example, R_SETUP is equal to R_SETUP width in EMIFA clock cycles minus 1 cycle.

$$\text{R_SETUP} + \text{R_STROBE} \geq \frac{(t_{EM_A} + t_{ACC}(m) + t_{SU} + t_{EM_D})}{t_{cyc}} - 2$$

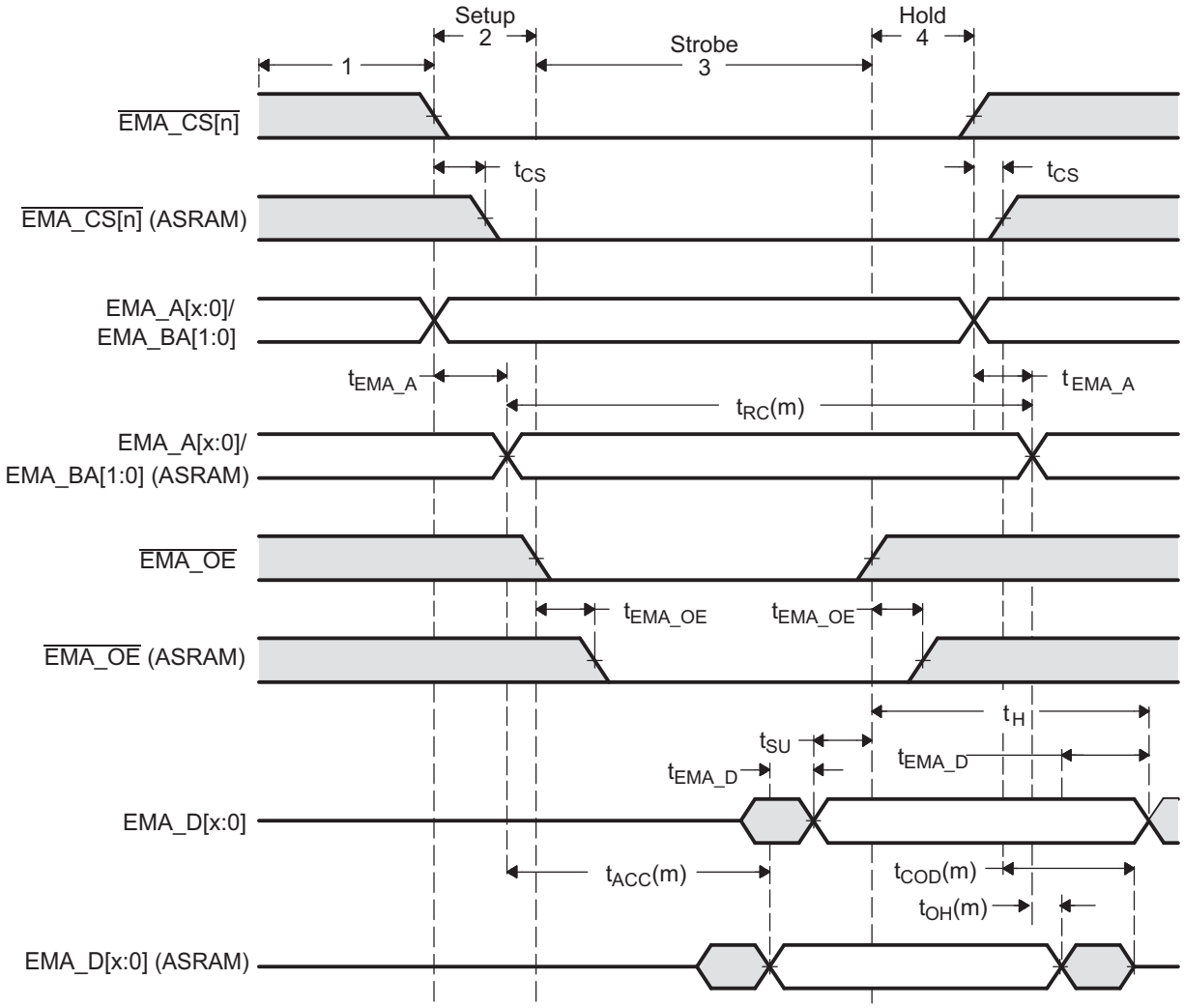
$$\text{R_SETUP} + \text{R_STROBE} + \text{R_HOLD} \geq \frac{t_{RC}(m)}{t_{cyc}} - 3$$

$$\text{R_HOLD} \geq \frac{\left(t_H - t_{EM_D} - t_{OH}(m) - t_{EM_A}\right)}{t_{cyc}} - 1$$

$$\text{TA} \geq \frac{\left(t_{EM_CS} + t_{COD}(m) + t_{EM_D}\right)}{t_{cyc}} - 1$$

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 18-25. Timing Waveform of an ASRAM Read with PCB Delays



From Figure 18-26, the following equations may be derived. t_{cyc} is the period at which the EMIFA operates. The W_SETUP, W_STROBE, and W_HOLD fields are programmed in terms of EMIFA cycles where as the data sheet specifications are typically given is nano seconds. This is explains the presence of t_{cyc} in the denominator of the following equations. A minus 1 is included in the equations because each field in CE*n*CFG is programmed in terms of EMIFA clock cycles, minus 1 cycle. For example, W_SETUP is equal to W_SETUP width in EMIFA clock cycles minus 1 cycle.

Copyright © 2013–2016, Texas Instruments Incorporated

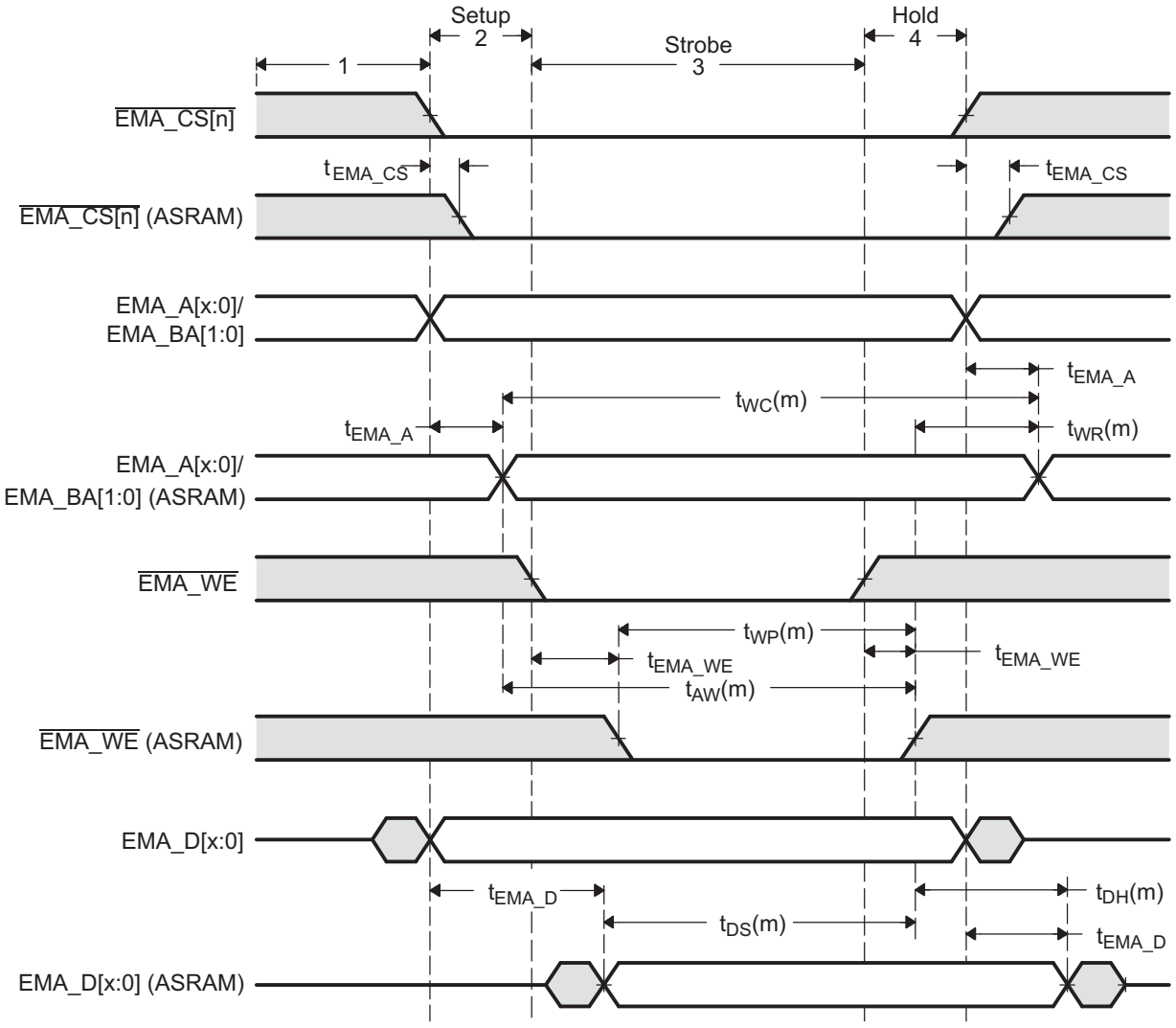
$$W_STROBE \geq \frac{t_{WP}(m)}{t_{cyc}} - 1$$

$$W_SETUP + W_STROBE \geq \max\left(\frac{(t_{EM_A} + t_{AW}(m) - t_{EM_WE})}{t_{cyc}}, \frac{(t_{EM_D} + t_{DS}(m) - t_{EM_WE})}{t_{cyc}}\right) - 2$$

$$W_HOLD \geq \max\left(\frac{(t_{EM_WE} + t_{WR}(m) - t_{EM_A})}{t_{cyc}}, \frac{(t_{EM_WE} + t_{DH}(m) - t_{EM_D})}{t_{cyc}}\right) - 1$$

$$W_SETUP + W_STROBE + W_HOLD \geq \frac{t_{WC}(m)}{t_{cyc}} - 3$$

Figure 18-26. Timing Waveform of an ASRAM Write with PCB Delays



Copyright © 2013–2016, Texas Instruments Incorporated

18.3.2.2.3 Example Using TC5516100FT-12

This section takes you through the configuration steps required to implement Toshiba's TC55V1664FT-12 ASRAM with the EMIFA. The following assumptions are made:

- ASRAM is connected to chip select space 3 ($\overline{\text{EMA_CS[3]}}$)
- EMIFA clock speed is 100 MHz (t_{cyc} = 10 nS)

Table 18-36 lists the data sheet specifications for the EMIFA and Table 18-37 lists the data sheet specifications for the ASRAM.

Table 18-36. EMIFA Timing Requirements for TC5516100FT-12 Example

Parameter	Description	Min	Max	Units
t_{SU}	Data Setup time, data valid before $\overline{\text{EMA_OE}}$ high	3 to 7 (1)		nS
t_H	Data Hold time, data valid after $\overline{\text{EMA_OE}}$ high	0		nS

(1) Depending on operating conditions. See your device-specific data manual for the value.

Table 18-37. ASRAM Timing Requirements for TC5516100FT-12 Example

Parameter	Description	Min	Max	Units
t_{ACC}	Address Access time		12	nS
t_{OH}	Output data Hold time for address change	3		nS
t_{RC}	Read cycle time	12		nS
t_{WP}	Write Pulse width	8		nS
t_{AW}	Address valid to end of Write	9		nS
t_{DS}	Data Setup time	7		nS
t_{WR}	Write Recovery time	0		nS
t_{DH}	Data Hold time	0		nS
t_{WC}	Write Cycle time	12		nS
t_{COD}	Output Disable time from chip enable		7	

Table 18-38 lists the values of the PCB board delays. The delays were estimated using the rule that there is 180 pS of delay for every 1 inch of trace.

Table 18-38. Measured PCB Delays for TC5516100FT-12 Example

Parameter	Description	Delay (ns)
	Read Access	
t_{EM_CS}	Delay on $\overline{\text{EMA_CS}}$ from EMIFA to ASRAM. $\overline{\text{EMA_CS}}$ is driven by EMIF.	0.36
t_{EM_A}	Delay on EMA_A from EMIFA to ASRAM. EMA_A is driven by EMIF.	0.27
t_{EM_OE}	Delay on $\overline{\text{EMA_OE}}$ from EMIFA to ASRAM. $\overline{\text{EMA_OE}}$ is driven by EMIF.	0.36
t_{EM_D}	Delay on EMA_D from ASRAM to EMIFA. EMA_D is driven by ASRAM.	0.45
	Write Access	
t_{EM_CS}	Delay on $\overline{\text{EMA_CS}}$ from EMIFA to ASRAM. $\overline{\text{EMA_CS}}$ is driven by EMIF.	0.36
t_{EM_A}	Delay on EMA_A from EMIFA to ASRAM. EMA_A is driven by EMIF.	0.27
t_{EM_WE}	Delay on $\overline{\text{EMA_WE}}$ from EMIFA to ASRAM. $\overline{\text{EMA_WE}}$ is driven by EMIF.	0.36
t_{EM_D}	Delay on EMA_D from EMIFA to ASRAM. EMA_D is driven by EMIF.	0.45

Copyright © 2013–2016, Texas Instruments Incorporated

Inserting these values into the equations defined above allows you to determine the values for SETUP, STROBE, HOLD, and TA. For a read:

$$R_SETUP + R_STROBE \geq \frac{(t_{EM_A} + t_{ACC}(m) + t_{SU} + t_{EM_D})}{t_{cyc}} - 2 \geq \frac{(0.27 + 12 + 5 + 0.45)}{10} - 2 \geq -0.23$$

$$R_SETUP + R_STROBE + R_HOLD \geq \frac{t_{RC}(m)}{t_{cyc}} - 3 \geq \left(\frac{12}{10}\right) - 3 \geq -1.8$$

$$R_HOLD \geq \frac{\left(t_H - t_{EM_D} - t_{OH}(m) - t_{EM_A}\right)}{t_{cyc}} - 1 \geq \frac{(0 - 0.45 - 3 - 0.27)}{10} - 1 \geq -1.37$$

$$TA \geq \frac{\left(t_{EM_CS} + T_{COD}(m) + t_{EM_D}\right)}{t_{cyc}} - 1 \geq \frac{(0.36 + 7 + 0.45)}{10} - 1 \geq -0.22$$

Therefore if R_SETUP = 0, then R_STROBE = 0, R_HOLD = 0, and TA = 0.

For a write:

$$W_STROBE \geq \frac{t_{WP}(m)}{t_{cyc}} - 1 \geq \left(\frac{8}{10}\right) - 1 \geq -0.2$$

$$W_SETUP + W_STROBE \geq \max\left(\frac{(t_{EM_A} + t_{AW}(m) - t_{EM_WE})}{t_{cyc}}, \frac{(t_{EM_D} + t_{DS}(m) - t_{EM_WE})}{t_{cyc}}\right) - 2$$

$$\geq \max\left(\frac{(0.36 + 0 - 0.27)}{10}, \frac{(0.36 + 0 - 0.45)}{10}\right) - 2 \geq -2.01$$

$$W_HOLD \geq \max\left(\frac{\left(t_{EM_WE} + t_{WR}(m) - t_{EM_A}\right)}{t_{cyc}}, \frac{\left(t_{EM_WE} + t_{DH}(m) - t_{EM_D}\right)}{t_{cyc}}\right) - 1$$

$$\geq \max\left(\frac{(0.27 + 9 - 0.36)}{10}, \frac{(0.45 + 7 - 0.36)}{10}\right) - 1 \geq -0.1$$

$$W_SETUP + W_STROBE + W_HOLD \geq \frac{t_{WC}(m)}{t_{cyc}} - 3 \geq \left(\frac{12}{10}\right) - 3 \geq -1.8$$

Therefore, W_SETUP = 0, W_STROBE = 0, and W_HOLD = 0.

Copyright © 2013–2016, Texas Instruments Incorporated

Since the value of the W_SETUP/R_SETUP, W_STROBE/R_STROBE, W_HOLD/R_HOLD, and TA fields are equal to EMIFA clock cycles minus 1 cycle, the CE3CFG should be configured as in Table 18-39. In this example, the EMA_WAIT signal is not implemented; therefore, the asynchronous wait cycle configuration register (AWCC) does not need to be programmed.

Table 18-39. Configuring CE3CFG for TC5516100FT-12 Example

Parameter	Setting
SS	**Select Strobe mode.** • SS = 0. Places EMIFA in Normal Mode.
EW	**Extended Wait mode enable.** • EW = 0. Disabled Extended wait mode.
W_SETUP/R_SETUP	**Read/Write setup widths.** • W_SETUP = 0 • R_SETUP = 0
W_STROBE/R_STROBE	**Read/Write strobe widths.** • W_STROBE = 0 • R_STROBE = 0
W_HOLD/R_HOLD	**Read/Write hold widths.** • W_HOLD = 0 • R_HOLD = 0
TA	**Minimum turnaround time.** • TA = 0
ASIZE	**Asynchronous Device Bus Width.** • ASIZE = 1, select a 16-bit data bus width

18.3.2.3 Interfacing to NAND Flash

The following example explains how to interface the EMIFA to the Hynix HY27UA081G1M NAND Flash device.

18.3.2.3.1 Margin Requirements

The Flash interface is typically a low-performance interface compared to synchronous memory interfaces, high-speed asynchronous memory interfaces, and high-speed FIFO interfaces. For this reason, this example gives little attention to minimizing the amount of margin required when programming the asynchronous timing parameters. The approach used requires approximately 10 ns of margin on all parameters, which is not significant for a 100-ns read or write cycle. For additional details on minimizing the amount of margin, see the ASRAM example given in Section 18.3.2.2.

Table 18-40. Recommended Margins

Timing Parameter	Recommended Margin
Output Setup	10 nS
Output Hold	10 nS
Input Setup	10 nS
Input Hold	10 nS

Copyright © 2013–2016, Texas Instruments Incorporated

18.3.2.3.2 Meeting AC Timing Requirements for NAND Flash

When configuring the EMIFA to interface to NAND Flash, you must consider the AC timing requirements of the NAND Flash as well as the AC timing requirements of the EMIFA. These can be found in the data sheet for each respective device. The read and write asynchronous cycles are programmed separately in the asynchronous configuration register (CE*n*CFG).

A NAND Flash access cycle is composed of a command, address, and data phases. The EMIFA will not automatically generate these three phases to complete a NAND access with one transfer request. To complete a NAND access cycle, multiple single asynchronous access cycles must be completed by the EMIFA. The command and address phases of a NAND Flash access cycle are asynchronous writes performed by the EMIFA where as the data phase can be either an asynchronous write or a read depending on whether the NAND Flash is being programmed or read.

Therefore, to determine the required EMIFA configuration to interface to the NAND Flash for a read operation, Table 18-41 and Table 18-42 list the AC timing parameters that must be considered.

Table 18-41. EMIFA Read Timing Requirements

Parameter	Description
t_{SU}	Data Setup time, data valid before $\overline{\text{EMA_OE}}$ high
t_{H}	Data Hold time, data valid after $\overline{\text{EMA_OE}}$ high

Table 18-42. NAND Flash Read Timing Requirements

Parameter	Description
t_{RP}	Read Pulse width
t_{REA}	Read Enable Access time
t_{CEA}	Chip Enable low to output valid
t_{CHZ}	Chip Enable high to output High-Z
t_{RC}	Read Cycle time
t_{RHZ}	Read enable high to output High-Z
t_{CLR}	Command Latch low to Read enable low

Figure 18-27 shows an asynchronous read access and describes how the EMIFA and NAND Flash AC timing requirements work together to define the values for R_SETUP, R_STROBE, and R_HOLD.

Copyright © 2013–2016, Texas Instruments Incorporated

From Figure 18-27, the following equations may be derived. t_{cyc} is the period at which the EMIFA operates. The R_SETUP, R_STROBE, and R_HOLD fields are programmed in terms of EMIFA cycles where as the data sheet specifications are typically given is nano seconds. This is explains the presence of t_{cyc} in the denominator of the following equations. A minus 1 is included in the equations because each field in CE*n*CFG is programmed in terms of EMIFA clock cycles, minus 1 cycle. For example, R_SETUP is equal to R_SETUP width in EMIFA clock cycles minus 1 cycle.

$$\text{R_SETUP} \geq \frac{t_{CLR}(m)}{t_{cyc}} - 1$$

$$\text{R_STROBE} \geq \max\left(\frac{(t_{REA}(m) + t_{SU})}{t_{cyc}}, \frac{t_{RP}(m)}{t_{cyc}}\right) - 1$$

$$\text{R_SETUP} + \text{R_STROBE} \geq \frac{(t_{CEA}(m) + t_{SU})}{t_{cyc}} - 2$$

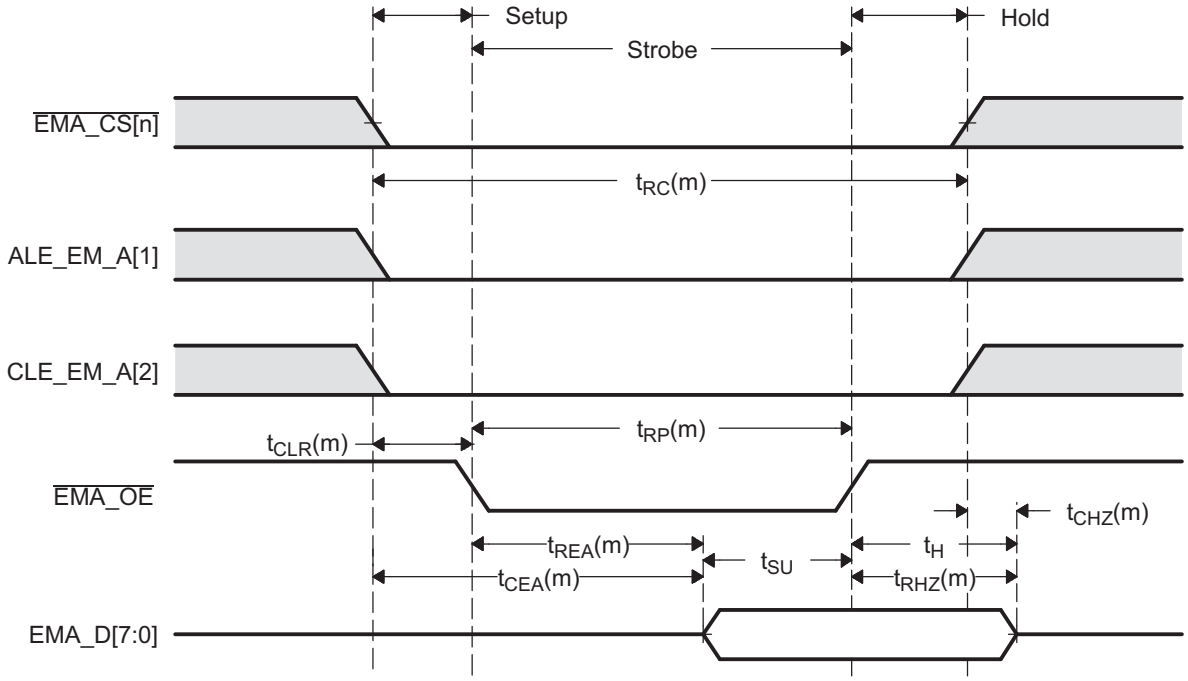
$$\text{R_HOLD} \geq \frac{(t_H - t_{CHZ}(m))}{t_{cyc}} - 1$$

$$\text{R_SETUP} + \text{R_STROBE} + \text{R_HOLD} \geq \frac{t_{RC}(m)}{t_{cyc}} - 3$$

The EMIFA offers an additional parameter, TA, that defines the turnaround time between read and write cycles. This parameter protects against the situation when the output turn-off time of the memory is longer than the time it takes to start the next write cycle. If this is the case, the EMIFA will drive data at the same time as the memory, causing contention on the bus. By examining Figure 18-27, the equation for TA can be derived as:

$$TA \geq \max\left(\frac{t_{CHZ}(m)}{t_{cyc}}, \frac{t_{RHZ}(m) - (\text{R_HOLD} + 1)t_{cyc}}{t_{cyc}}\right) - 1$$

Figure 18-27. Timing Waveform of a NAND Flash Read



Copyright © 2013–2016, Texas Instruments Incorporated

To determine the required EMIFA configuration to interface to the NAND Flash for a write operation, Table 18-43 lists the NAND AC timing parameters for a command latch, address latch, and data input latch that must be considered.

Table 18-43. NAND Flash Write Timing Requirements

Parameter	Description
t_{WP}	Write Pulse width
t_{CLS}	CLE Setup time
t_{ALS}	ALE Setup time
t_{CS}	$\overline{CS}$ Setup time
t_{DS}	Data Setup time
t_{CLH}	CLE Hold time
t_{ALH}	ALE Hold time
t_{CH}	$\overline{CS}$ Hold time
t_{DH}	Data Hold time
t_{WC}	Write Cycle time

Figure 18-28 to Figure 18-30 show the command latch, address latch, and data input latch of the NAND access.

From Figure 18-28 to Figure 18-30, the following equations may be derived. t_{cyc} is the period at which the EMIFA operates. The W_SETUP, W_STROBE, and W_HOLD fields are programmed in terms of EMIFA cycles where as the data sheet specifications are typically given is nano seconds. This is explains the presence of t_{cyc} in the denominator of the following equations. A minus 1 is included in the equations because each field in CE*n*CFG is programmed in terms of EMIFA clock cycles, minus 1 cycle. For example, W_SETUP is equal to W_SETUP width in EMIFA clock cycles minus 1 cycle.

$$\text{W_SETUP} \geq \max\left(\frac{t_{CLS}(m)}{t_{cyc}}, \frac{t_{ALS}(m)}{t_{cyc}}, \frac{t_{CS}(m)}{t_{cyc}}\right) - 1$$

$$\text{W_STROBE} \geq \frac{t_{WP}(m)}{t_{cyc}} - 1$$

$$\text{W_SETUP} + \text{W_STROBE} \geq \frac{t_{DS}(m)}{t_{cyc}} - 2$$

$$\text{W_HOLD} \geq \max\left(\frac{t_{CLH}(m)}{t_{cyc}}, \frac{t_{ALH}(m)}{t_{cyc}}, \frac{t_{CH}(m)}{t_{cyc}}, \frac{t_{DH}(m)}{t_{cyc}}\right) - 1$$

$$\text{W_SETUP} + \text{W_STROBE} + \text{W_HOLD} \geq \frac{t_{WC}(m)}{t_{cyc}} - 3$$

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 18-28. Timing Waveform of a NAND Flash Command Write

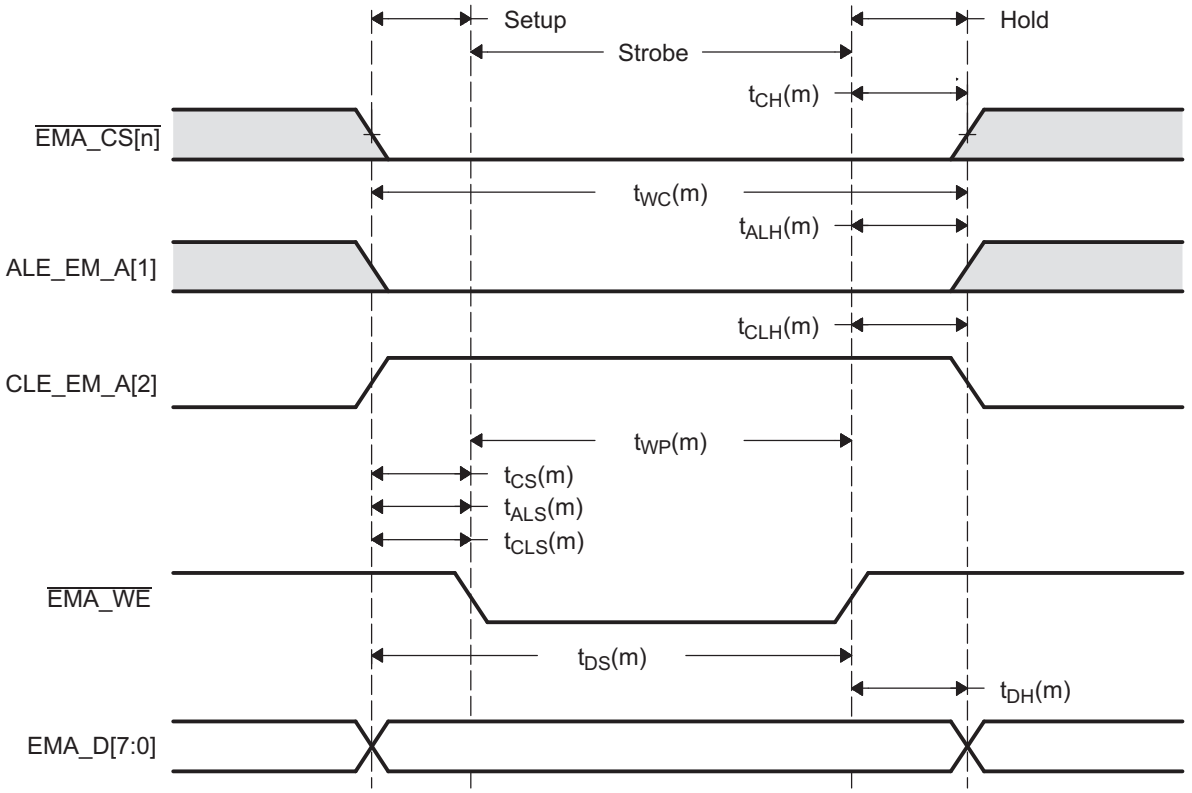
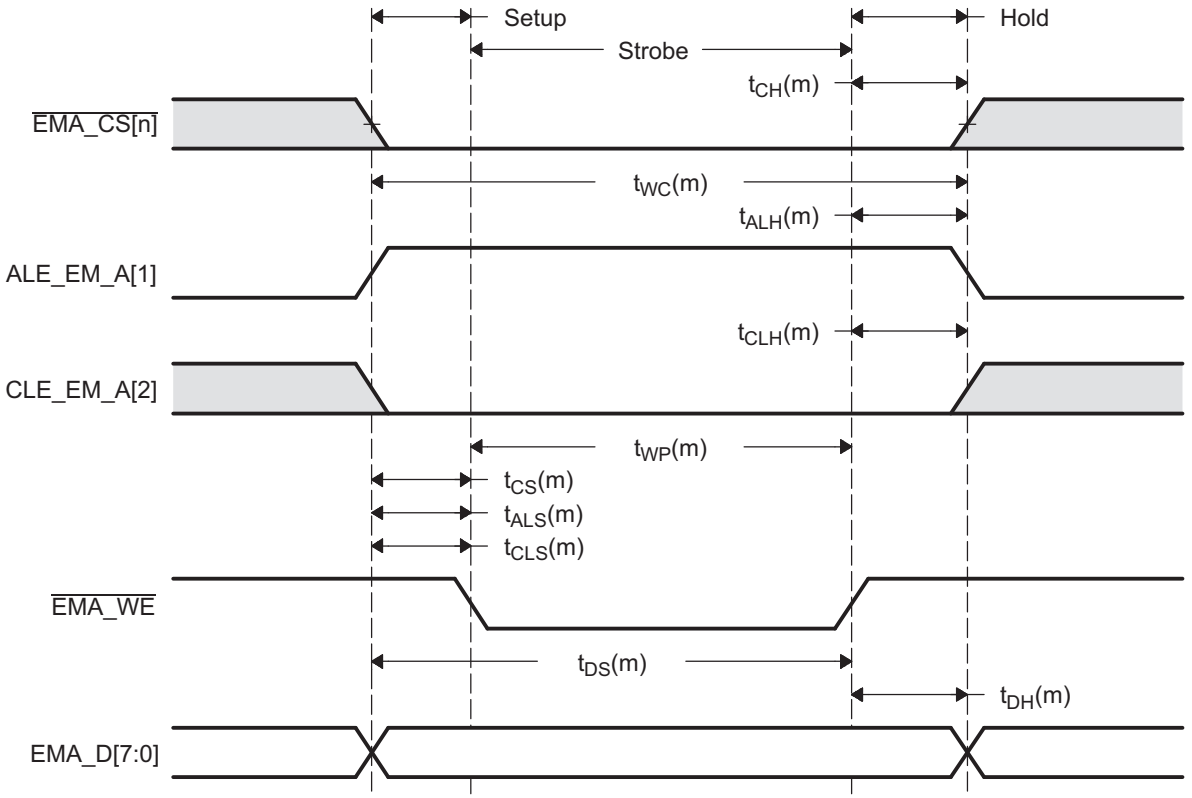
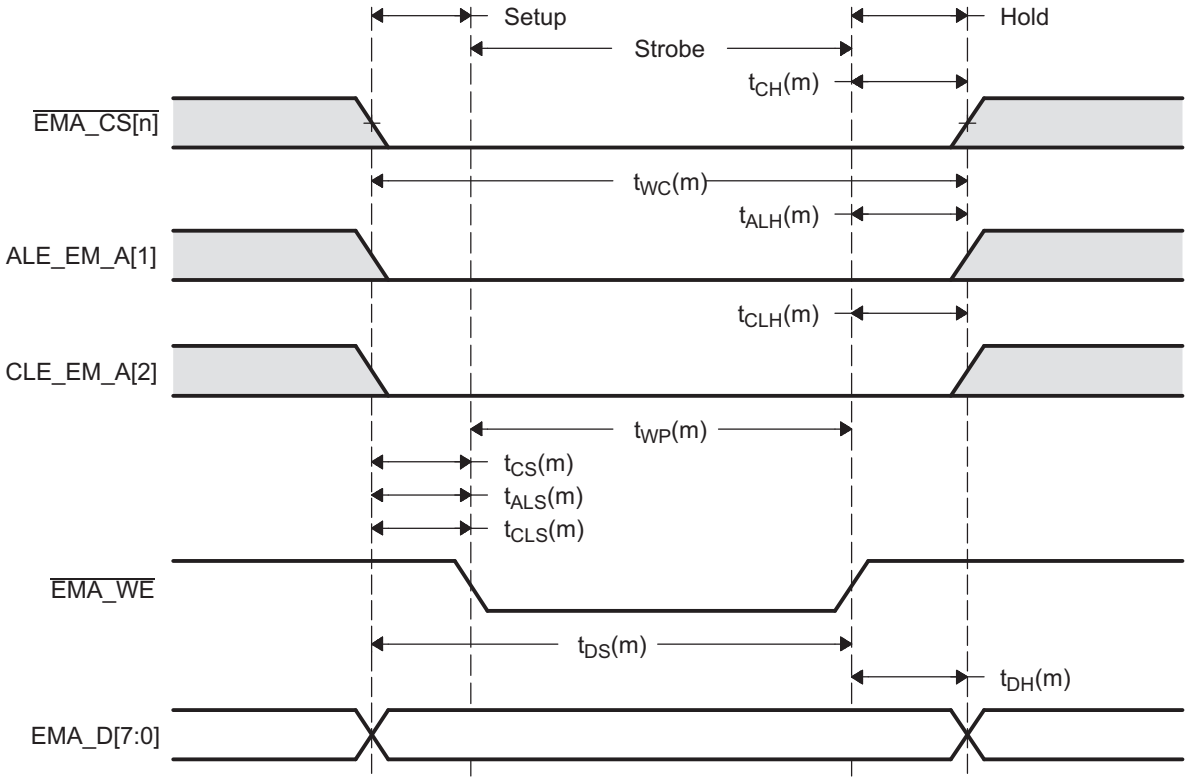


Figure 18-29. Timing Waveform of a NAND Flash Address Write



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 18-30. Timing Waveform of a NAND Flash Data Write



Copyright © 2013–2016, Texas Instruments Incorporated

18.3.2.3.3 Example Using Hynix HY27UA081G1M

This section takes you through the configuration steps required to implement Hynix's HY27UA081G1M NAND Flash with the EMIFA. The following assumptions are made:

- NAND Flash is connected to chip select space 2 ($\overline{\text{EMA_CS[2]}}$)
- EMIFA clock speed is 100 MHz (t_{cyc} = 10 nS)

Table 18-44 lists the data sheet specifications for the EMIFA and Table 18-45 lists the data sheet specifications for the NAND Flash.

Table 18-44. EMIFA Timing Requirements for HY27UA081G1M Example

Parameter	Description	Min	Max	Units
t_{SU}	Data Setup time, data valid before $\overline{\text{EMA_OE}}$ high	3 to 7 (1)		nS
t_H	Data Hold time, data valid after $\overline{\text{EMA_OE}}$ high	0		nS

(1) Depending on operating conditions. See your device-specific data manual for the value.

Table 18-45. NAND Flash Timing Requirements for HY27UA081G1M Example

Parameter	Description	Min	Max	Units
t_{RP}	Read Pulse width	60		nS
t_{REA}	Read Enable Access time		60	nS
t_{CEA}	Chip Enable low to output valid		75	nS
t_{CHZ}	Chip Enable high to output High-Z		20	nS
t_{RC}	Read Cycle time	80		nS
t_{RHZ}	Read Enable high to output High-Z		30	nS
t_{CLR}	Command Latch low to Read enable low	10		nS
t_{WP}	Write Pulse width	60		nS
t_{CLS}	CLE Setup time	0		nS
t_{ALS}	ALE Setup time	0		nS
t_{CS}	$\overline{\text{CS}}$ Setup time	0		nS
t_{DS}	Data Setup time	20		nS
t_{CLH}	CLE Hold time	10		nS
t_{ALH}	ALE Hold time	10		nS
t_{CH}	$\overline{\text{CS}}$ Hold time	10		nS
t_{DH}	Data Hold time	10		nS
t_{WC}	Write Cycle time	80		nS

Copyright © 2013–2016, Texas Instruments Incorporated

Inserting these values into the equations defined above allows you to determine the values for SETUP, STROBE, HOLD, and TA. For a read:

$$R_SETUP \geq \frac{t_{CLR}(m)}{t_{cyc}} - 1 \geq \left(\frac{10}{10}\right) - 1 \geq 0$$

$$R_STROBE \geq \max\left(\frac{\left(t_{REA}(m) + t_{SU}\right)}{t_{cyc}}, \frac{t_{RP}}{t_{cyc}}\right) - 1 \geq \left(\frac{65}{10}\right) - 1 \geq 5.5$$

$$R_SETUP + R_STROBE \geq \frac{\left(t_{CEA} + t_{SU}\right)}{t_{cyc}} - 2 \geq \frac{(75 + 5)}{10} - 2 \geq 6$$

$$R_HOLD \geq \frac{\left(t_H - t_{CHZ}(m)\right)}{t_{cyc}} - 1 \geq \frac{(0 - 20)}{10} - 1 \geq -3$$

$$R_SETUP + R_STROBE + R_HOLD \geq \frac{t_{RC}(m)}{t_{cyc}} - 3 \geq \left(\frac{80}{10}\right) - 3 \geq 5$$

Therefore with a 10 nS margin added in, R_SETUP ≥ 1.0, R_STROBE ≥ 6.5, and R_HOLD ≥ 0.

After solving for R_HOLD, TA may be calculated:

$$TA \geq \max\left(\frac{t_{CHZ}(m)}{t_{cyc}}, \frac{t_{RHZ}(m) - (R_HOLD + 1)t_{cyc}}{t_{cyc}}\right) - 1 \geq \left(\frac{20}{10}\right) - 1 \geq 1$$

Adding a 10 ns margin, TA ≥ 2.

For a write:

$$W_STROBE \geq \frac{t_{WP}(m)}{t_{cyc}} - 1 \geq \left(\frac{60}{10}\right) - 1 \geq 5$$

$$W_SETUP \geq \max\left(\frac{t_{CLS}(m)}{t_{cyc}}, \frac{t_{ALS}(m)}{t_{cyc}}, \frac{t_{CS}(m)}{t_{cyc}}\right) - 1 \geq \left(\frac{0}{10}\right) - 1 \geq -1$$

$$W_SETUP + W_STROBE \geq \frac{t_{DS}(m)}{t_{cyc}} - 2 \geq \frac{20}{10} - 2 \geq 0$$

$$W_HOLD \geq \max\left(\frac{t_{CLH}(m)}{t_{cyc}}, \frac{t_{ALH}(m)}{t_{cyc}}, \frac{t_{CH}(m)}{t_{cyc}}, \frac{t_{DH}(m)}{t_{cyc}}\right) - 1 \geq \left(\frac{10}{10}\right) - 1 \geq 0$$

$$W_SETUP + W_STROBE + W_HOLD \geq \frac{t_{WC}(m)}{t_{cyc}} - 3 \geq \left(\frac{80}{10}\right) - 3 \geq 5$$

Therefore with a 10 nS margin added in, W_SETUP ≥ 0, W_STROBE ≥ 6, and W_HOLD ≥ 1.

Copyright © 2013–2016, Texas Instruments Incorporated

Since the value of the W_SETUP/R_SETUP, W_STROBE/R_STROBE, W_HOLD/R_HOLD, and TA fields are equal to EMIFA clock cycles minus 1 cycle, the CE2CFG should be configured as in Table 18-46. In this example, although the EMA_WAIT signal is connected to the R/$\overline{B}$ signal of the NAND Flash the Extended Wait mode of the EMIFA is not used, therefore the asynchronous wait cycle configuration register (AWCC) does not need to be programmed.

Table 18-46. Configuring CE2CFG for HY27UA081G1M Example

Parameter	Setting
SS	**Select Strobe mode.** • SS = 0. Places EMIFA in Normal Mode.
EW	**Extended Wait mode enable.** • EW = 0. Disabled Extended wait mode.
W_SETUP/R_SETUP	**Read/Write setup widths.** • W_SETUP = 0 • R_SETUP = 2
W_STROBE/R_STROBE	**Read/Write strobe widths.** • W_STROBE = 6 • R_STROBE = 7
W_HOLD/R_HOLD	**Read/Write hold widths.** • W_HOLD = 1 • R_HOLD = 0
TA	**Minimum turnaround time.** • TA = 2
ASIZE	**Asynchronous device bus width.** • ASIZE = 0, select an 8-bit data bus width.

Since this is a NAND Flash example, the EMIFA must be configured for NAND Flash mode. This is accomplished by configuring the NAND Flash control register (NANDFCR) as in Table 18-47. In NANDFCR, chip select space 2 must be configured with NAND Flash mode enabled.

Table 18-47. Configuring NANDFCR for HY27UA081G1M Example

Parameter	Setting
CS5ECC	**NAND Flash ECC start for chip select 5.** • CS5ECC = 0. Not set during configuration. Only set just prior to reading or writing data.
CS4ECC	**NAND Flash ECC start for chip select 4.** • CS4ECC = 0. Not set during configuration. Only set just prior to reading or writing data.
CS3ECC	**NAND Flash ECC start for chip select 3.** • CS3ECC = 0. Not set during configuration. Only set just prior to reading or writing data.
CS2ECC	**NAND Flash ECC start for chip select 2.** • CS2ECC = 0. Not set during configuration. Only set just prior to reading or writing data.
CS5NAND	**NAND Flash mode for chip select 5.** • CS5NAND = 0. NAND Flash mode is disabled.
CS4NAND	**NAND Flash mode for chip select 4.** • CS4NAND = 0. NAND Flash mode is disabled.
CS3NAND	**NAND Flash mode for chip select 3.** • CS3NAND = 0. NAND Flash mode is disabled.
CS2NAND	**NAND Flash mode for chip select 2.** • CS5NAND = 1. NAND Flash mode is enabled.

Copyright © 2013–2016, Texas Instruments Incorporated

18.4 Registers

The external memory interface (EMIFA) is controlled by programming its internal memory-mapped registers (MMRs). Table 18-48 lists the memory-mapped registers for the EMIFA.

NOTE: All EMIFA MMRs, except SDCR, support only word (32-bit) accesses. Performing a byte (8-bit) or halfword (16-bit) write to these registers results in undefined behavior. The SDCR is byte writable to allow the setting of the SR, PD and PDWR bits without triggering the SDRAM initialization sequence.

The EMIFA registers must always be accessed using 32-bit accesses (unless otherwise specified in this chapter). For the base address of the memory-mapped registers of EMIFA, see your device-specific data manual.

Table 18-48. External Memory Interface (EMIFA) Registers

Offset	Acronym	Register Description	Section
0h	MIDR	Module ID Register	Section 18.4.1
4h	AWCC	Asynchronous Wait Cycle Configuration Register	Section 18.4.2
8h	SDCR	SDRAM Configuration Register	Section 18.4.3
Ch	SDRCR	SDRAM Refresh Control Register	Section 18.4.4
10h	CE2CFG	Asynchronous 1 Configuration Register	Section 18.4.5
14h	CE3CFG	Asynchronous 2 Configuration Register	Section 18.4.5
18h	CE4CFG	Asynchronous 3 Configuration Register	Section 18.4.5
1Ch	CE5CFG	Asynchronous 4 Configuration Register	Section 18.4.5
20h	SDTIMR	SDRAM Timing Register	Section 18.4.6
3Ch	SDSRETR	SDRAM Self Refresh Exit Timing Register	Section 18.4.7
40h	INTRAW	EMIFA Interrupt Raw Register	Section 18.4.8
44h	INTMSK	EMIFA Interrupt Mask Register	Section 18.4.9
48h	INTMSKSET	EMIFA Interrupt Mask Set Register	Section 18.4.10
4Ch	INTMSKCLR	EMIFA Interrupt Mask Clear Register	Section 18.4.11
60h	NANDFCR	NAND Flash Control Register	Section 18.4.12
64h	NANDFSR	NAND Flash Status Register	Section 18.4.13
70h	NANDF1ECC	NAND Flash 1 ECC Register (CS2 Space)	Section 18.4.14
74h	NANDF2ECC	NAND Flash 2 ECC Register (CS3 Space)	Section 18.4.14
78h	NANDF3ECC	NAND Flash 3 ECC Register (CS4 Space)	Section 18.4.14
7Ch	NANDF4ECC	NAND Flash 4 ECC Register (CS5 Space)	Section 18.4.14
BCh	NAND4BITECCLOAD	NAND Flash 4-Bit ECC Load Register	Section 18.4.15
C0h	NAND4BITECC1	NAND Flash 4-Bit ECC Register 1	Section 18.4.16
C4h	NAND4BITECC2	NAND Flash 4-Bit ECC Register 2	Section 18.4.17
C8h	NAND4BITECC3	NAND Flash 4-Bit ECC Register 3	Section 18.4.18
CCh	NAND4BITECC4	NAND Flash 4-Bit ECC Register 4	Section 18.4.19
D0h	NANDERRADD1	NAND Flash 4-Bit ECC Error Address Register 1	Section 18.4.20
D4h	NANDERRADD2	NAND Flash 4-Bit ECC Error Address Register 2	Section 18.4.21
D8h	NANDERRVAL1	NAND Flash 4-Bit ECC Error Value Register 1	Section 18.4.22
DCh	NANDERRVAL2	NAND Flash 4-Bit ECC Error Value Register 2	Section 18.4.23

Copyright © 2013–2016, Texas Instruments Incorporated

18.4.1 Module ID Register (MIDR)

This is a read-only register indicating the module ID of the EMIFA. The MIDR is shown in Figure 18-31 and described in Table 18-49.

Figure 18-31. Module ID Register (MIDR)

31	0
REV	
R-4000 0205h	

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

Table 18-49. Module ID Register (MIDR) Field Descriptions

Bit	Field	Value	Description
31-0	REV	4000 0205h	Module ID of EMIFA.

18.4.2 Asynchronous Wait Cycle Configuration Register (AWCC)

The asynchronous wait cycle configuration register (AWCC) is used to configure the parameters for extended wait cycles. Both the polarity of the EMA_WAIT pin(s) and the maximum allowable number of extended wait cycles can be configured. The AWCC is shown in Figure 18-32 and described in Table 18-50. Not all devices support both EMA_WAIT[1] and EMA_WAIT[0], see the device-specific data manual to determine support on each device.

NOTE: The EW bit in the asynchronous *n* configuration register (CE*n*CFG) must be set to allow for the insertion of extended wait cycles.

Figure 18-32. Asynchronous Wait Cycle Configuration Register (AWCCR)

31	30	29	28	27	24	23	22	21	20	19	18	17	16
Reserved		WP1	WP0	Reserved		CS5_WAIT		CS4_WAIT		CS3_WAIT		CS2_WAIT	
R-0		R/W-1	R/W-1	R-0		R/W-0		R/W-0		R/W-0		R/W-0	

15	8	7	0
Reserved		MAX_EXT_WAIT	
R-0		R/W-80h	

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

Copyright © 2013–2016, Texas Instruments Incorporated

Table 18-50. Asynchronous Wait Cycle Configuration Register (AWCCR) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Reserved
29	WP1		EMA_WAIT[1] polarity bit. This bit defines the polarity of the EMA_WAIT[1] pin.
		0	Insert wait cycles if EMA_WAIT[1] pin is low.
		1	Insert wait cycles if EMA_WAIT[1] pin is high.
28	WP0		EMA_WAIT[0] polarity bit. This bit defines the polarity of the EMA_WAIT[0] pin.
		0	Insert wait cycles if EMA_WAIT[0] pin is low.
		1	Insert wait cycles if EMA_WAIT[0] pin is high.
27-24	Reserved	0	Reserved
23-22	CS5_WAIT	0-3h	Chip Select 5 WAIT signal selection. This signal determines which EMA_WAIT[*n*] signal will be used for memory accesses to chip select 5 memory space.
		0	EMA_WAIT[0] pin is used to control external wait states.
		1h	EMA_WAIT[1] pin is used to control external wait states.
		2h-3h	Reserved
21-20	CS4_WAIT	0-3h	Chip Select 4 WAIT signal selection. This signal determines which EMA_WAIT[*n*] signal will be used for memory accesses to chip select 4 memory space.
		0	EMA_WAIT[0] pin is used to control external wait states.
		1h	EMA_WAIT[1] pin is used to control external wait states.
		2h-3h	Reserved
19-18	CS3_WAIT	0-3h	Chip Select 3 WAIT signal selection. This signal determines which EMA_WAIT[*n*] signal will be used for memory accesses to chip select 3 memory space.
		0	EMA_WAIT[0] pin is used to control external wait states.
		1h	EMA_WAIT[1] pin is used to control external wait states.
		2h-3h	Reserved
17-16	CS2_WAIT	0-3h	Chip Select 2 WAIT signal selection. This signal determines which EMA_WAIT[*n*] signal will be used for memory accesses to chip select 2 memory space.
		0	EMA_WAIT[0] pin is used to control external wait states..
		1h	EMA_WAIT[1] pin is used to control external wait states.
		2h-3h	Reserved
15-8	Reserved	0	Reserved
7-0	MAX_EXT_WAIT	0-FFh	Maximum extended wait cycles. The EMIFA will wait for a maximum of (MAX_EXT_WAIT + 1) × 16 clock cycles before it stops inserting asynchronous wait cycles and proceeds to the hold period of the access.

Copyright © 2013–2016, Texas Instruments Incorporated

18.4.3 SDRAM Configuration Register (SDCR)

The SDRAM configuration register (SDCR) is used to configure various parameters of the SDRAM controller such as the number of internal banks, the internal page size, and the CAS latency to match those of the attached SDRAM device. In addition, this register is used to put the attached SDRAM device into Self-Refresh mode. The SDCR is shown in Figure 18-33 and described in Table 18-51.

NOTE: Writing to the lower three bytes of this register will cause the EMIFA to start the SDRAM initialization sequence described in Section 18.2.4.4.

Figure 18-33. SDRAM Configuration Register (SDCR)

31	30	29	28	24
SR	PD	PDWR	Reserved	
R/W-0	R/W-0	R/W-0	R-0	

23	16
Reserved	
R-0	

15	14	13	12	11	9	8
Reserved	NM(A)	Reserved		CL		BIT11_9LOCK
R-0	R/W-0	R-0		R/W-3h		R/W-0

7	6	4	3	2	0
Reserved	IBANK		Reserved	PAGESIZE	
R-0	R/W-2h		R-0	R/W-0	

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

A. The NM bit must be set to 1 if the EMIFA on your device only has 16 data bus pins.

Table 18-51. SDRAM Configuration Register (SDCR) Field Descriptions

Bit	Field	Value	Description
31	SR		Self-Refresh mode bit. This bit controls entering and exiting of the Self-Refresh mode described in Section 18.2.4.7. The field should be written using a byte-write to the upper byte of SDCR to avoid triggering the SDRAM initialization sequence.
		0	Writing a 0 to this bit will cause connected SDRAM devices and the EMIFA to exit the Self-Refresh mode.
		1	Writing a 1 to this bit will cause connected SDRAM devices and the EMIFA to enter the Self-Refresh mode.
30	PD		Power Down bit. This bit controls entering and exiting of the power-down mode. The field should be written using a byte-write to the upper byte of SDCR to avoid triggering the SDRAM initialization sequence. If both SR and PD bits are set, the EMIFA will go into Self Refresh.
		0	Writing a 0 to this bit will cause connected SDRAM devices and the EMIFA to exit the power-down mode.
		1	Writing a 1 to this bit will cause connected SDRAM devices and the EMIFA to enter the power-down mode.
29	PDWR		Perform refreshes during power down. Writing a 1 to this bit will cause EMIFA to exit power-down state and issue and AUTO REFRESH command every time Refresh May level is set.
28-15	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
14	NM		Narrow mode bit. This bit defines whether a 16- or 32-bit-wide SDRAM is connected to the EMIFA. This bit field must always be set to 1. Writing to this field triggers the SDRAM initialization sequence.
		0	32-bit SDRAM data bus is used.
		1	16-bit SDRAM data bus is used.
13-12	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 18-51. SDRAM Configuration Register (SDCR) Field Descriptions (continued)

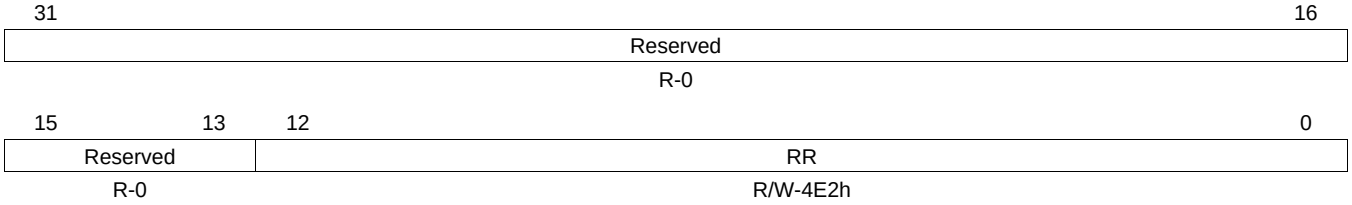
Bit	Field	Value	Description
11-9	CL	0-7h	CAS Latency. This field defines the CAS latency to be used when accessing connected SDRAM devices. A 1 must be simultaneously written to the BIT11_9LOCK bit field of this register in order to write to the CL bit field. Writing to this field triggers the SDRAM initialization sequence.
		0-1h	Reserved
		2h	CAS latency = 2 EMA_CLK cycles
		3h	CAS latency = 3 EMA_CLK cycles
		4h-7h	Reserved
8	BIT11_9LOCK		Bits 11 to 9 lock. CL can only be written if BIT11_9LOCK is simultaneously written with a 1. BIT11_9LOCK is always read as 0. Writing to this field triggers the SDRAM initialization sequence.
		0	CL cannot be written.
		1	CL can be written.
7	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
6-4	IBANK	0-7h	Internal SDRAM Bank size. This field defines number of banks inside the connected SDRAM devices. Writing to this field triggers the SDRAM initialization sequence.
		0	1 bank SDRAM devices.
		1	2 bank SDRAM devices.
		2	4 bank SDRAM devices.
		3h-7h	Reserved.
3	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
2-0	PAGESIZE	0-7h	Page Size. This field defines the internal page size of connected SDRAM devices. Writing to this field triggers the SDRAM initialization sequence.
		0	8 column address bits (256 elements per row)
		1h	9 column address bits (512 elements per row)
		2h	10 column address bits (1024 elements per row)
		3h	11 column address bits (2048 elements per row)
		4h-7h	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

18.4.4 SDRAM Refresh Control Register (SDRCR)

The SDRAM refresh control register (SDRCR) is used to configure the rate at which connected SDRAM devices will be automatically refreshed by the EMIFA. Refer to Section 18.2.4.6 on the refresh controller for more details. The SDRCR is shown in Figure 18-34 and described in Table 18-52.

Figure 18-34. SDRAM Refresh Control Register (SDRCR)



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

Table 18-52. SDRAM Refresh Control Register (SDRCR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
12-0	RR	0-1FFFh	Refresh Rate. This field is used to define the SDRAM refresh period in terms of EMA_CLK cycles. Writing a value < 0x0020 to this field will cause it to be loaded with (2 × T_RFC) + 1 value from the SDRAM timing register (SDTIMR).

Copyright © 2013–2016, Texas Instruments Incorporated

18.4.5 Asynchronous *n* Configuration Registers (CE2CFG-CE5CFG)

The asynchronous *n* configuration registers (CE2CFG, CE3CFG, CE4CFG, and CE5CFG) are used to configure the shaping of the address and control signals during an access to asynchronous memory connected to CS2, CS3, CS4, and CS5, respectively. It is also used to program the width of asynchronous interface and to select from various modes of operation. This register can be written prior to any transfer, and any asynchronous transfer following the write will use the new configuration. The CE*n*CFG is shown in Figure 18-35 and described in Table 18-53.

Figure 18-35. Asynchronous *n* Configuration Register (CE*n*CFG)

31	30	29 – 26	25 – 24
SS	EW(A)	W_SETUP	W_STROBE(B)
R/W-0	R/W-0	R/W-Fh	R/W-3Fh

23 – 20	19 – 17	16
W_STROBE(B)	W_HOLD	R_SETUP
R/W-3Fh	R/W-7h	R/W-Fh

15 – 13	12 – 7	6 – 4	3 – 2	1 – 0
R_SETUP	R_STROBE(B)	R_HOLD	TA	ASIZE
R/W-Fh	R/W-3Fh	R/W-7h	R/W-3h	R/W-0

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

A. The EW bit must be cleared to 0 when operating in NAND Flash mode.

B. This bit field must be cleared to 0 if the EMIFA on your device does not have an EMA_WAIT pin.

Table 18-53. Asynchronous *n* Configuration Register (CE*n*CFG) Field Descriptions

Bit	Field	Value	Description
31	SS		Select Strobe bit. This bit defines whether the asynchronous interface operates in Normal Mode or Select Strobe Mode. See Section 18.2.5 for details on the two modes of operation.
		0	Normal Mode enabled.
		1	Select Strobe Mode enabled.
30	EW		Extend Wait bit. This bit defines whether extended wait cycles will be enabled. See Section 18.2.5.7 on extended wait cycles for details. This bit field must be cleared to 0, if the EMIFA on your device does not have an EMA_WAIT pin. The CS*n*_WAIT bit in the asynchronous wait cycle configuration register (AWCC) must also be configured to determine which EMA_WAIT pin is used for memory accesses.
		0	Extended wait cycles disabled.
		1	Extended wait cycles enabled.
29-26	W_SETUP	0-Fh	Write setup width in the format n - 1, where n = number of EMA_CLK cycles. See Section 18.2.5.3 for details. 0h = Divide-by-1 1h = Divide-by-2 … 2h – Fh = Divide-by-3 to Divide-by-16
25-20	W_STROBE	0-3Fh	Write strobe width in the format n - 1, where n = number of EMA_CLK cycles. See Section 18.2.5.3 for details. 0h = Divide-by-1 1h = Divide-by-2 … 2h – 3Fh = Divide-by-3 to Divide-by-64
19-17	W_HOLD	0-7h	Write hold width in the format n - 1, where n = number of EMA_CLK cycles. See Section 18.2.5.3 for details. 0h = Divide-by-1 1h = Divide-by-2 … 2h – 7h = Divide-by-3 to Divide-by-8
16-13	R_SETUP	0-Fh	Read setup width in the format n - 1, where n = number of EMA_CLK cycles. See Section 18.2.5.3 for details. 0h = Divide-by-1 1h = Divide-by-2 … 2h – 1Fh = Divide-by-3 to Divide-by-16

Copyright © 2013–2016, Texas Instruments Incorporated

Table 18-53. Asynchronous *n* Configuration Register (CE*n*CFG) Field Descriptions (continued)

Bit	Field	Value	Description
12-7	R_STROBE	0-3Fh	Read strobe width in the format n - 1, where n = number of EMA_CLK cycles. See Section 18.2.5.3 for details. 0h = Divide-by-1 1h = Divide-by-2 … 2h – 3Fh = Divide-by-3 to Divide-by-64
6-4	R_HOLD	0-7h	Read hold width in the format n - 1, where n = number of EMA_CLK cycles. See Section 18.2.5.3 for details. 0h = Divide-by-1 1h = Divide-by-2 … 2h – 7h = Divide-by-3 to Divide-by-8
3-2	TA	0-3h	Minimum Turn-Around time. This field defines the minimum number of EMA_CLK cycles between reads and writes, minus one cycle. See Section 18.2.5.3 for details.
1-0	ASIZE	0-3h	Asynchronous Data Bus Width. This field defines the width of the asynchronous device's data bus.
		0	8-bit data bus
		1h	16-bit data bus
		2h-3h	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

18.4.6 SDRAM Timing Register (SDTIMR)

The SDRAM timing register (SDTIMR) is used to program many of the SDRAM timing parameters. Consult the SDRAM datasheet for information on the appropriate values to program into each field. The SDTIMR is shown in Figure 18-36 and described in Table 18-54.

Figure 18-36. SDRAM Timing Register (SDTIMR)

31 27	26 24	23	22 20	19	18 16
T_RFC	T_RP	Rsvd	T_RCD	Rsvd	T_WR
R/W-8h	R/W-2h	R-0	R/W-2h	R-0	R/W-1h

15 12	11 8	7	6 4	3 0
T_RAS	T_RC	Rsvd	T_RRD	Reserved
R/W-5h	R/W-8h	R-0	R/W-1h	R-0

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

Table 18-54. SDRAM Timing Register (SDTIMR) Field Descriptions

Bit	Field	Value	Description
31-27	T_RFC	0-1Fh	Specifies the Trfc value of the SDRAM. This defines the minimum number of EMA_CLK cycles from Refresh (REFR) to Refresh (REFR), minus 1: T_RFC = (Trfc/t_{EMA_CLK}) - 1
26-24	T_RP	0-7h	Specifies the Trp value of the SDRAM. This defines the minimum number of EMA_CLK cycles from Precharge (PRE) to Activate (ACTV) or Refresh (REFR) command, minus 1: T_RP = (Trp/t_{EMA_CLK}) - 1
23	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
22-20	T_RCD	0-7h	Specifies the Trcd value of the SDRAM. This defines the minimum number of EMA_CLK cycles from Active (ACTV) to Read (READ) or Write (WRT), minus 1: T_RCD = (Trcd/t_{EMA_CLK}) - 1
19	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
18-16	T_WR	0-7h	Specifies the Twr value of the SDRAM. This defines the minimum number of EMA_CLK cycles from last Write (WRT) to Precharge (PRE), minus 1: T_WR = (Twr/t_{EMA_CLK}) - 1
15-12	T_RAS	0-Fh	Specifies the Tras value of the SDRAM. This defines the minimum number of EMA_CLK clock cycles from Activate (ACTV) to Precharge (PRE), minus 1: T_RAS = (Tras/t_{EMA_CLK}) - 1
11-8	T_RC	0-Fh	Specifies the Trc value of the SDRAM. This defines the minimum number of EMA_CLK clock cycles from Activate (ACTV) to Activate (ACTV), minus 1: T_RC = (Trc/t_{EMA_CLK}) - 1
7	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
6-4	T_RRD	0-7h	Specifies the Trrd value of the SDRAM. This defines the minimum number of EMA_CLK clock cycles from Activate (ACTV) to Activate (ACTV) for a different bank, minus 1: T_RRD = (Trrd/t_{EMA_CLK}) - 1
3-0	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.

Copyright © 2013–2016, Texas Instruments Incorporated

18.4.7 SDRAM Self Refresh Exit Timing Register (SDSRETR)

The SDRAM self refresh exit timing register (SDSRETR) is used to program the amount of time between when the SDRAM exits Self-Refresh mode and when the EMIFA issues another command. The SDSRETR is shown in Figure 18-37 and described in Table 18-55.

Figure 18-37. SDRAM Self Refresh Exit Timing Register (SDSRETR)

31	16
Reserved	
R-0	

15	5	4	0
Reserved		T_XS	
R-0		R/W-9h	

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

Table 18-55. SDRAM Self Refresh Exit Timing Register (SDSRETR) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved. The reserved bit location is always read as 0.
4-0	T_XS	0-1Fh	This field specifies the minimum number of ECLKOUT cycles from Self-Refresh exit to any command, minus one. T_XS = Txsr / t_{EMA_CLK} - 1

Copyright © 2013–2016, Texas Instruments Incorporated

18.4.8 EMIFA Interrupt Raw Register (INTRAW)

The EMIFA interrupt raw register (INTRAW) is used to monitor and clear the EMIFA's hardware-generated Asynchronous Timeout Interrupt. The AT bit in this register will be set when an Asynchronous Timeout occurs regardless of the status of the EMIFA interrupt mask set register (INTMSKSET) and EMIFA interrupt mask clear register (INTMSKCLR). Writing a 1 to this bit will clear it. The EMIFA on some devices does not have the EMA_WAIT pin; therefore, these registers and fields are reserved on those devices. The INTRAW is shown in Figure 18-38 and described in Table 18-56.

Figure 18-38. EMIFA Interrupt Raw Register (INTRAW)

31	8
Reserved	
R-0	

7	3	2	1	0
Reserved		WR	LT	AT
R-0		R/W1C-0	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing 0 has no effect); -*n* = value after reset

Table 18-56. EMIFA Interrupt Raw Register (INTRAW) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
2	WR		Wait Rise. This bit is set to 1 by hardware to indicate that a rising edge on the EMA_WAIT pin has occurred.
		0	Indicates that a rising edge has not occurred on the EMA_WAIT pin. Writing a 0 has no effect.
		1	Indicates that a rising edge has occurred on the EMA_WAIT pin. Writing a 1 will clear this bit and the WR_MASKED bit in the EMIFA interrupt masked register (INTMSK).
1	LT		Line Trap. Set to 1 by hardware to indicate illegal memory access type or invalid cache line size.
		0	Writing a 0 has no effect.
		1	Indicates that a line trap has occurred. Writing a 1 will clear this bit as well as the LT_MASKED bit in the EMIFA interrupt masked register(INTMSK).
0	AT		Asynchronous Timeout. This bit is set to 1 by hardware to indicate that during an extended asynchronous memory access cycle, the EMA_WAIT pin did not go inactive within the number of cycles defined by the MAX_EXT_WAIT field in the asynchronous wait cycle configuration register (AWCC).
		0	Indicates that an Asynchronous Timeout has not occurred. Writing a 0 has no effect.
		1	Indicates that an Asynchronous Timeout has occurred. Writing a 1 will clear this bit as well as the AT_MASKED bit in the EMIFA interrupt masked register (INTMSK).

Copyright © 2013–2016, Texas Instruments Incorporated

18.4.9 EMIFA Interrupt Masked Register (INTMSK)

Like the EMIFA interrupt raw register (INTRAW), the EMIFA interrupt masked register (INTMSK) is used to monitor and clear the status of the EMIFA's hardware-generated Asynchronous Timeout Interrupt. The main difference between the two registers is that when the AT_MASKED bit in this register is set, an active-high pulse will be sent to the CPU interrupt controller. Also, the AT_MASKED bit field in INTMSK is only set to 1 if the associated interrupt has been enabled in the EMIFA interrupt mask set register (INTMSKSET). The EMIFA on some devices does not have the EMA_WAIT pin, therefore, these registers and fields are reserved on those devices. The INTMSK is shown in Figure 18-39 and described in Table 18-57.

Figure 18-39. EMIFA Interrupt Mask Register (INTMSK)

31	8
Reserved	
R-0	

7	3	2	1	0
Reserved		WR_MASKED	LT_MASKED	AT_MASKED
R-0		R/W1C-0	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing 0 has no effect); -*n* = value after reset

Table 18-57. EMIFA Interrupt Mask Register (INTMSK) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
2	WR_MASKED		Wait Rise Masked. This bit is set to 1 by hardware to indicate a rising edge has occurred on the EMA_WAIT pin, provided that the WR_MASK_SET bit is set to 1 in the EMIFA interrupt mask set register (INTMSKSET).
		0	Indicates that a wait rise interrupt has not been generated. Writing a 0 has no effect.
		1	Indicates that a wait rise interrupt has been generated. Writing a 1 will clear this bit and the WR bit in the EMIFA interrupt raw register (INTRAW).
1	LT_MASKED		Masked Line Trap. Set to 1 by hardware to indicate illegal memory access type or invalid cache line size, only if the LT_MASK_SET bit in the EMIFA interrupt mask set register (INTMSKSET) is set to 1.
		0	Writing a 0 has no effect.
		1	Writing a 1 will clear this bit as well as the LT bit in the EMIFA interrupt raw register(INTRAW).
0	AT_MASKED		Asynchronous Timeout Masked. This bit is set to 1 by hardware to indicate that during an extended asynchronous memory access cycle, the EMA_WAIT pin did not go inactive within the number of cycles defined by the MAX_EXT_WAIT field in the asynchronous wait cycle configuration register (AWCC), provided that the AT_MASK_SET bit is set to 1 in the EMIFA interrupt mask set register (INTMSKSET).
		0	Indicates that an Asynchronous Timeout Interrupt has not been generated. Writing a 0 has no effect.
		1	Indicates that an Asynchronous Timeout Interrupt has been generated. Writing a 1 will clear this bit as well as the AT bit in the EMIFA interrupt raw register (INTRAW).

Copyright © 2013–2016, Texas Instruments Incorporated

18.4.10 EMIFA Interrupt Mask Set Register (INTMSKSET)

The EMIFA interrupt mask set register (INTMSKSET) is used to enable the Asynchronous Timeout Interrupt. If read as 1, the AT_MASKED bit in the EMIFA interrupt masked register (INTMSK) will be set and an interrupt will be generated when an Asynchronous Timeout occurs. If read as 0, the AT_MASKED bit will always read 0 and no interrupt will be generated when an Asynchronous Timeout occurs. Writing a 1 to the AT_MASK_SET bit enables the Asynchronous Timeout Interrupt. The EMIFA on some devices does not have the EMA_WAIT pin; therefore, these registers and fields are reserved on those devices. The INTMSKSET is shown in Figure 18-40 and described in Table 18-58.

Figure 18-40. EMIFA Interrupt Mask Set Register (INTMSKSET)

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		WR_MASK_SET	Reserved	AT_MASK_SET
R-0		R/W-0	R-0	R/W-0

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

Table 18-58. EMIFA Interrupt Mask Set Register (INTMSKSET) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
2	WR_MASK_SET		Wait Rise Mask Set. This bit determines whether or not the wait rise Interrupt is enabled. Writing a 1 to this bit sets this bit, sets the WR_MASK_CLR bit in the EMIFA interrupt mask clear register (INTMSKCLR), and enables the wait rise interrupt. To clear this bit, a 1 must be written to the WR_MASK_CLR bit in INTMSKCLR.
		0	Indicates that the wait rise interrupt is disabled. Writing a 0 has no effect.
		1	Indicates that the wait rise interrupt is enabled. Writing a 1 sets this bit and the WR_MASK_CLR bit in the EMIFA interrupt mask clear register (INTMSKCLR).
1	LT_MASK_SET		Mask set for LT_MASKED bit in the EMIFA interrupt mask register (INTMSK).
		0	Indicates that the line trap interrupt is disabled. Writing a 0 has no effect.
		1	Indicates that the line trap interrupt is enabled. Writing a 1 sets this bit and the LT_MASK_CLR bit in the EMIFA interrupt mask clear register (INTMSKCLR).
0	AT_MASK_SET		Asynchronous Timeout Mask Set. This bit determines whether or not the Asynchronous Timeout Interrupt is enabled. Writing a 1 to this bit sets this bit, sets the AT_MASK_CLR bit in the EMIFA interrupt mask clear register (INTMSKCLR), and enables the Asynchronous Timeout Interrupt. To clear this bit, a 1 must be written to the AT_MASK_CLR bit of the EMIFA interrupt mask clear register (INTMSKCLR).
		0	Indicates that the Asynchronous Timeout Interrupt is disabled. Writing a 0 has no effect.
		1	Indicates that the Asynchronous Timeout Interrupt is enabled. Writing a 1 sets this bit and the AT_MASK_CLR bit in the EMIFA interrupt mask clear register (INTMSKCLR).

Copyright © 2013–2016, Texas Instruments Incorporated

18.4.11 EMIFA Interrupt Mask Clear Register (INTMSKCLR)

The EMIFA interrupt mask clear register (INTMSKCLR) is used to disable the Asynchronous Timeout Interrupt. If read as 1, the AT_MASKED bit in the EMIFA interrupt masked register (INTMSK) will be set and an interrupt will be generated when an Asynchronous Timeout occurs. If read as 0, the AT_MASKED bit will always read 0 and no interrupt will be generated when an Asynchronous Timeout occurs. Writing a 1 to the AT_MASK_CLR bit disables the Asynchronous Timeout Interrupt. The EMIFA on some devices does not have the EMA_WAIT pin, therefore, these registers and fields are reserved on those devices. The INTMSKCLR is shown in Figure 18-41 and described in Table 18-59.

Figure 18-41. EMIFA Interrupt Mask Clear Register (INTMSKCLR)

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		WR_MASK_CLR	Reserved	AT_MASK_CLR
R-0		R/W-0	R-0	R/W-0

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

Table 18-59. EMIFA Interrupt Mask Clear Register (INTMSKCLR) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved. The reserved bit location is always read as 0. If writing to this field, always write the default value of 0.
2	WR_MASK_CLR		Wait Rise Mask Clear. This bit determines whether or not the wait rise interrupt is enabled. Writing a 1 to this bit clears this bit, clears the WR_MASK_SET bit in the EMIFA interrupt mask set register (INTMSKSET), and disables the wait rise interrupt. To set this bit, a 1 must be written to the WR_MASK_SET bit in INTMSKSET.
		0	Indicates that the wait rise interrupt is disabled. Writing a 0 has no effect.
		1	Indicates that the wait rise interrupt is enabled. Writing a 1 clears this bit and the WR_MASK_SET bit in the EMIFA interrupt mask set register (INTMSKSET).
1	LT_MASK_CLR		Line trap Mask Clear. This bit determines whether or not the line trap interrupt is enabled. Writing a 1 to this bit clears this bit, clears the LT_MASK_SET bit in the EMIFA interrupt mask set register (INTMSKSET), and disables the line trap interrupt. To set this bit, a 1 must be written to the LT_MASK_SET bit in INTMSKSET.
		0	Indicates that the line trap interrupt is disabled. Writing a 0 has no effect.
		1	Indicates that the line trap interrupt is enabled. Writing a 1 clears this bit and the LT_MASK_SET bit in the EMIFA interrupt mask set register (INTMSKSET).
0	AT_MASK_CLR		Asynchronous Timeout Mask Clear. This bit determines whether or not the Asynchronous Timeout Interrupt is enabled. Writing a 1 to this bit clears this bit, clears the AT_MASK_SET bit in the EMIFA interrupt mask set register (INTMSKSET), and disables the Asynchronous Timeout Interrupt. To set this bit, a 1 must be written to the AT_MASK_SET bit of the EMIFA interrupt mask set register (INTMSKSET).
		0	Indicates that the Asynchronous Timeout Interrupt is disabled. Writing a 0 has no effect.
		1	Indicates that the Asynchronous Timeout Interrupt is enabled. Writing a 1 clears this bit and the AT_MASK_SET bit in the EMIFA interrupt mask set register (INTMSKSET).

Copyright © 2013–2016, Texas Instruments Incorporated

18.4.12 NAND Flash Control Register (NANDFCR)

The NAND Flash control register (NANDFCR) is shown in Figure 18-42 and described in Table 18-60.

Figure 18-42. NAND Flash Control Register (NANDFCR)

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		4BITECC_ADD_CALC_START	4BITECC_START	CS5ECC	CS4ECC	CS3ECC	CS2ECC
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
Reserved		4BITECCSEL		CS5NAND	CS4NAND	CS3NAND	CS2NAND
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 18-60. NAND Flash Control Register (NANDFCR) Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Reserved
13	4BITECC_ADD_CALC_START		NAND Flash 4-bit ECC address and error value calculation Start. Set to 1 to start 4_bit ECC error address and error value calculation on read syndrome. This bit is cleared when any of the NAND Flash error address registers or NAND Flash error value registers are read.
		1	start 4_bit ECC error address and error value calculation on read syndrome.
12	4BITECC_START		Nand Flash 4-bit ECC start for the selected chip select. Set to 1 to start 4_bit ECC calculation on data for NAND Flash on chip select selected by bit 4BITECCSEL. This bit is cleared when ay of the NAND Flash 4_bit ECC registers are read.
		1	start 4_bit ECC calculation on data for NAND Flash on chip select selected by bit 4BITECCSEL.
11	CS5ECC		NAND Flash ECC start for chip select 5. Set to 1 to start 1_bit ECC calculation on data for NAND Flash for this chip select. This bit is cleared when CS5 1_bit ECC register is read.
		0	Do not start ECC calculation.
		1	Start ECC calculation on data for NAND Flash on $\overline{\text{EMA_CS5}}$.
10	CS4ECC		NAND Flash ECC start for chip select 4. Set to 1 to start 1_bit ECC calculation on data for NAND Flash for this chip select. This bit is cleared when CS4 1_bit ECC register is read.
		0	Do not start ECC calculation.
		1	Start ECC calculation on data for NAND Flash on $\overline{\text{EMA_CS4}}$.
9	CS3ECC		NAND Flash ECC start for chip select 3. Set to 1 to start 1_bit ECC calculation on data for NAND Flash for this chip select. This bit is cleared when CS3 1_bit ECC register is read.
		0	Do not start ECC calculation.
		1	Start ECC calculation on data for NAND Flash on $\overline{\text{EMA_CS3}}$.
8	CS2ECC		NAND Flash ECC start for chip select 2. This bit is cleared when CS2 1_bit ECC register is read.
		0	Do not start ECC calculation.
		1	Start ECC calculation on data for NAND Flash on $\overline{\text{EMA_CS2}}$.
7-6	Reserved	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

Table 18-60. NAND Flash Control Register (NANDFCR) Field Descriptions (continued)

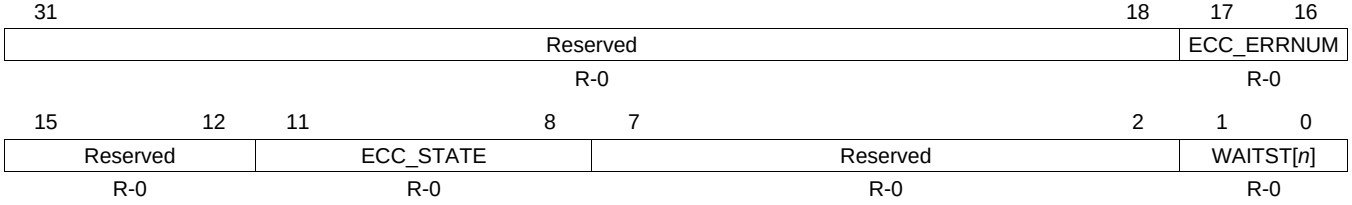
Bit	Field	Value	Description
5-4	4BITECCSEL	0-3h	4-bit ECC selection. This field selects the chip select on which 4-bit ECC will be calculated.
		0	ECC will be calculated for CS2.
		1h	ECC will be calculated for CS3.
		2h	ECC will be calculated for CS4.
		3h	ECC will be calculated for CS5.
3	CS5NAND		NAND Flash mode for chip select 5.
		0	Not using NAND Flash.
		1	Using NAND Flash on $\overline{\text{EMA_CS5}}$.
2	CS4NAND		NAND Flash mode for chip select 4.
		0	Not using NAND Flash.
		1	Using NAND Flash on $\overline{\text{EMA_CS4}}$.
1	CS3NAND		NAND Flash mode for chip select 3.
		0	Not using NAND Flash.
		1	Using NAND Flash on $\overline{\text{EMA_CS3}}$.
0	CS2NAND		NAND Flash mode for chip select 2.
		0	Not using NAND Flash.
		1	Using NAND Flash on $\overline{\text{EMA_CS2}}$.

Copyright © 2013–2016, Texas Instruments Incorporated

18.4.13 NAND Flash Status Register (NANDFSR)

The NAND Flash status register (NANDFSR) is shown in Figure 18-43 and described in Table 18-61.

Figure 18-43. NAND Flash Status Register (NANDFSR)



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

Table 18-61. NAND Flash Status Register (NANDFSR) Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	ECC_ERRNUM	0-3h	Number of Errors found after the 4-Bit ECC Error Address and Error Value Calculation.
		0	1 error found.
		1h	2 errors found.
		2h	3 errors found.
		3h	4 errors found.
15-12	Reserved	0	Reserved.
11-8	ECC_STATE	0-Fh	ECC correction state while performing 4-bit ECC Address and Error Value Calculation
		0	No errors detected
		1h	Errors cannot be corrected (5 or more)
		2h	Error correction complete(errors on bit 8 or 9).
		3h	Error correction complete(error exists).
		4h	Reserved.
		5h	Calculating number of errors
		6h-7h	Preparing for error search
		8h	Searching for errors
		9h-Bh	Reserved.
		Ch-Fh	Calculating error value
7-2	Reserved	0	Reserved.
1-0	WAITST[*n*]		Status of the EMA_WAIT[*n*] input pins. Not all devices support both EMA_WAIT[1] and EMA_WAIT[0], see the device-specific data manual to determine support on each device. The WP*n* bit in the asynchronous wait cycle configuration register (AWCC) has no effect on WAITST.
		0	EMA_WAIT[*n*] pin is low.
		1	EMA_WAIT[*n*] pin is high.

Copyright © 2013–2016, Texas Instruments Incorporated

18.4.14 NAND Flash *n* ECC Registers (NANDF1ECC-NANDF4ECC)

The NAND Flash *n* ECC register (NANDF*n*ECC) is shown in Figure 18-44 and described in Table 18-62. For 8-bit NAND Flash, the P1 to P4 bits are column parities; the P8 to P2048 bits are row parities. For 16-bit NAND Flash, the P1 to P8 bits are column parities; the P16 to P2048 bits are row parities.

Figure 18-44. NAND Flash *n* ECC Register (NANDF*n*ECC)

31			28	27	26	25	24
Reserved				P2048O	P1024O	P512O	P256O
R-0				R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
P128O	P64O	P32O	P16O	P8O	P4O	P2O	P1O
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15			12	11	10	9	8
Reserved				P2048E	P1024E	P512E	P256E
R-0				R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
P128E	P64E	P32E	P16E	P8E	P4E	P2E	P1E
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

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

Table 18-62. NAND Flash *n* ECC Register (NANDF*n*ECC) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reserved
27	P2048O	0-1	ECC code calculated while reading/writing NAND Flash.
26	P1024O	0-1	ECC code calculated while reading/writing NAND Flash.
25	P512O	0-1	ECC code calculated while reading/writing NAND Flash.
24	P256O	0-1	ECC code calculated while reading/writing NAND Flash.
23	P128O	0-1	ECC code calculated while reading/writing NAND Flash.
22	P64O	0-1	ECC code calculated while reading/writing NAND Flash.
21	P32O	0-1	ECC code calculated while reading/writing NAND Flash.
20	P16O	0-1	ECC code calculated while reading/writing NAND Flash.
19	P8O	0-1	ECC code calculated while reading/writing NAND Flash.
18	P4O	0-1	ECC code calculated while reading/writing NAND Flash.
17	P2O	0-1	ECC code calculated while reading/writing NAND Flash.
16	P1O	0-1	ECC code calculated while reading/writing NAND Flash.
15-12	Reserved	0	Reserved
11	P2948E	0-1	ECC code calculated while reading/writing NAND Flash.
10	P102E	0-1	ECC code calculated while reading/writing NAND Flash.
9	P512E	0-1	ECC code calculated while reading/writing NAND Flash.
8	P256E	0-1	ECC code calculated while reading/writing NAND Flash.
7	P128E	0-1	ECC code calculated while reading/writing NAND Flash.
6	P64E	0-1	ECC code calculated while reading/writing NAND Flash.
5	P32E	0-1	ECC code calculated while reading/writing NAND Flash.
4	P15E	0-1	ECC code calculated while reading/writing NAND Flash.
3	P8E	0-1	ECC code calculated while reading/writing NAND Flash.
2	P4E	0-1	ECC code calculated while reading/writing NAND Flash.
1	P2E	0-1	ECC code calculated while reading/writing NAND Flash.
0	P1E	0-1	ECC code calculated while reading/writing NAND Flash.

Copyright © 2013–2016, Texas Instruments Incorporated

18.4.15 NAND Flash 4-Bit ECC LOAD Register (NAND4BITECCLOAD)

The NAND Flash 4-bit ECC load register (NAND4BITECCLOAD) is shown in Figure 18-45 and described in Table 18-63.

Figure 18-45. NAND Flash 4-Bit ECC LOAD Register (NAND4BITECCLOAD)

31	16
Reserved	
R-0	

15	10	9	0
Reserved		4BITECCLOAD	
R-0		R/W-0	

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

Table 18-63. NAND Flash 4-Bit ECC LOAD Register (NAND4BITECCLOAD) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9-0	4BITECCLOAD	0-3FFh	4-bit ECC load. This value is used to load the ECC values when performing the Syndrome calculation during reads.

Copyright © 2013–2016, Texas Instruments Incorporated

18.4.16 NAND Flash 4-Bit ECC Register 1 (NAND4BITECC1)

The NAND Flash 4-bit ECC register 1 (NAND4BITECC1) is shown in Figure 18-46 and described in Table 18-64.

Figure 18-46. NAND Flash 4-Bit ECC Register 1 (NAND4BITECC1)

31	26	25	16
Reserved		4BITECCVAL2	
R-0		R/W-0	

15	10	9	0
Reserved		4BITECCVAL1	
R-0		R/W-0	

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

Table 18-64. NAND Flash 4-Bit ECC Register 1 (NAND4BITECC1) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	Reserved
25-16	4BITECCVAL2	0-3FFh	Calculated 4-bit ECC or Syndrom Value2.
15-10	Reserved	0	Reserved
9-0	4BITECCVAL1	0-3FFh	Calculated 4-bit ECC or Syndrom Value1.

18.4.17 NAND Flash 4-Bit ECC Register 2 (NAND4BITECC2)

The NAND Flash 4-bit ECC register 2 (NAND4BITECC2) is shown in Figure 18-47 and described in Table 18-65.

Figure 18-47. NAND Flash 4-Bit ECC Register 2 (NAND4BITECC2)

31	26	25	16
Reserved		4BITECCVAL4	
R-0		R/W-0	

15	10	9	0
Reserved		4BITECCVAL3	
R-0		R/W-0	

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

Table 18-65. NAND Flash 4-Bit ECC Register 2 (NAND4BITECC2) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	Reserved
25-16	4BITECCVAL4	0-3FFh	Calculated 4-bit ECC or Syndrom Value4.
15-10	Reserved	0	Reserved
9-0	4BITECCVAL3	0-3FFh	Calculated 4-bit ECC or Syndrom Value3.

Copyright © 2013–2016, Texas Instruments Incorporated

18.4.18 NAND Flash 4-Bit ECC Register 3 (NAND4BITECC3)

The NAND Flash 4-bit ECC register 3 (NAND4BITECC3) is shown in Figure 18-48 and described in Table 18-66.

Figure 18-48. NAND Flash 4-Bit ECC Register 3 (NAND4BITECC3)

31	26	25	16
Reserved		4BITECCVAL6	
R-0		R/W-0	

15	10	9	0
Reserved		4BITECCVAL5	
R-0		R/W-0	

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

Table 18-66. NAND Flash 4-Bit ECC Register 3 (NAND4BITECC3) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	Reserved
25-16	4BITECCVAL6	0-3FFh	Calculated 4-bit ECC or Syndrom Value6.
15-10	Reserved	0	Reserved
9-0	4BITECCVAL5	0-3FFh	Calculated 4-bit ECC or Syndrom Value5.

18.4.19 NAND Flash 4-Bit ECC Register 4 (NAND4BITECC4)

The NAND Flash 4-bit ECC register 4 (NAND4BITECC4) is shown in Figure 18-49 and described in Table 18-67.

Figure 18-49. NAND Flash 4-Bit ECC Register 4 (NAND4BITECC4)

31	26	25	16
Reserved		4BITECCVAL8	
R-0		R/W-0	

15	10	9	0
Reserved		4BITECCVAL7	
R-0		R/W-0	

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

Table 18-67. NAND Flash 4-Bit ECC Register 4 (NAND4BITECC4) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	Reserved
25-16	4BITECCVAL8	0-3FFh	Calculated 4-bit ECC or Syndrom Value8.
15-10	Reserved	0	Reserved
9-0	4BITECCVAL7	0-3FFh	Calculated 4-bit ECC or Syndrom Value7.

Copyright © 2013–2016, Texas Instruments Incorporated

18.4.20 NAND Flash 4-Bit ECC Error Address Register 1 (NANDERRADD1)

The NAND Flash 4-bit ECC error register 1 (NANDERRADD1) is shown in Figure 18-50 and described in Table 18-68.

Figure 18-50. NAND Flash 4-Bit ECC Error Address Register 1 (NANDERRADD1)

31	26	25	16
Reserved		4BITECCERRADD2	
R-0		R/W-0	

15	10	9	0
Reserved		4BITECCERRADD1	
R-0		R/W-0	

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

Table 18-68. NAND Flash 4-Bit ECC Error Address Register 1 (NANDERRADD1) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	Reserved
25-16	4BITECCERRADD2	0-3FFh	Calculated 4-bit ECC Error Address 2.
15-10	Reserved	0	Reserved
9-0	4BITECCERRADD1	0-3FFh	Calculated 4-bit ECC Error Address 1.

18.4.21 NAND Flash 4-Bit ECC Error Address Register 2 (NANDERRADD2)

The NAND Flash 4-bit ECC error register 2 (NANDERRADD2) is shown in Figure 18-51and described in Table 18-69.

Figure 18-51. NAND Flash 4-Bit ECC Error Address Register 2 (NANDERRADD2)

31	26	25	16
Reserved		4BITECCERRADD4	
R-0		R/W-0	

15	10	9	0
Reserved		4BITECCERRADD3	
R-0		R/W-0	

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

Table 18-69. NAND Flash 4-Bit ECC Error Address Register 2 (NANDERRADD2) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	Reserved
25-16	4BITECCERRADD4	0-3FFh	Calculated 4-bit ECC Error Address 4.
15-10	Reserved	0	Reserved
9-0	4BITECCERRADD3	0-3FFh	Calculated 4-bit ECC Error Address 3.

Copyright © 2013–2016, Texas Instruments Incorporated

18.4.22 NAND Flash 4-Bit ECC Error Value Register 1 (NANDERRVAL1)

The NAND Flash 4-bit ECC error value register 1 (NANDERRVAL1) is shown in Figure 18-52 and described in Table 18-70.

Figure 18-52. NAND Flash 4-Bit ECC Error Value Register 1 (NANDERRVAL1)

31	26	25	16
Reserved		4BITECCERRVAL2	
R-0		R/W-0	

15	10	9	0
Reserved		4BITECCERRVAL1	
R-0		R/W-0	

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

Table 18-70. NAND Flash 4-Bit ECC Error Value Register 1 (NANDERRVAL1) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	Reserved
25-16	4BITECCERRVAL2	0-3FFh	Calculated 4-bit ECC Error Value 2.
15-10	Reserved	0	Reserved
9-0	4BITECCERRVAL1	0-3FFh	Calculated 4-bit ECC Error Value 1.

18.4.23 NAND Flash 4-Bit ECC Error Value Register 2 (NANDERRVAL2)

The NAND Flash 4-bit ECC error value register 2 (NANDERRVAL2) is shown in Figure 18-53 and described in Table 18-71.

Figure 18-53. NAND Flash 4-Bit ECC Error Value Register 2 (NANDERRVAL2)

31	26	25	16
Reserved		4BITECCERRVAL4	
R-0		R/W-0	

15	10	9	0
Reserved		4BITECCERRVAL3	
R-0		R/W-0	

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

Table 18-71. NAND Flash 4-Bit ECC Error Value Register 2 (NANDERRVAL2) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	Reserved
25-16	4BITECCERRVAL4	0-3FFh	Calculated 4-bit ECC Error Value 4.
15-10	Reserved	0	Reserved
9-0	4BITECCERRVAL3	0-3FFh	Calculated 4-bit ECC Error Value 3.

Copyright © 2013–2016, Texas Instruments Incorporated

General-Purpose Input/Output (GPIO)

The GPIO peripheral provides dedicated general-purpose pins that can be configured as either inputs or outputs. When configured as an output, you can write to an internal register to control the state driven on the output pin. When configured as an input, you can detect the state of the input by reading the state of an internal register. This chapter describes the GPIO.

Topic **Page**

19.1 Introduction **784**
19.2 Architecture **785**
19.3 Registers **793**

Copyright © 2013–2016, Texas Instruments Incorporated

19.1 Introduction

19.1.1 Purpose of the Peripheral

Most system-on-chip (SoC) devices require some general-purpose input/output (GPIO) functionality in order to interact with other components in the system using low-speed interface pins. The control and use of the GPIO capability on this device is grouped together in the GPIO peripheral and is described in the following sections.

19.1.2 Features

The GPIO peripheral consists of the following features.

- Output set/clear functionality through separate data set and clear registers allows multiple software processes to control GPIO signals without critical section protection.
- Set/clear functionality through writing to a single output data register is also supported.
- Separate input/output registers
  - Output register can be read to reflect output drive status.
  - Input register can be read to reflect pin status.
- All GPIO signals can be used as interrupt sources with configurable edge detection.
- All GPIO signals can be used to generate events to the EDMA.

19.1.3 Functional Block Diagram

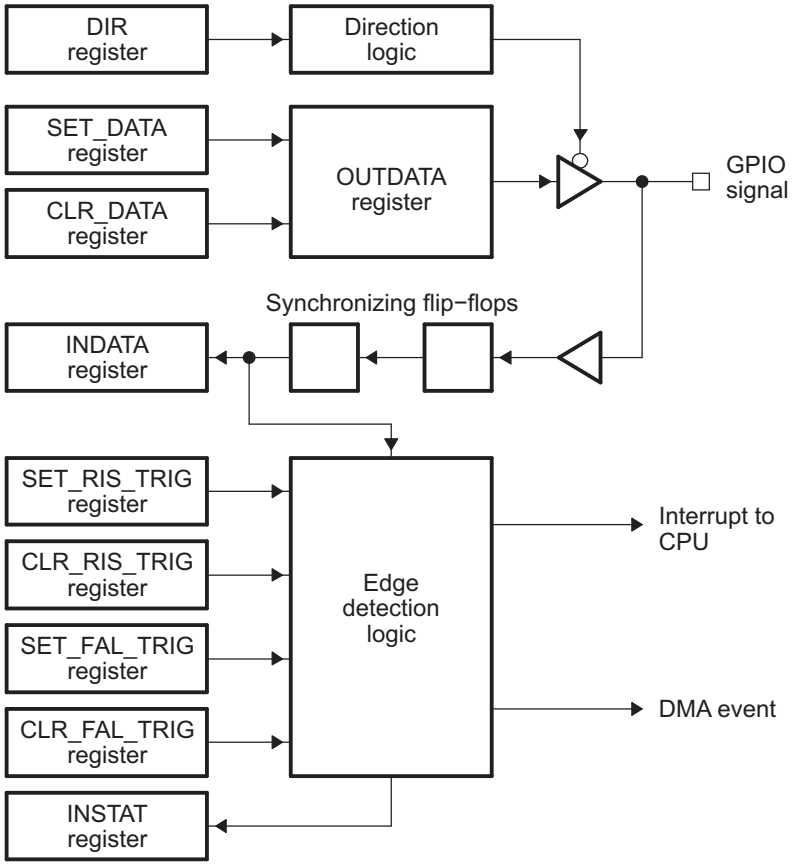
Figure 19-1 shows a block diagram of the GPIO peripheral.

19.1.4 Industry Standard(s) Compliance Statement

The GPIO peripheral connects to external devices. While it is possible that the software implements some standard connectivity protocol over GPIO, the GPIO peripheral itself is not compliant with any such standards.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 19-1. GPIO Block Diagram



19.2 Architecture

The following sections describe the GPIO peripheral.

19.2.1 Clock Control

The input clock to the GPIO peripheral is indicated in the device datasheet. The maximum operating speed of the GPIO peripheral is limited by system-level latencies. More specifically, how quickly the GPIO registers can be written to or read from.

19.2.2 Signal Descriptions

The number of GPIO signals supported will vary between devices. For information on the number of signals supported and the package pinout of each GPIO signal, see your device-specific data manual.

19.2.3 Pin Multiplexing

Extensive pin multiplexing is used to accommodate the largest number of peripheral functions in the smallest possible package. Pin multiplexing is controlled using a combination of hardware configuration at device reset and software programmable register settings. Refer to the device-specific data manual to determine how pin multiplexing affects the GPIO module.

19.2.4 Endianness Considerations

The GPIO operation is independent of endianness; therefore, there are no endianness considerations for the GPIO module.

Copyright © 2013–2016, Texas Instruments Incorporated

19.2.5 GPIO Register Structure

The GPIO signals are grouped by banks of 16 signals per bank. Each bank of GPIO signals has several registers with various control fields for each GPIO signal. Each 32-bit GPIO control register controls a pair of GPIO banks.

The register names for each bank of control registers (or pair of banks of GPIO bits) are all of the form *register_nameXY*, where *X* and *Y* are the two banks of GPIO bits controlled, such as 01, 23, 45, etc. The register fields associated with each GPIO are all of the form B*k*P*j*, where *k* is the GPIO bank and *j* is the pin number within the GPIO bank. For example, for GP2[5], which is located in GPIO bank 2, the control register names are of the form *register_name*23, and the register field associated with GP2[5] is GP2P5.

Table 19-1 shows the banks and register control bit information associated with each GPIO pin for up to 144 supportable pins. The table is not indicative of how many GPIO pins are supported on a device; it is only a reference for what register and field mappings look like for the first 144 supportable GPIO pins. For devices with less than 144 GPIO pins, assume that the extraneous fields and registers listed in the table are Reserved with no function. For devices with more than 144 GPIO pins, additional control registers and fields should be appended using the same numbering scheme in the table. Detailed information regarding the specific register names for each bank and the contents and function of these registers is presented in Section 19.3.

Table 19-1. GPIO Register Bits and Banks Associated With GPIO Signals

GPIO Pin Number	GPIO Signal Name	Bank Number	Control Registers	Register Bit	Register Field
1	GP0[0]	0	*register_name*01	Bit 0	GP0P0
2	GP0[1]	0	*register_name*01	Bit 1	GP0P1
3	GP0[2]	0	*register_name*01	Bit 2	GP0P2
4	GP0[3]	0	*register_name*01	Bit 3	GP0P3
5	GP0[4]	0	*register_name*01	Bit 4	GP0P4
6	GP0[5]	0	*register_name*01	Bit 5	GP0P5
7	GP0[6]	0	*register_name*01	Bit 6	GP0P6
8	GP0[7]	0	*register_name*01	Bit 7	GP0P7
9	GP0[8]	0	*register_name*01	Bit 8	GP0P8
10	GP0[9]	0	*register_name*01	Bit 9	GP0P9
11	GP0[10]	0	*register_name*01	Bit 10	GP0P10
12	GP0[11]	0	*register_name*01	Bit 11	GP0P11
13	GP0[12]	0	*register_name*01	Bit 12	GP0P12
14	GP0[13]	0	*register_name*01	Bit 13	GP0P13
15	GP0[14]	0	*register_name*01	Bit 14	GP0P14
16	GP0[15]	0	*register_name*01	Bit 15	GP0P15
17	GP1[0]	1	*register_name*01	Bit 16	GP1P0
18	GP1[1]	1	*register_name*01	Bit 17	GP1P1
19	GP1[2]	1	*register_name*01	Bit 18	GP1P2
20	GP1[3]	1	*register_name*01	Bit 19	GP1P3
21	GP1[4]	1	*register_name*01	Bit 20	GP1P4
22	GP1[5]	1	*register_name*01	Bit 21	GP1P5
23	GP1[6]	1	*register_name*01	Bit 22	GP1P6
24	GP1[7]	1	*register_name*01	Bit 23	GP1P7
25	GP1[8]	1	*register_name*01	Bit 24	GP1P8
26	GP1[9]	1	*register_name*01	Bit 25	GP1P9
27	GP1[10]	1	*register_name*01	Bit 26	GP1P10
28	GP1[11]	1	*register_name*01	Bit 27	GP1P11
29	GP1[12]	1	*register_name*01	Bit 28	GP1P12
30	GP1[13]	1	*register_name*01	Bit 29	GP1P13

Copyright © 2013–2016, Texas Instruments Incorporated

Table 19-1. GPIO Register Bits and Banks Associated With GPIO Signals (continued)

GPIO Pin Number	GPIO Signal Name	Bank Number	Control Registers	Register Bit	Register Field
31	GP1[14]	1	*register_name*01	Bit 30	GP1P14
32	GP1[15]	1	*register_name*01	Bit 31	GP1P15
33	GP2[0]	2	*register_name*23	Bit 0	GP2P0
34	GP2[1]	2	*register_name*23	Bit 1	GP2P1
35	GP2[2]	2	*register_name*23	Bit 2	GP2P2
36	GP2[3]	2	*register_name*23	Bit 3	GP2P3
37	GP2[4]	2	*register_name*23	Bit 4	GP2P4
38	GP2[5]	2	*register_name*23	Bit 5	GP2P5
39	GP2[6]	2	*register_name*23	Bit 6	GP2P6
40	GP2[7]	2	*register_name*23	Bit 7	GP2P7
41	GP2[8]	2	*register_name*23	Bit 8	GP2P8
42	GP2[9]	2	*register_name*23	Bit 9	GP2P9
43	GP2[10]	2	*register_name*23	Bit 10	GP2P10
44	GP2[11]	2	*register_name*23	Bit 11	GP2P11
45	GP2[12]	2	*register_name*23	Bit 12	GP2P12
46	GP2[13]	2	*register_name*23	Bit 13	GP2P13
47	GP2[14]	2	*register_name*23	Bit 14	GP2P14
48	GP2[15]	2	*register_name*23	Bit 15	GP2P15
49	GP3[0]	3	*register_name*23	Bit 16	GP3P0
50	GP3[1]	3	*register_name*23	Bit 17	GP3P1
51	GP3[2]	3	*register_name*23	Bit 18	GP3P2
52	GP3[3]	3	*register_name*23	Bit 19	GP3P3
53	GP3[4]	3	*register_name*23	Bit 20	GP3P4
54	GP3[5]	3	*register_name*23	Bit 21	GP3P5
55	GP3[6]	3	*register_name*23	Bit 22	GP3P6
56	GP3[7]	3	*register_name*23	Bit 23	GP3P7
57	GP3[8]	3	*register_name*23	Bit 24	GP3P8
58	GP3[9]	3	*register_name*23	Bit 25	GP3P9
59	GP3[10]	3	*register_name*23	Bit 26	GP3P10
60	GP3[11]	3	*register_name*23	Bit 27	GP3P11
61	GP3[12]	3	*register_name*23	Bit 28	GP3P12
62	GP3[13]	3	*register_name*23	Bit 29	GP3P13
63	GP3[14]	3	*register_name*23	Bit 30	GP3P14
64	GP3[15]	3	*register_name*23	Bit 31	GP3P15
65	GP4[0]	4	*register_name*45	Bit 0	GP4P0
66	GP4[1]	4	*register_name*45	Bit 1	GP4P1
67	GP4[2]	4	*register_name*45	Bit 2	GP4P2
68	GP4[3]	4	*register_name*45	Bit 3	GP4P3
69	GP4[4]	4	*register_name*45	Bit 4	GP4P4
70	GP4[5]	4	*register_name*45	Bit 5	GP4P5
71	GP4[6]	4	*register_name*45	Bit 6	GP4P6
72	GP4[7]	4	*register_name*45	Bit 7	GP4P7
73	GP4[8]	4	*register_name*45	Bit 8	GP4P8
74	GP4[9]	4	*register_name*45	Bit 9	GP4P9
75	GP4[10]	4	*register_name*45	Bit 10	GP4P10
76	GP4[11]	4	*register_name*45	Bit 11	GP4P11
77	GP4[12]	4	*register_name*45	Bit 12	GP4P12

Copyright © 2013–2016, Texas Instruments Incorporated

Table 19-1. GPIO Register Bits and Banks Associated With GPIO Signals (continued)

GPIO Pin Number	GPIO Signal Name	Bank Number	Control Registers	Register Bit	Register Field
78	GP4[13]	4	*register_name*45	Bit 13	GP4P13
79	GP4[14]	4	*register_name*45	Bit 14	GP4P14
80	GP4[15]	4	*register_name*45	Bit 15	GP4P15
81	GP5[0]	5	*register_name*45	Bit 16	GP5P0
82	GP5[1]	5	*register_name*45	Bit 17	GP5P1
83	GP5[2]	5	*register_name*45	Bit 18	GP5P2
84	GP5[3]	5	*register_name*45	Bit 19	GP5P3
85	GP5[4]	5	*register_name*45	Bit 20	GP5P4
86	GP5[5]	5	*register_name*45	Bit 21	GP5P5
87	GP5[6]	5	*register_name*45	Bit 22	GP5P6
88	GP5[7]	5	*register_name*45	Bit 23	GP5P7
89	GP5[8]	5	*register_name*45	Bit 24	GP5P8
90	GP5[9]	5	*register_name*45	Bit 25	GP5P9
91	GP5[10]	5	*register_name*45	Bit 26	GP5P10
92	GP5[11]	5	*register_name*45	Bit 27	GP5P11
93	GP5[12]	5	*register_name*45	Bit 28	GP5P12
94	GP5[13]	5	*register_name*45	Bit 29	GP5P13
95	GP5[14]	5	*register_name*45	Bit 30	GP5P14
96	GP5[15]	5	*register_name*45	Bit 31	GP5P15
97	GP6[0]	6	*register_name*67	Bit 0	GP6P0
98	GP6[1]	6	*register_name*67	Bit 1	GP6P1
99	GP6[2]	6	*register_name*67	Bit 2	GP6P2
100	GP6[3]	6	*register_name*67	Bit 3	GP6P3
101	GP6[4]	6	*register_name*67	Bit 4	GP6P4
102	GP6[5]	6	*register_name*67	Bit 5	GP6P5
103	GP6[6]	6	*register_name*67	Bit 6	GP6P6
104	GP6[7]	6	*register_name*67	Bit 7	GP6P7
105	GP6[8]	6	*register_name*67	Bit 8	GP6P8
106	GP6[9]	6	*register_name*67	Bit 9	GP6P9
107	GP6[10]	6	*register_name*67	Bit 10	GP6P10
108	GP6[11]	6	*register_name*67	Bit 11	GP6P11
109	GP6[12]	6	*register_name*67	Bit 12	GP6P12
110	GP6[13]	6	*register_name*67	Bit 13	GP6P13
111	GP6[14]	6	*register_name*67	Bit 14	GP6P14
112	GP6[15]	6	*register_name*67	Bit 15	GP6P15
113	GP7[0]	7	*register_name*67	Bit 16	GP7P0
114	GP7[1]	7	*register_name*67	Bit 17	GP7P1
115	GP7[2]	7	*register_name*67	Bit 18	GP7P2
116	GP7[3]	7	*register_name*67	Bit 19	GP7P3
117	GP7[4]	7	*register_name*67	Bit 20	GP7P4
118	GP7[5]	7	*register_name*67	Bit 21	GP7P5
119	GP7[6]	7	*register_name*67	Bit 22	GP7P6
120	GP7[7]	7	*register_name*67	Bit 23	GP7P7
121	GP7[8]	7	*register_name*67	Bit 24	GP7P8
122	GP7[9]	7	*register_name*67	Bit 25	GP7P9
123	GP7[10]	7	*register_name*67	Bit 26	GP7P10
124	GP7[11]	7	*register_name*67	Bit 27	GP7P11

Copyright © 2013–2016, Texas Instruments Incorporated

Table 19-1. GPIO Register Bits and Banks Associated With GPIO Signals (continued)

GPIO Pin Number	GPIO Signal Name	Bank Number	Control Registers	Register Bit	Register Field
125	GP7[12]	7	*register_name*67	Bit 28	GP7P12
126	GP7[13]	7	*register_name*67	Bit 29	GP7P13
127	GP7[14]	7	*register_name*67	Bit 30	GP7P14
128	GP7[15]	7	*register_name*67	Bit 31	GP7P15
129	GP8[0]	8	*register_name*8	Bit 0	GP8P0
130	GP8[1]	8	*register_name*8	Bit 1	GP8P1
131	GP8[2]	8	*register_name*8	Bit 2	GP8P2
132	GP8[3]	8	*register_name*8	Bit 3	GP8P3
133	GP8[4]	8	*register_name*8	Bit 4	GP8P4
134	GP8[5]	8	*register_name*8	Bit 5	GP8P5
135	GP8[6]	8	*register_name*8	Bit 6	GP8P6
136	GP8[7]	8	*register_name*8	Bit 7	GP8P7
137	GP8[8]	8	*register_name*8	Bit 8	GP8P8
138	GP8[9]	8	*register_name*8	Bit 9	GP8P9
139	GP8[10]	8	*register_name*8	Bit 10	GP8P10
140	GP8[11]	8	*register_name*8	Bit 11	GP8P11
141	GP8[12]	8	*register_name*8	Bit 12	GP8P12
142	GP8[13]	8	*register_name*8	Bit 13	GP8P13
143	GP8[14]	8	*register_name*8	Bit 14	GP8P14
144	GP8[15]	8	*register_name*8	Bit 15	GP8P15

19.2.6 Using a GPIO Signal as an Output

GPIO signals are configured to operate as inputs or outputs by writing the appropriate value to the GPIO direction register (DIR). This section describes using the GPIO signal as an output signal.

19.2.6.1 Configuring a GPIO Output Signal

To configure a given GPIO signal as an output, clear the bit in DIR that is associated with the desired GPIO signal. For detailed information on DIR, see Section 19.3.

19.2.6.2 Controlling the GPIO Output Signal State

There are three registers that control the output state driven on a GPIO signal configured as an output:

1. GPIO set data register (SET_DATA) controls driving GPIO signals high.
2. GPIO clear data register (CLR_DATA) controls driving GPIO signals low.
3. GPIO output data register (OUT_DATA) contains the current state of the output signals.

Reading SET_DATA, CLR_DATA, and OUT_DATA returns the output state, not necessarily the actual signal state (since some signals may be configured as inputs). The actual signal state is read using the GPIO input data register (IN_DATA) associated with the desired GPIO signal. IN_DATA contains the actual logic state on the external signal.

For detailed information on these registers, see Section 19.3.

Copyright © 2013–2016, Texas Instruments Incorporated

19.2.6.2.1 Driving a GPIO Output Signal High

To drive a GPIO signal high, use one of the following methods:

- Write a logic 1 to the bit in SET_DATA associated with the desired GPIO signal(s) to be driven high. Bit positions in SET_DATA containing logic 0 do not affect the state of the associated output signals.
- Modify the bit in OUT_DATA associated with the desired GPIO signal by using a read-modify-write operation. The logic states driven on the GPIO output signals match the logic values written to all bits in OUT_DATA.

For GPIO signals configured as inputs, the values written to the associated SET_DATA, CLR_DATA, and OUT_DATA bits have no effect.

19.2.6.2.2 Driving a GPIO Output Signal Low

To drive a GPIO signal low, use one of the following methods:

- Write a logic 1 to the bit in CLR_DATA associated with the desired GPIO signal(s) to be driven low. Bit positions in CLR_DATA containing logic 0 do not affect the state of the associated output signals.
- Modify the bit in OUT_DATA associated with the desired GPIO signal by using a read-modify-write operation. The logic states driven on the GPIO output signals match the logic values written to all bits in OUT_DATA.

For GPIO signals configured as inputs, the values written to the associated SET_DATA, CLR_DATA, and OUT_DATA bits have no effect.

19.2.7 Using a GPIO Signal as an Input

GPIO signals are configured to operate as inputs or outputs by writing the appropriate value to the GPIO direction register (DIR). This section describes using the GPIO signal as an input signal.

19.2.7.1 Configuring a GPIO Input Signal

To configure a given GPIO signal as an input, set the bit in DIR that is associated with the desired GPIO signal. For detailed information on DIR, see Section 19.3.

19.2.7.2 Reading a GPIO Input Signal

The current state of the GPIO signals is read using the GPIO input data register (IN_DATA).

- For GPIO signals configured as inputs, reading IN_DATA returns the state of the input signal synchronized to the GPIO peripheral clock.
- For GPIO signals configured as outputs, reading IN_DATA returns the output value being driven by the device.

Some signals may utilize open-drain output buffers for wired-logic operations. For open-drain GPIO signals, reading IN_DATA returns the wired-logic value on the signal (which will not be driven by the device alone). Information on any signals using open-drain outputs is available in your device-specific data manual.

To use GPIO input signals as interrupt sources, see Section 19.2.10.

19.2.8 Reset Considerations

The GPIO peripheral has two reset sources: software reset and hardware reset.

19.2.8.1 Software Reset Considerations

A software reset (such as a reset initiated through the emulator) does not modify the configuration and state of the GPIO signals. A reset invoked via the Power and Sleep Controller (PSC) (GPIO clock disable, PSC reset, followed by GPIO clock enable) will result in the default configuration register settings. For details on the PSC, see the *Power and Sleep Controller (PSC)* chapter.

Copyright © 2013–2016, Texas Instruments Incorporated

19.2.8.2 Hardware Reset Considerations

A hardware reset does reset the GPIO configuration and data registers to their default states; therefore, affecting the configuration and state of the GPIO signals.

19.2.9 Initialization

The following steps are required to configure the GPIO module after a hardware reset:

1. Perform the necessary device pin multiplexing setup (see your device-specific data manual).
2. Program the Power and Sleep Controller (PSC) to enable the GPIO module. For details on the PSC, see the *Power and Sleep Controller (PSC)* chapter.
3. Program the direction, data, and interrupt control registers to set the configuration of the desired GPIO pins (described in this chapter).

The GPIO module is now ready to perform data transactions.

19.2.10 Interrupt Support

The GPIO peripheral can send an interrupt event to the CPU.

19.2.10.1 Interrupt Events and Requests

All GPIO signals can be configured to generate interrupts. The device supports interrupts from single GPIO signals, interrupts from banks of GPIO signals, or both.

Note that the GPIO interrupts may also be used to provide synchronization events to the DMA controller.

19.2.10.2 Enabling GPIO Interrupt Events

GPIO interrupt events are enabled in banks of 16 by setting the appropriate bit(s) in the GPIO interrupt per-bank enable register (BINTEN). For example, to enable bank 0 interrupts (events from GP0[15-0]), set bit 0 in BINTEN; to enable bank 3 interrupts (events from GP3[15-0]), set bit 3 in BINTEN.

For detailed information on BINTEN, see Section 19.3.

19.2.10.3 Configuring GPIO Interrupt Edge Triggering

Each GPIO interrupt source can be configured to generate an interrupt on the GPIO signal rising edge, falling edge, both edges, or neither edge (no event). The edge detection is synchronized to the GPIO peripheral module clock.

The following four registers control the configuration of the GPIO interrupt edge detection:

1. The GPIO set rising edge interrupt register (SET_RIS_TRIG) enables GPIO interrupts on the occurrence of a rising edge on the GPIO signal.
2. The GPIO clear rising edge interrupt register (CLR_RIS_TRIG) disables GPIO interrupts on the occurrence of a rising edge on the GPIO signal.
3. The GPIO set falling edge interrupt register (SET_FAL_TRIG) enables GPIO interrupts on the occurrence of a falling edge on the GPIO signal.
4. The GPIO clear falling edge interrupt register (CLR_FAL_TRIG) disables GPIO interrupts on the occurrence of a falling edge on the GPIO signal.

To configure a GPIO interrupt to occur only on rising edges of the GPIO signal:

- Write a logic 1 to the associated bit in SET_RIS_TRIG.
- Write a logic 1 to the associated bit in CLR_FAL_TRIG.

To configure a GPIO interrupt to occur only on falling edges of the GPIO signal:

- Write a logic 1 to the associated bit in SET_FAL_TRIG.
- Write a logic 1 to the associated bit in CLR_RIS_TRIG.

Copyright © 2013–2016, Texas Instruments Incorporated

To configure a GPIO interrupt to occur on both the rising and falling edges of the GPIO signal:

- Write a logic 1 to the associated bit in SET_RIS_TRIG.
- Write a logic 1 to the associated bit in SET_FAL_TRIG.

To disable a specific GPIO interrupt:

- Write a logic 1 to the associated bit in CLR_RIS_TRIG.
- Write a logic 1 to the associated bit in CLR_FAL_TRIG.

For detailed information on these registers, see Section 19.3.

Note that the direction of the GPIO signal does not have to be an input for the interrupt event generation to work. When a GPIO signal is configured as an output, the software can change the GPIO signal state and, in turn, generate an interrupt. This can be useful for debugging interrupt signal connectivity.

19.2.10.4 GPIO Interrupt Status

The status of GPIO interrupt events can be monitored by reading the GPIO interrupt status register (INTSTAT). Pending GPIO interrupts are indicated with a logic 1 in the associated bit position; interrupts that are not pending are indicated with a logic 0.

For the GPIO bank interrupts, INTSTAT can be used to determine which GPIO interrupt occurred. It is the responsibility of software to ensure that all pending GPIO interrupts are appropriately serviced.

Pending GPIO interrupt flags can be cleared by writing a logic 1 to the associated bit position in INTSTAT.

For detailed information on INTSTAT, see Section 19.3.

19.2.10.5 Interrupt Multiplexing

GPIO interrupts may be multiplexed with other interrupt functions on the device.

19.2.11 EDMA Event Support

The GPIO peripheral may provide synchronization events to the DMA controller.

19.2.12 Power Management

The GPIO peripheral can be placed in reduced-power modes to conserve power during periods of low activity. The power management of the GPIO peripheral is controlled by the processor Power and Sleep Controller (PSC). The PSC acts as a master controller for power management for all of the peripherals on the device. For detailed information on power management procedures using the PSC, see the *Power and Sleep Controller (PSC)* chapter.

When the GPIO peripheral is placed in a low-power state by the PSC, the interrupt generation capability is suspended until the GPIO peripheral is removed from the low-power state. While in the low-power state, the GPIO signals configured as outputs are maintained at their state prior to the GPIO peripheral entering the low-power state.

19.2.13 Emulation Considerations

The GPIO peripheral is not affected by emulation suspend events (such as halts and breakpoints).

Copyright © 2013–2016, Texas Instruments Incorporated

19.3 Registers

Table 19-2 lists the memory-mapped registers for the general-purpose input/output (GPIO). The table enumerates the registers required to support 144 GPIO pins, however not all devices will support 144 GPIO pins. For devices with less than 144 GPIO pins, assume that the extraneous fields and registers are Reserved and serve no function. For devices with more than 144 GPIO pins, append registers and fields as necessary using the address offset scheme in the table. See your device-specific data manual for the number of GPIO pins supported and the base memory address for these registers.

Table 19-2. GPIO Registers

Offset	Acronym	Register Description	Section
0h	REVID	Revision ID Register	Section 19.3.1
8h	BINTEN	GPIO Interrupt Per-Bank Enable Register	Section 19.3.2
		GPIO Banks 0 and 1	
10h	DIR01	GPIO Banks 0 and 1 Direction Register	Section 19.3.3
14h	OUT_DATA01	GPIO Banks 0 and 1 Output Data Register	Section 19.3.4
18h	SET_DATA01	GPIO Banks 0 and 1 Set Data Register	Section 19.3.5
1Ch	CLR_DATA01	GPIO Banks 0 and 1 Clear Data Register	Section 19.3.6
20h	IN_DATA01	GPIO Banks 0 and 1 Input Data Register	Section 19.3.7
24h	SET_RIS_TRIG01	GPIO Banks 0 and 1 Set Rising Edge Interrupt Register	Section 19.3.8
28h	CLR_RIS_TRIG01	GPIO Banks 0 and 1 Clear Rising Edge Interrupt Register	Section 19.3.9
2Ch	SET_FAL_TRIG01	GPIO Banks 0 and 1 Set Falling Edge Interrupt Register	Section 19.3.10
30h	CLR_FAL_TRIG01	GPIO Banks 0 and 1 Clear Falling Edge Interrupt Register	Section 19.3.11
34h	INTSTAT01	GPIO Banks 0 and 1 Interrupt Status Register	Section 19.3.12
		GPIO Banks 2 and 3	
38h	DIR23	GPIO Banks 2 and 3 Direction Register	Section 19.3.3
3Ch	OUT_DATA23	GPIO Banks 2 and 3 Output Data Register	Section 19.3.4
40h	SET_DATA23	GPIO Banks 2 and 3 Set Data Register	Section 19.3.5
44h	CLR_DATA23	GPIO Banks 2 and 3 Clear Data Register	Section 19.3.6
48h	IN_DATA23	GPIO Banks 2 and 3 Input Data Register	Section 19.3.7
4Ch	SET_RIS_TRIG23	GPIO Banks 2 and 3 Set Rising Edge Interrupt Register	Section 19.3.8
50h	CLR_RIS_TRIG23	GPIO Banks 2 and 3 Clear Rising Edge Interrupt Register	Section 19.3.9
54h	SET_FAL_TRIG23	GPIO Banks 2 and 3 Set Falling Edge Interrupt Register	Section 19.3.10
58h	CLR_FAL_TRIG23	GPIO Banks 2 and 3 Clear Falling Edge Interrupt Register	Section 19.3.11
5Ch	INTSTAT23	GPIO Banks 2 and 3 Interrupt Status Register	Section 19.3.12
		GPIO Banks 4 and 5	
60h	DIR45	GPIO Banks 4 and 5 Direction Register	Section 19.3.3
64h	OUT_DATA45	GPIO Banks 4 and 5 Output Data Register	Section 19.3.4
68h	SET_DATA45	GPIO Banks 4 and 5 Set Data Register	Section 19.3.5
6Ch	CLR_DATA45	GPIO Banks 4 and 5 Clear Data Register	Section 19.3.6
70h	IN_DATA45	GPIO Banks 4 and 5 Input Data Register	Section 19.3.7
74h	SET_RIS_TRIG45	GPIO Banks 4 and 5 Set Rising Edge Interrupt Register	Section 19.3.8
78h	CLR_RIS_TRIG45	GPIO Banks 4 and 5 Clear Rising Edge Interrupt Register	Section 19.3.9
7Ch	SET_FAL_TRIG45	GPIO Banks 4 and 5 Set Falling Edge Interrupt Register	Section 19.3.10
80h	CLR_FAL_TRIG45	GPIO Banks 4 and 5 Clear Falling Edge Interrupt Register	Section 19.3.11
84h	INTSTAT45	GPIO Banks 4 and 5 Interrupt Status Register	Section 19.3.12

Copyright © 2013–2016, Texas Instruments Incorporated

Table 19-2. GPIO Registers (continued)

Offset	Acronym	Register Description	Section
		GPIO Banks 6 and 7	
88h	DIR67	GPIO Banks 6 and 7 Direction Register	Section 19.3.3
8Ch	OUT_DATA67	GPIO Banks 6 and 7 Output Data Register	Section 19.3.4
90h	SET_DATA67	GPIO Banks 6 and 7 Set Data Register	Section 19.3.5
94h	CLR_DATA67	GPIO Banks 6 and 7 Clear Data Register	Section 19.3.6
98h	IN_DATA67	GPIO Banks 6 and 7 Input Data Register	Section 19.3.7
9Ch	SET_RIS_TRIG67	GPIO Banks 6 and 7 Set Rising Edge Interrupt Register	Section 19.3.8
A0h	CLR_RIS_TRIG67	GPIO Banks 6 and 7 Clear Rising Edge Interrupt Register	Section 19.3.9
A4h	SET_FAL_TRIG67	GPIO Banks 6 and 7 Set Falling Edge Interrupt Register	Section 19.3.10
A8h	CLR_FAL_TRIG67	GPIO Banks 6 and 7 Clear Falling Edge Interrupt Register	Section 19.3.11
ACh	INTSTAT67	GPIO Banks 6 and 7 Interrupt Status Register	Section 19.3.12
		GPIO Bank 8	
B0h	DIR8	GPIO Bank 8 Direction Register	Section 19.3.3
B4h	OUT_DATA8	GPIO Bank 8 Output Data Register	Section 19.3.4
B8h	SET_DATA8	GPIO Bank 8 Set Data Register	Section 19.3.5
BCh	CLR_DATA8	GPIO Bank 8 Clear Data Register	Section 19.3.6
C0h	IN_DATA8	GPIO Bank 8 Input Data Register	Section 19.3.7
C4h	SET_RIS_TRIG8	GPIO Bank 8 Set Rising Edge Interrupt Register	Section 19.3.8
C8h	CLR_RIS_TRIG8	GPIO Bank 8 Clear Rising Edge Interrupt Register	Section 19.3.9
CCh	SET_FAL_TRIG8	GPIO Bank 8 Set Falling Edge Interrupt Register	Section 19.3.10
D0h	CLR_FAL_TRIG8	GPIO Bank 8 Clear Falling Edge Interrupt Register	Section 19.3.11
D4h	INTSTAT8	GPIO Bank 8 Interrupt Status Register	Section 19.3.12

19.3.1 Revision ID Register (REVID)

The revision ID register (REVID) contains the peripheral version information. REVID is shown in Figure 19-2 and described in Table 19-3.

Figure 19-2. Revision ID Register (REVID)

31	0
REV	
R-4483 0105h	

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

Table 19-3. Revision ID Register (REVID) Field Descriptions

Bit	Field	Value	Description
31-0	REV	4483 0105h	Peripheral Revision

Copyright © 2013–2016, Texas Instruments Incorporated

19.3.2 GPIO Interrupt Per-Bank Enable Register (BINTEN)

The GPIO interrupt per-bank enable register (BINTEN) is shown in Figure 19-3 and described in Table 19-4. For information on which GPIO signals are associated with each bank, see Table 19-1. Note that the bits in BINTEN control both the interrupt and EDMA events.

Figure 19-3. GPIO Interrupt Per-Bank Enable Register (BINTEN)

31	16
Reserved	
R-0	

15	9	8	7	6	5	4	3	2	1	0
Reserved		EN8	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0
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

Table 19-4. GPIO Interrupt Per-Bank Enable Register (BINTEN) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	EN8		Bank 8 interrupt enable is used to disable or enable the bank 8 interrupts (events from GP8[15-0]).
		0	Bank 8 interrupts are disabled.
		1	Bank 8 interrupts are enabled.
7	EN7		Bank 7 interrupt enable is used to disable or enable the bank 7 interrupts (events from GP7[15-0]).
		0	Bank 7 interrupts are disabled.
		1	Bank 7 interrupts are enabled.
6	EN6		Bank 6 interrupt enable is used to disable or enable the bank 6 interrupts (events from GP6[15-0]).
		0	Bank 6 interrupts are disabled.
		1	Bank 6 interrupts are enabled.
5	EN5		Bank 5 interrupt enable is used to disable or enable the bank 5 interrupts (events from GP5[15-0]).
		0	Bank 5 interrupts are disabled.
		1	Bank 5 interrupts are enabled.
4	EN4		Bank 4 interrupt enable is used to disable or enable the bank 4 interrupts (events from GP4[15-0]).
		0	Bank 4 interrupts are disabled.
		1	Bank 4 interrupts are enabled.
3	EN3		Bank 3 interrupt enable is used to disable or enable the bank 3 interrupts (events from GP3[15-0])
		0	Bank 3 interrupts are disabled.
		1	Bank 3 interrupts are enabled.
2	EN2		Bank 2 interrupt enable is used to disable or enable the bank 2 interrupts (events from GP2[15-0]).
		0	Bank 2 interrupts are disabled.
		1	Bank 2 interrupts are enabled.
1	EN1		Bank 1 interrupt enable is used to disable or enable the bank 1 interrupts (events from GP1[15-0]).
		0	Bank 1 interrupts are disabled.
		1	Bank 1 interrupts are enabled.
0	EN0		Bank 0 interrupt enable is used to disable or enable the bank 0 interrupts (events from GP0[15-0]).
		0	Bank 0 interrupts are disabled.
		1	Bank 0 interrupts are enabled.

Copyright © 2013–2016, Texas Instruments Incorporated

19.3.3 GPIO Direction Registers (DIR*n*)

The GPIO direction register (DIR*n*) determines if GPIO pin *j* in GPIO bank *k* is an input or an output. Each of the GPIO banks may have up to 16 GPIO pins. By default, all the GPIO pins are configured as inputs (bit value = 1). The GPIO direction register (DIR01) is shown in Figure 19-4, DIR23 is shown in Figure 19-5, DIR45 is shown in Figure 19-6, DIR67 is shown in Figure 19-7, DIR8 is shown in Figure 19-8, and described in Table 19-5. See Table 19-1 to determine the DIR*n* bit associated with each GPIO bank and pin number.

Figure 19-4. GPIO Banks 0 and 1 Direction Register (DIR01)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP1P15	GP1P14	GP1P13	GP1P12	GP1P11	GP1P10	GP1P9	GP1P8	GP1P7	GP1P6	GP1P5	GP1P4	GP1P3	GP1P2	GP1P1	GP1P0
R/W-1															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP0P15	GP0P14	GP0P13	GP0P12	GP0P11	GP0P10	GP0P9	GP0P8	GP0P7	GP0P6	GP0P5	GP0P4	GP0P3	GP0P2	GP0P1	GP0P0
R/W-1															

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

Figure 19-5. GPIO Banks 2 and 3 Direction Register (DIR23)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP3P15	GP3P14	GP3P13	GP3P12	GP3P11	GP3P10	GP3P9	GP3P8	GP3P7	GP3P6	GP3P5	GP3P4	GP3P3	GP3P2	GP3P1	GP3P0
R/W-1															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP2P15	GP2P14	GP2P13	GP2P12	GP2P11	GP2P10	GP2P9	GP2P8	GP2P7	GP2P6	GP2P5	GP2P4	GP2P3	GP2P2	GP2P1	GP2P0
R/W-1															

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

Figure 19-6. GPIO Banks 4 and 5 Direction Register (DIR45)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP5P15	GP5P14	GP5P13	GP5P12	GP5P11	GP5P10	GP5P9	GP5P8	GP5P7	GP5P6	GP5P5	GP5P4	GP5P3	GP5P2	GP5P1	GP5P0
R/W-1															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP4P15	GP4P14	GP4P13	GP4P12	GP4P11	GP4P10	GP4P9	GP4P8	GP4P7	GP4P6	GP4P5	GP4P4	GP4P3	GP4P2	GP4P1	GP4P0
R/W-1															

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

Figure 19-7. GPIO Banks 6 and 7 Direction Register (DIR67)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP7P15	GP7P14	GP7P13	GP7P12	GP7P11	GP7P10	GP7P9	GP7P8	GP7P7	GP7P6	GP7P5	GP7P4	GP7P3	GP7P2	GP7P1	GP7P0
R/W-1															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP6P15	GP6P14	GP6P13	GP6P12	GP6P11	GP6P10	GP6P9	GP6P8	GP6P7	GP6P6	GP6P5	GP6P4	GP6P3	GP6P2	GP6P1	GP6P0
R/W-1															

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

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 19-8. GPIO Bank 8 Direction Register (DIR8)

31															16
Reserved															
R/W-1															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP8P15	GP8P14	GP8P13	GP8P12	GP8P11	GP8P10	GP8P9	GP8P8	GP8P7	GP8P6	GP8P5	GP8P4	GP8P3	GP8P2	GP8P1	GP8P0
R/W-1															

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

Table 19-5. GPIO Direction Register (DIR*n*) Field Descriptions

Bit	Field	Value	Description
31-0	GP*k*P*j*		Direction of pin GP*k*[*j*]. The GP*k*P*j* bit is used to control the direction (output = 0, input = 1) of pin *j* in GPIO bank*k*.
		0	GP*k*[*j*] is an output.
		1	GP*k*[*j*] is an input.

Copyright © 2013–2016, Texas Instruments Incorporated

19.3.4 GPIO Output Data Registers (OUT_DATA*n*)

The GPIO output data register (OUT_DATA*n*) determines the value driven on the corresponding GPIO pin *j* in GPIO bank *k*, if the pin is configured as an output (DIR*n* = 0). Writes do not affect pins not configured as GPIO outputs. The bits in OUT_DATA*n* are set or cleared by writing directly to this register. A read of OUT_DATA*n* returns the value of the register not the value at the pin (that might be configured as an input). The GPIO output data register (OUT_DATA01) is shown in Figure 19-9, OUT_DATA23 is shown in Figure 19-10, OUT_DATA45 is shown in Figure 19-11, OUT_DATA67 is shown in Figure 19-12, OUT_DATA8 is shown in Figure 19-13, and described in Table 19-6. See Table 19-1 to determine the OUT_DATA*n* bit associated with each GPIO bank and pin number.

Figure 19-9. GPIO Banks 0 and 1 Output Data Register (OUT_DATA01)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP1P15	GP1P14	GP1P13	GP1P12	GP1P11	GP1P10	GP1P9	GP1P8	GP1P7	GP1P6	GP1P5	GP1P4	GP1P3	GP1P2	GP1P1	GP1P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP0P15	GP0P14	GP0P13	GP0P12	GP0P11	GP0P10	GP0P9	GP0P8	GP0P7	GP0P6	GP0P5	GP0P4	GP0P3	GP0P2	GP0P1	GP0P0
R/W-0															

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

Figure 19-10. GPIO Banks 2 and 3 Output Data Register (OUT_DATA23)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP3P15	GP3P14	GP3P13	GP3P12	GP3P11	GP3P10	GP3P9	GP3P8	GP3P7	GP3P6	GP3P5	GP3P4	GP3P3	GP3P2	GP3P1	GP3P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP2P15	GP2P14	GP2P13	GP2P12	GP2P11	GP2P10	GP2P9	GP2P8	GP2P7	GP2P6	GP2P5	GP2P4	GP2P3	GP2P2	GP2P1	GP2P0
R/W-0															

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

Figure 19-11. GPIO Banks 4 and 5 Output Data Register (OUT_DATA45)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP5P15	GP5P14	GP5P13	GP5P12	GP5P11	GP5P10	GP5P9	GP5P8	GP5P7	GP5P6	GP5P5	GP5P4	GP5P3	GP5P2	GP5P1	GP5P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP4P15	GP4P14	GP4P13	GP4P12	GP4P11	GP4P10	GP4P9	GP4P8	GP4P7	GP4P6	GP4P5	GP4P4	GP4P3	GP4P2	GP4P1	GP4P0
R/W-0															

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

Figure 19-12. GPIO Banks 6 and 7 Output Data Register (OUT_DATA67)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP7P15	GP7P14	GP7P13	GP7P12	GP7P11	GP7P10	GP7P9	GP7P8	GP7P7	GP7P6	GP7P5	GP7P4	GP7P3	GP7P2	GP7P1	GP7P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP6P15	GP6P14	GP6P13	GP6P12	GP6P11	GP6P10	GP6P9	GP6P8	GP6P7	GP6P6	GP6P5	GP6P4	GP6P3	GP6P2	GP6P1	GP6P0
R/W-0															

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

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 19-13. GPIO Bank 8 Output Data Register (OUT_DATA8)

31															16
Reserved															
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP8P15	GP8P14	GP8P13	GP8P12	GP8P11	GP8P10	GP8P9	GP8P8	GP8P7	GP8P6	GP8P5	GP8P4	GP8P3	GP8P2	GP8P1	GP8P0
R/W-0															

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

Table 19-6. GPIO Output Data Register (OUT_DATA*n*) Field Descriptions

Bit	Field	Value	Description
31-0	GP*k*P*j*		Output drive state of GP*k*[*j*]. The GP*k*P*j* bit is used to drive the output (low = 0, high = 1) of pin *j* in GPIO bank*k*. The GP*k*P*j* bit is ignored when GP*k*[*j*] is configured as an input.
		0	GP*k*[*j*] is driven low.
		1	GP*k*[*j*] is driven high.

Copyright © 2013–2016, Texas Instruments Incorporated

19.3.5 GPIO Set Data Registers (SET_DATA*n*)

The GPIO set data register (SET_DATA*n*) controls driving high of the corresponding GPIO pin *j* in GPIO bank *k*, if the pin is configured as an output (DIR*n* = 0). Writes do not affect pins not configured as GPIO outputs. Writing a 1 to a specific bit in SET_DATA*n* sets the corresponding GPIO pin *j* in GPIO bank *k*. A read of the B*k*P*j* bit returns the output drive state of the corresponding pin GPIO*k*[*j*]. The GPIO set data register (SET_DATA01) is shown in Figure 19-14, SET_DATA23 is shown in Figure 19-15, SET_DATA45 is shown in Figure 19-16, SET_DATA67 is shown in Figure 19-17, SET_DATA8 is shown in Figure 19-18, and described in Table 19-7. See Table 19-1 to determine the SET_DATA*n* bit associated with each GPIO bank and pin number.

Figure 19-14. GPIO Banks 0 and 1 Set Data Register (SET_DATA01)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP1P15	GP1P14	GP1P13	GP1P12	GP1P11	GP1P10	GP1P9	GP1P8	GP1P7	GP1P6	GP1P5	GP1P4	GP1P3	GP1P2	GP1P1	GP1P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP0P15	GP0P14	GP0P13	GP0P12	GP0P11	GP0P10	GP0P9	GP0P8	GP0P7	GP0P6	GP0P5	GP0P4	GP0P3	GP0P2	GP0P1	GP0P0
R/W-0															

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

Figure 19-15. GPIO Banks 2 and 3 Set Data Register (SET_DATA23)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP3P15	GP3P14	GP3P13	GP3P12	GP3P11	GP3P10	GP3P9	GP3P8	GP3P7	GP3P6	GP3P5	GP3P4	GP3P3	GP3P2	GP3P1	GP3P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP2P15	GP2P14	GP2P13	GP2P12	GP2P11	GP2P10	GP2P9	GP2P8	GP2P7	GP2P6	GP2P5	GP2P4	GP2P3	GP2P2	GP2P1	GP2P0
R/W-0															

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

Figure 19-16. GPIO Banks 4 and 5 Set Data Register (SET_DATA45)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP5P15	GP5P14	GP5P13	GP5P12	GP5P11	GP5P10	GP5P9	GP5P8	GP5P7	GP5P6	GP5P5	GP5P4	GP5P3	GP5P2	GP5P1	GP5P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP4P15	GP4P14	GP4P13	GP4P12	GP4P11	GP4P10	GP4P9	GP4P8	GP4P7	GP4P6	GP4P5	GP4P4	GP4P3	GP4P2	GP4P1	GP4P0
R/W-0															

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

Figure 19-17. GPIO Banks 6 and 7 Set Data Register (SET_DATA67)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP7P15	GP7P14	GP7P13	GP7P12	GP7P11	GP7P10	GP7P9	GP7P8	GP7P7	GP7P6	GP7P5	GP7P4	GP7P3	GP7P2	GP7P1	GP7P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP6P15	GP6P14	GP6P13	GP6P12	GP6P11	GP6P10	GP6P9	GP6P8	GP6P7	GP6P6	GP6P5	GP6P4	GP6P3	GP6P2	GP6P1	GP6P0
R/W-0															

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

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 19-18. GPIO Bank 8 Set Data Register (SET_DATA8)

31															16
Reserved															
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP8P15	GP8P14	GP8P13	GP8P12	GP8P11	GP8P10	GP8P9	GP8P8	GP8P7	GP8P6	GP8P5	GP8P4	GP8P3	GP8P2	GP8P1	GP8P0
R/W-0															

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

Table 19-7. GPIO Set Data Register (SET_DATA*n*) Field Descriptions

Bit	Field	Value	Description
31-0	GP*k*P*j*		Set the output drive state of GP*k*[*j*] to logic high. The GP*k*P*j* bit is used to drive the output high on pin *j* in GPIO bank*k*. The GP*k*P*j* bit is ignored when GP*k*[*j*] is configured as an input. Reading the GP*k*P*j* bit returns the output drive state of GP*k*[*j*].
		0	No effect.
		1	GP*k*[*j*] is set to output logic high.

Copyright © 2013–2016, Texas Instruments Incorporated

19.3.6 GPIO Clear Data Registers (CLR_DATA*n*)

The GPIO clear data register (CLR_DATA*n*) controls clearing low of the corresponding GPIO pin *j* in GPIO bank *k*, if the pin is configured as an output (DIR*n* = 0). Writes do not affect pins not configured as GPIO outputs. Writing a 1 to a specific bit in CLR_DATA*n* resets the corresponding GPIO pin *j* in GPIO bank *k*. A read of the B*k*P*j* bit returns the output drive state of the corresponding pin GPIO*k*[*j*]. The GPIO clear data register (CLR_DATA01) is shown in Figure 19-19, CLR_DATA23 is shown in Figure 19-20, CLR_DATA45 is shown in Figure 19-21, CLR_DATA67 is shown in Figure 19-22, CLR_DATA8 is shown in Figure 19-23, and described in Table 19-8. See Table 19-1 to determine the CLR_DATA*n* bit associated with each GPIO bank and pin number.

Figure 19-19. GPIO Banks 0 and 1 Clear Data Register (CLR_DATA01)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP1P15	GP1P14	GP1P13	GP1P12	GP1P11	GP1P10	GP1P9	GP1P8	GP1P7	GP1P6	GP1P5	GP1P4	GP1P3	GP1P2	GP1P1	GP1P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP0P15	GP0P14	GP0P13	GP0P12	GP0P11	GP0P10	GP0P9	GP0P8	GP0P7	GP0P6	GP0P5	GP0P4	GP0P3	GP0P2	GP0P1	GP0P0
R/W-0															

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

Figure 19-20. GPIO Banks 2 and 3 Clear Data Register (CLR_DATA23)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP3P15	GP3P14	GP3P13	GP3P12	GP3P11	GP3P10	GP3P9	GP3P8	GP3P7	GP3P6	GP3P5	GP3P4	GP3P3	GP3P2	GP3P1	GP3P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP2P15	GP2P14	GP2P13	GP2P12	GP2P11	GP2P10	GP2P9	GP2P8	GP2P7	GP2P6	GP2P5	GP2P4	GP2P3	GP2P2	GP2P1	GP2P0
R/W-0															

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

Figure 19-21. GPIO Banks 4 and 5 Clear Data Register (CLR_DATA45)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP5P15	GP5P14	GP5P13	GP5P12	GP5P11	GP5P10	GP5P9	GP5P8	GP5P7	GP5P6	GP5P5	GP5P4	GP5P3	GP5P2	GP5P1	GP5P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP4P15	GP4P14	GP4P13	GP4P12	GP4P11	GP4P10	GP4P9	GP4P8	GP4P7	GP4P6	GP4P5	GP4P4	GP4P3	GP4P2	GP4P1	GP4P0
R/W-0															

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

Figure 19-22. GPIO Banks 6 and 7 Clear Data Register (CLR_DATA67)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP7P15	GP7P14	GP7P13	GP7P12	GP7P11	GP7P10	GP7P9	GP7P8	GP7P7	GP7P6	GP7P5	GP7P4	GP7P3	GP7P2	GP7P1	GP7P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP6P15	GP6P14	GP6P13	GP6P12	GP6P11	GP6P10	GP6P9	GP6P8	GP6P7	GP6P6	GP6P5	GP6P4	GP6P3	GP6P2	GP6P1	GP6P0
R/W-0															

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

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 19-23. GPIO Bank 8 Clear Data Register (CLR_DATA8)

31															16
Reserved															
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP8P15	GP8P14	GP8P13	GP8P12	GP8P11	GP8P10	GP8P9	GP8P8	GP8P7	GP8P6	GP8P5	GP8P4	GP8P3	GP8P2	GP8P1	GP8P0
R/W-0															

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

Table 19-8. GPIO Clear Data Register (CLR_DATA*n*) Field Descriptions

Bit	Field	Value	Description
31-0	GP*k*P*j*		Clear the output drive state of GP*k*[*j*] to logic low. The GP*k*P*j* bit is used to drive the output low on pin *j* in GPIO bank*k*. The GP*k*P*j* bit is ignored when GP*k*[*j*] is configured as an input. Reading the GP*k*P*j* bit returns the output drive state of GP*k*[*j*].
		0	No effect.
		1	GP*k*[*j*] is set to output logic low.

Copyright © 2013–2016, Texas Instruments Incorporated

19.3.7 GPIO Input Data Registers (IN_DATA*n*)

The current state of the GPIO signals is read using the GPIO input data register (IN_DATA*n*).

- For GPIO signals configured as inputs, reading IN_DATA*n* returns the state of the input signal synchronized to the GPIO peripheral clock.
- For GPIO signals configured as outputs, reading IN_DATA*n* returns the output value being driven by the device.

The GPIO input data register (IN_DATA01) is shown in Figure 19-24, IN_DATA23 is shown in Figure 19-25, IN_DATA45 is shown in Figure 19-26, IN_DATA67 is shown in Figure 19-27, IN_DATA8 is shown in Figure 19-28, and described in Table 19-9. See Table 19-1 to determine the IN_DATA*n* bit associated with each GPIO bank and pin number.

Figure 19-24. GPIO Banks 0 and 1 Input Data Register (IN_DATA01)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP1P15	GP1P14	GP1P13	GP1P12	GP1P11	GP1P10	GP1P9	GP1P8	GP1P7	GP1P6	GP1P5	GP1P4	GP1P3	GP1P2	GP1P1	GP1P0
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP0P15	GP0P14	GP0P13	GP0P12	GP0P11	GP0P10	GP0P9	GP0P8	GP0P7	GP0P6	GP0P5	GP0P4	GP0P3	GP0P2	GP0P1	GP0P0
R-0															

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

Figure 19-25. GPIO Banks 2 and 3 Input Data Register (IN_DATA23)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP3P15	GP3P14	GP3P13	GP3P12	GP3P11	GP3P10	GP3P9	GP3P8	GP3P7	GP3P6	GP3P5	GP3P4	GP3P3	GP3P2	GP3P1	GP3P0
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP2P15	GP2P14	GP2P13	GP2P12	GP2P11	GP2P10	GP2P9	GP2P8	GP2P7	GP2P6	GP2P5	GP2P4	GP2P3	GP2P2	GP2P1	GP2P0
R-0															

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

Figure 19-26. GPIO Banks 4 and 5 Input Data Register (IN_DATA45)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP5P15	GP5P14	GP5P13	GP5P12	GP5P11	GP5P10	GP5P9	GP5P8	GP5P7	GP5P6	GP5P5	GP5P4	GP5P3	GP5P2	GP5P1	GP5P0
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP4P15	GP4P14	GP4P13	GP4P12	GP4P11	GP4P10	GP4P9	GP4P8	GP4P7	GP4P6	GP4P5	GP4P4	GP4P3	GP4P2	GP4P1	GP4P0
R-0															

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

Figure 19-27. GPIO Banks 6 and 7 Input Data Register (IN_DATA67)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP7P15	GP7P14	GP7P13	GP7P12	GP7P11	GP7P10	GP7P9	GP7P8	GP7P7	GP7P6	GP7P5	GP7P4	GP7P3	GP7P2	GP7P1	GP7P0
R-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP6P15	GP6P14	GP6P13	GP6P12	GP6P11	GP6P10	GP6P9	GP6P8	GP6P7	GP6P6	GP6P5	GP6P4	GP6P3	GP6P2	GP6P1	GP6P0
R-0															

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

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 19-28. GPIO Bank 8 Input Data Register (IN_DATA8)

31															16
Reserved															
R-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP8P15	GP8P14	GP8P13	GP8P12	GP8P11	GP8P10	GP8P9	GP8P8	GP8P7	GP8P6	GP8P5	GP8P4	GP8P3	GP8P2	GP8P1	GP8P0
R-0															

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

Table 19-9. GPIO Input Data Register (IN_DATA*n*) Field Descriptions

Bit	Field	Value	Description
31-0	GP*k*P*j*		Status of pin GP*k*[*j*]. Reading the GP*k*P*j* bit returns the state of pin *j* in GPIO bank *k*.
		0	GP*k*[*j*] is logic low.
		1	GP*k*[*j*] is logic high.

Copyright © 2013–2016, Texas Instruments Incorporated

19.3.8 GPIO Set Rising Edge Interrupt Registers (SET_RIS_TRIG*n*)

The GPIO set rising edge trigger interrupt register (SET_RIS_TRIG*n*) enables a rising edge trigger on the GPIO pin to generate a GPIO interrupt. The GPIO set rising edge interrupt register (SET_RIS_TRIG01) is shown in Figure 19-29, SET_RIS_TRIG23 is shown in Figure 19-30, SET_RIS_TRIG45 is shown in Figure 19-31, SET_RIS_TRIG67 is shown in Figure 19-32, SET_RIS_TRIG8 is shown in Figure 19-33, and described in Table 19-10. See Table 19-1 to determine the SET_RIS_TRIG*n* bit associated with each GPIO bank and pin number.

Figure 19-29. GPIO Banks 0 and 1 Set Rise Trigger Register (SET_RIS_TRIG01)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP1P15	GP1P14	GP1P13	GP1P12	GP1P11	GP1P10	GP1P9	GP1P8	GP1P7	GP1P6	GP1P5	GP1P4	GP1P3	GP1P2	GP1P1	GP1P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP0P15	GP0P14	GP0P13	GP0P12	GP0P11	GP0P10	GP0P9	GP0P8	GP0P7	GP0P6	GP0P5	GP0P4	GP0P3	GP0P2	GP0P1	GP0P0
R/W-0															

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

Figure 19-30. GPIO Banks 2 and 3 Set Rise Trigger Register (SET_RIS_TRIG23)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP3P15	GP3P14	GP3P13	GP3P12	GP3P11	GP3P10	GP3P9	GP3P8	GP3P7	GP3P6	GP3P5	GP3P4	GP3P3	GP3P2	GP3P1	GP3P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP2P15	GP2P14	GP2P13	GP2P12	GP2P11	GP2P10	GP2P9	GP2P8	GP2P7	GP2P6	GP2P5	GP2P4	GP2P3	GP2P2	GP2P1	GP2P0
R/W-0															

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

Figure 19-31. GPIO Banks 4 and 5 Set Rise Trigger Register (SET_RIS_TRIG45)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP5P15	GP5P14	GP5P13	GP5P12	GP5P11	GP5P10	GP5P9	GP5P8	GP5P7	GP5P6	GP5P5	GP5P4	GP5P3	GP5P2	GP5P1	GP5P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP4P15	GP4P14	GP4P13	GP4P12	GP4P11	GP4P10	GP4P9	GP4P8	GP4P7	GP4P6	GP4P5	GP4P4	GP4P3	GP4P2	GP4P1	GP4P0
R/W-0															

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

Figure 19-32. GPIO Banks 6 and 7 Set Rise Trigger Register (SET_RIS_TRIG67)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP7P15	GP7P14	GP7P13	GP7P12	GP7P11	GP7P10	GP7P9	GP7P8	GP7P7	GP7P6	GP7P5	GP7P4	GP7P3	GP7P2	GP7P1	GP7P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP6P15	GP6P14	GP6P13	GP6P12	GP6P11	GP6P10	GP6P9	GP6P8	GP6P7	GP6P6	GP6P5	GP6P4	GP6P3	GP6P2	GP6P1	GP6P0
R/W-0															

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

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 19-33. GPIO Bank 8 Set Rise Trigger Register (SET_RIS_TRIG8)

31															16
Reserved															
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP8P15	GP8P14	GP8P13	GP8P12	GP8P11	GP8P10	GP8P9	GP8P8	GP8P7	GP8P6	GP8P5	GP8P4	GP8P3	GP8P2	GP8P1	GP8P0
R/W-0															

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

Table 19-10. GPIO Set Rising Edge Trigger Interrupt Register (SET_RIS_TRIG*n*) Field Descriptions

Bit	Field	Value	Description
31-0	GP*j*P*k*		Enable rising edge trigger interrupt detection on GP*k*[*j*]. Reading the GP*k*P*j* bit in either SET_RIS_TRIG*n* or CLR_RIS_TRIG*n* always returns an indication of whether the rising edge interrupt generation function is enabled for pin GP*k*[*j*]. Therefore, this bit will be one in both registers if the function is enabled, and zero in both registers if the function is disabled.
		0	No effect.
		1	Interrupt is caused by a low-to-high transition on GP*k*[*j*].

Copyright © 2013–2016, Texas Instruments Incorporated

19.3.9 GPIO Clear Rising Edge Interrupt Registers (CLR_RIS_TRIG*n*)

The GPIO clear rising edge trigger interrupt register (CLR_RIS_TRIG*n*) disables the rising edge trigger on the GPIO pin to generate a GPIO interrupt. The GPIO clear rising edge interrupt register (CLR_RIS_TRIG01) is shown in Figure 19-34, CLR_RIS_TRIG23 is shown in Figure 19-35, CLR_RIS_TRIG45 is shown in Figure 19-36, CLR_RIS_TRIG67 is shown in Figure 19-37, CLR_RIS_TRIG8 is shown in Figure 19-38, and described in Table 19-11. See Table 19-1 to determine the CLR_RIS_TRIG*n* bit associated with each GPIO bank and pin number.

Figure 19-34. GPIO Banks 0 and 1 Clear Rise Trigger Register (CLR_RIS_TRIG01)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP1P15	GP1P14	GP1P13	GP1P12	GP1P11	GP1P10	GP1P9	GP1P8	GP1P7	GP1P6	GP1P5	GP1P4	GP1P3	GP1P2	GP1P1	GP1P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP0P15	GP0P14	GP0P13	GP0P12	GP0P11	GP0P10	GP0P9	GP0P8	GP0P7	GP0P6	GP0P5	GP0P4	GP0P3	GP0P2	GP0P1	GP0P0
R/W-0															

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

Figure 19-35. GPIO Banks 2 and 3 Clear Rise Trigger Register (CLR_RIS_TRIG23)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP3P15	GP3P14	GP3P13	GP3P12	GP3P11	GP3P10	GP3P9	GP3P8	GP3P7	GP3P6	GP3P5	GP3P4	GP3P3	GP3P2	GP3P1	GP3P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP2P15	GP2P14	GP2P13	GP2P12	GP2P11	GP2P10	GP2P9	GP2P8	GP2P7	GP2P6	GP2P5	GP2P4	GP2P3	GP2P2	GP2P1	GP2P0
R/W-0															

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

Figure 19-36. GPIO Banks 4 and 5 Clear Rise Trigger Register (CLR_RIS_TRIG45)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP5P15	GP5P14	GP5P13	GP5P12	GP5P11	GP5P10	GP5P9	GP5P8	GP5P7	GP5P6	GP5P5	GP5P4	GP5P3	GP5P2	GP5P1	GP5P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP4P15	GP4P14	GP4P13	GP4P12	GP4P11	GP4P10	GP4P9	GP4P8	GP4P7	GP4P6	GP4P5	GP4P4	GP4P3	GP4P2	GP4P1	GP4P0
R/W-0															

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

Figure 19-37. GPIO Banks 6 and 7 Clear Rise Trigger Register (CLR_RIS_TRIG67)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP7P15	GP7P14	GP7P13	GP7P12	GP7P11	GP7P10	GP7P9	GP7P8	GP7P7	GP7P6	GP7P5	GP7P4	GP7P3	GP7P2	GP7P1	GP7P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP6P15	GP6P14	GP6P13	GP6P12	GP6P11	GP6P10	GP6P9	GP6P8	GP6P7	GP6P6	GP6P5	GP6P4	GP6P3	GP6P2	GP6P1	GP6P0
R/W-0															

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

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 19-38. GPIO Bank 8 Clear Rise Trigger Register (CLR_RIS_TRIG8)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP8P15	GP8P14	GP8P13	GP8P12	GP8P11	GP8P10	GP8P9	GP8P8	GP8P7	GP8P6	GP8P5	GP8P4	GP8P3	GP8P2	GP8P1	GP8P0
R/W-0															

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

Table 19-11. GPIO Clear Rising Edge Interrupt Register (CLR_RIS_TRIG*n*) Field Descriptions

Bit	Field	Value	Description
31-0	GP*k*P*j*		Disable rising edge interrupt detection on GP*k*[*j*]. Reading the GP*k*P*j* bit in either SET_RIS_TRIG*n* or CLR_RIS_TRIG*n* always returns an indication of whether the rising edge interrupt generation function is enabled for GP*k*[*j*]. Therefore, this bit will be one in both registers if the function is enabled, and zero in both registers if the function is disabled.
		0	No effect.
		1	No interrupt is caused by a low-to-high transition on GP*k*[*j*].

Copyright © 2013–2016, Texas Instruments Incorporated

19.3.10 GPIO Set Falling Edge Interrupt Registers (SET_FAL_TRIG*n*)

The GPIO set falling edge trigger interrupt register (SET_FAL_TRIG*n*) enables a falling edge trigger on the GPIO pin to generate a GPIO interrupt. The GPIO set falling edge interrupt register (SET_FAL_TRIG01) is shown in Figure 19-39, SET_FAL_TRIG23 is shown in Figure 19-40, SET_FAL_TRIG45 is shown in Figure 19-41, SET_FAL_TRIG67 is shown in Figure 19-42, SET_FAL_TRIG8 is shown in Figure 19-43, and described in Table 19-12. See Table 19-1 to determine the SET_FAL_TRIG*n* bit associated with each GPIO bank and pin number.

Figure 19-39. GPIO Banks 0 and 1 Set Rise Trigger Register (SET_FAL_TRIG01)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP1P15	GP1P14	GP1P13	GP1P12	GP1P11	GP1P10	GP1P9	GP1P8	GP1P7	GP1P6	GP1P5	GP1P4	GP1P3	GP1P2	GP1P1	GP1P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP0P15	GP0P14	GP0P13	GP0P12	GP0P11	GP0P10	GP0P9	GP0P8	GP0P7	GP0P6	GP0P5	GP0P4	GP0P3	GP0P2	GP0P1	GP0P0
R/W-0															

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

Figure 19-40. GPIO Banks 2 and 3 Set Rise Trigger Register (SET_FAL_TRIG23)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP3P15	GP3P14	GP3P13	GP3P12	GP3P11	GP3P10	GP3P9	GP3P8	GP3P7	GP3P6	GP3P5	GP3P4	GP3P3	GP3P2	GP3P1	GP3P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP2P15	GP2P14	GP2P13	GP2P12	GP2P11	GP2P10	GP2P9	GP2P8	GP2P7	GP2P6	GP2P5	GP2P4	GP2P3	GP2P2	GP2P1	GP2P0
R/W-0															

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

Figure 19-41. GPIO Banks 4 and 5 Set Rise Trigger Register (SET_FAL_TRIG45)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP5P15	GP5P14	GP5P13	GP5P12	GP5P11	GP5P10	GP5P9	GP5P8	GP5P7	GP5P6	GP5P5	GP5P4	GP5P3	GP5P2	GP5P1	GP5P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP4P15	GP4P14	GP4P13	GP4P12	GP4P11	GP4P10	GP4P9	GP4P8	GP4P7	GP4P6	GP4P5	GP4P4	GP4P3	GP4P2	GP4P1	GP4P0
R/W-0															

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

Figure 19-42. GPIO Banks 6 and 7 Set Rise Trigger Register (SET_FAL_TRIG67)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP7P15	GP7P14	GP7P13	GP7P12	GP7P11	GP7P10	GP7P9	GP7P8	GP7P7	GP7P6	GP7P5	GP7P4	GP7P3	GP7P2	GP7P1	GP7P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP6P15	GP6P14	GP6P13	GP6P12	GP6P11	GP6P10	GP6P9	GP6P8	GP6P7	GP6P6	GP6P5	GP6P4	GP6P3	GP6P2	GP6P1	GP6P0
R/W-0															

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

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 19-43. GPIO Bank 8 Set Rise Trigger Register (SET_FAL_TRIG8)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP8P15	GP8P14	GP8P13	GP8P12	GP8P11	GP8P10	GP8P9	GP8P8	GP8P7	GP8P6	GP8P5	GP8P4	GP8P3	GP8P2	GP8P1	GP8P0
R/W-0															

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

Table 19-12. GPIO Set Falling Edge Trigger Interrupt Register (SET_FAL_TRIG*n*) Field Descriptions

Bit	Field	Value	Description
31-0	GP*k*P*j*		Enable falling edge trigger interrupt detection on GP*k*[*j*]. Reading the GP*k*P*j* bit in either SET_FAL_TRIG*n* or CLR_FAL_TRIG*n* always returns an indication of whether the falling edge interrupt generation function is enabled for pin GP*k*[*j*]. Therefore, this bit will be one in both registers if the function is enabled, and zero in both registers if the function is disabled.
		0	No effect.
		1	Interrupt is caused by a high-to-low transition on GP*k*[*j*].

Copyright © 2013–2016, Texas Instruments Incorporated

19.3.11 GPIO Clear Falling Edge Interrupt Registers (CLR_FAL_TRIG*n*)

The GPIO clear falling edge trigger interrupt register (CLR_FAL_TRIG*n*) disables the falling edge trigger on the GPIO pin to generate a GPIO interrupt. The GPIO clear falling edge interrupt register (CLR_FAL_TRIG01) is shown in Figure 19-44, CLR_FAL_TRIG23 is shown in Figure 19-45, CLR_FAL_TRIG45 is shown in Figure 19-46, CLR_FAL_TRIG67 is shown in Figure 19-47, CLR_FAL_TRIG8 is shown in Figure 19-48, and described in Table 19-13. See Table 19-1 to determine the CLR_FAL_TRIG*n* bit associated with each GPIO bank and pin number.

Figure 19-44. GPIO Banks 0 and 1 Clear Rise Trigger Register (CLR_FAL_TRIG01)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP1P15	GP1P14	GP1P13	GP1P12	GP1P11	GP1P10	GP1P9	GP1P8	GP1P7	GP1P6	GP1P5	GP1P4	GP1P3	GP1P2	GP1P1	GP1P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP0P15	GP0P14	GP0P13	GP0P12	GP0P11	GP0P10	GP0P9	GP0P8	GP0P7	GP0P6	GP0P5	GP0P4	GP0P3	GP0P2	GP0P1	GP0P0
R/W-0															

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

Figure 19-45. GPIO Banks 2 and 3 Clear Rise Trigger Register (CLR_FAL_TRIG23)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP3P15	GP3P14	GP3P13	GP3P12	GP3P11	GP3P10	GP3P9	GP3P8	GP3P7	GP3P6	GP3P5	GP3P4	GP3P3	GP3P2	GP3P1	GP3P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP2P15	GP2P14	GP2P13	GP2P12	GP2P11	GP2P10	GP2P9	GP2P8	GP2P7	GP2P6	GP2P5	GP2P4	GP2P3	GP2P2	GP2P1	GP2P0
R/W-0															

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

Figure 19-46. GPIO Banks 4 and 5 Clear Rise Trigger Register (CLR_FAL_TRIG45)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP5P15	GP5P14	GP5P13	GP5P12	GP5P11	GP5P10	GP5P9	GP5P8	GP5P7	GP5P6	GP5P5	GP5P4	GP5P3	GP5P2	GP5P1	GP5P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP4P15	GP4P14	GP4P13	GP4P12	GP4P11	GP4P10	GP4P9	GP4P8	GP4P7	GP4P6	GP4P5	GP4P4	GP4P3	GP4P2	GP4P1	GP4P0
R/W-0															

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

Figure 19-47. GPIO Banks 6 and 7 Clear Rise Trigger Register (CLR_FAL_TRIG67)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP7P15	GP7P14	GP7P13	GP7P12	GP7P11	GP7P10	GP7P9	GP7P8	GP7P7	GP7P6	GP7P5	GP7P4	GP7P3	GP7P2	GP7P1	GP7P0
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP6P15	GP6P14	GP6P13	GP6P12	GP6P11	GP6P10	GP6P9	GP6P8	GP6P7	GP6P6	GP6P5	GP6P4	GP6P3	GP6P2	GP6P1	GP6P0
R/W-0															

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

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 19-48. GPIO Bank 8 Clear Rise Trigger Register (CLR_FAL_TRIG8)

31															16
Reserved															
R/W-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP8P15	GP8P14	GP8P13	GP8P12	GP8P11	GP8P10	GP8P9	GP8P8	GP8P7	GP8P6	GP8P5	GP8P4	GP8P3	GP8P2	GP8P1	GP8P0
R/W-0															

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

Table 19-13. GPIO Clear Falling Edge Interrupt Register (CLR_FAL_TRIG*n*) Field Descriptions

Bit	Field	Value	Description
31-0	GP*k*P*j*		Disable falling edge interrupt detection on GP*k*[*j*]. Reading the GP*k*P*j* bit in either SET_FAL_TRIG*n* or CLR_FAL_TRIG*n* always returns an indication of whether the falling edge interrupt generation function is enabled for GP*k*[*j*]. Therefore, this bit will be one in both registers if the function is enabled, and zero in both registers if the function is disabled.
		0	No effect.
		1	No interrupt is caused by a high-to-low transition on GP*k*[*j*].

Copyright © 2013–2016, Texas Instruments Incorporated

19.3.12 GPIO Interrupt Status Registers (INTSTAT*n*)

The status of GPIO interrupt events can be monitored by reading the GPIO interrupt status register (INTSTAT*n*). In the associated bit position, pending GPIO interrupts are indicated with a logic 1 and GPIO interrupts that are not pending are indicated with a logic 0. The GPIO interrupt status register (INTSTAT01) is shown in Figure 19-49, INTSTAT23 is shown in Figure 19-50, INTSTAT45 is shown in Figure 19-51, INTSTAT67 is shown in Figure 19-52, INTSTAT8 is shown in Figure 19-53, and described in Table 19-14. See Table 19-1 to determine the INTSTAT*n* bit associated with each GPIO bank and pin number.

Figure 19-49. GPIO Banks 0 and 1 Interrupt Status Register (INTSTAT01)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP1P15	GP1P14	GP1P13	GP1P12	GP1P11	GP1P10	GP1P9	GP1P8	GP1P7	GP1P6	GP1P5	GP1P4	GP1P3	GP1P2	GP1P1	GP1P0
R/W1C-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP0P15	GP0P14	GP0P13	GP0P12	GP0P11	GP0P10	GP0P9	GP0P8	GP0P7	GP0P6	GP0P5	GP0P4	GP0P3	GP0P2	GP0P1	GP0P0
R/W1C-0															

LEGEND: R/W = Read/Write; W1C = Write 1 to clear bit (writing 0 has no effect); -*n* = value after reset

Figure 19-50. GPIO Banks 2 and 3 Interrupt Status Register (INTSTAT23)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP3P15	GP3P14	GP3P13	GP3P12	GP3P11	GP3P10	GP3P9	GP3P8	GP3P7	GP3P6	GP3P5	GP3P4	GP3P3	GP3P2	GP3P1	GP3P0
R/W1C-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP2P15	GP2P14	GP2P13	GP2P12	GP2P11	GP2P10	GP2P9	GP2P8	GP2P7	GP2P6	GP2P5	GP2P4	GP2P3	GP2P2	GP2P1	GP2P0
R/W1C-0															

LEGEND: R/W = Read/Write; W1C = Write 1 to clear bit (writing 0 has no effect); -*n* = value after reset

Figure 19-51. GPIO Banks 4 and 5 Interrupt Status Register (INTSTAT45)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP5P15	GP5P14	GP5P13	GP5P12	GP5P11	GP5P10	GP5P9	GP5P8	GP5P7	GP5P6	GP5P5	GP5P4	GP5P3	GP5P2	GP5P1	GP5P0
R/W1C-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP4P15	GP4P14	GP4P13	GP4P12	GP4P11	GP4P10	GP4P9	GP4P8	GP4P7	GP4P6	GP4P5	GP4P4	GP4P3	GP4P2	GP4P1	GP4P0
R/W1C-0															

LEGEND: R/W = Read/Write; W1C = Write 1 to clear bit (writing 0 has no effect); -*n* = value after reset

Figure 19-52. GPIO Banks 6 and 7 Interrupt Status Register (INTSTAT67)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GP7P15	GP7P14	GP7P13	GP7P12	GP7P11	GP7P10	GP7P9	GP7P8	GP7P7	GP7P6	GP7P5	GP7P4	GP7P3	GP7P2	GP7P1	GP7P0
R/W1C-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP6P15	GP6P14	GP6P13	GP6P12	GP6P11	GP6P10	GP6P9	GP6P8	GP6P7	GP6P6	GP6P5	GP6P4	GP6P3	GP6P2	GP6P1	GP6P0
R/W1C-0															

LEGEND: R/W = Read/Write; W1C = Write 1 to clear bit (writing 0 has no effect); -*n* = value after reset

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 19-53. GPIO Bank 8 Interrupt Status Register (INTSTAT8)

31															16
Reserved															
R/W1C-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GP8P15	GP8P14	GP8P13	GP8P12	GP8P11	GP8P10	GP8P9	GP8P8	GP8P7	GP8P6	GP8P5	GP8P4	GP8P3	GP8P2	GP8P1	GP8P0
R/W1C-0															

LEGEND: R/W = Read/Write; W1C = Write 1 to clear bit (writing 0 has no effect); -*n* = value after reset

Table 19-14. GPIO Interrupt Status Register (INTSTAT*n*) Field Descriptions

Bit	Field	Value	Description
31-0	GP*k*P*j*		Interrupt status of GP*k*[*j*]. The GP*k*P*j* bit is used to monitor pending GPIO interrupts on pin *j* of GPIO bank *k*. Write a 1 to the GP*k*P*j* bit to clear the status bit; a write of 0 has no effect.
		0	No pending interrupt on GP*k*[*j*].
		1	Pending interrupt on GP*k*[*j*].

Copyright © 2013–2016, Texas Instruments Incorporated

Host Port Interface (HPI)

This chapter describes the host port interface (HPI).

Topic **Page**

20.1 Introduction **817**
20.2 Architecture **820**
20.3 Registers **840**

Copyright © 2013–2016, Texas Instruments Incorporated

20.1 Introduction

The host port interface (HPI) provides a parallel port interface through which an external host processor can directly access the processor's resources (configuration and program/data memories). The external host device is asynchronous to the CPU clock and functions as a master to the HPI interface. The HPI enables a host device and the processor to exchange information via internal or external memory. Dedicated address (HPIA) and data (HPID) registers within the HPI provide the data path between the external host interface and the processor resources. An HPI control register (HPIC) is available to the host and the CPU for various configuration and interrupt functions.

20.1.1 Purpose of the Peripheral

The HPI enables an external host processor (host) to directly access program/data memory on the processor using a parallel interface. The primary purpose is to provide a mechanism to move data to and from the processor. In addition to data transfer, the host can also use the HPI to bootload the processor by downloading program and data information to the processor's memory after power-up.

20.1.2 Features

The HPI supports the following features:

- Multiplexed address/data
- Dual 16-bit halfword cycle access (internal data word is 32-bits wide)
- 16-bit-wide host data bus interface
- Internal data bursting using 8-word read and write first-in, first-out (FIFO) buffers
- HPI control register (HPIC) accessible by both the CPU and the external host
- HPI address register (HPIA) accessible by both the CPU and the external host
- Separate HPI address registers for read (HPIAR) and write (HPIAW) with configurable option for operating as a single HPI address register
- HPI data register (HPID)/FIFOs providing data-path between external host interface and CPU resources
- Multiple strobes and control signals to allow flexible host connection
- Asynchronous UHPI_HRDY output to allow the HPI to insert wait states to the host
- Software control of data prefetching to the HPID/FIFOs
- Processor-to-Host interrupt output signal controlled by HPIC accesses
- Host-to-Processor interrupt controlled by HPIC accesses
- Register controlled HPIA and HPIC ownership and FIFO timeout
- Memory-mapped peripheral identification register (PID)
- Bus holders on host data and address buses (these are actually external to HPI module)

Copyright © 2013–2016, Texas Instruments Incorporated

20.1.3 Functional Block Diagram

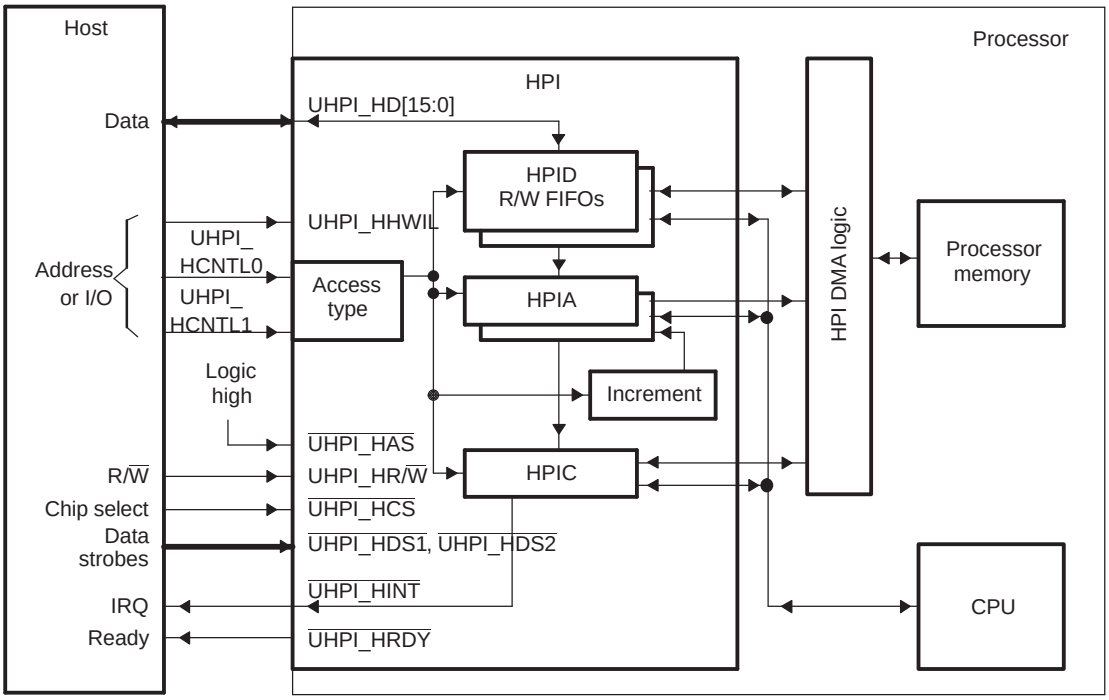
Figure 20-1 is a high-level block diagram showing how the HPI connects a host (left side of figure) and the processor internal memory (right side of figure). Host activity is asynchronous to the internal processor clock that drives the HPI. The host functions as a master to the HPI. When HPI resources are temporarily busy or unavailable, the HPI communicates this to the host by deasserting the HPI ready (UHPI_HRDY) output signal.

The HPI supports multiplexed operation meaning the data bus is used for both address and data. Each host cycle consists of two consecutive 16-bit transfers. When the host drives an address on the bus, the address is stored in a 32-bit address register (HPIA) in the HPI, so that the bus can then be used for data. The HPI contains two address registers (HPIAR and HPIAW), which can be used as separate address registers for read accesses and write accesses (for details, see Section 20.2.6.1).

A control register (HPIC) is accessible by the CPU and the host. The CPU uses HPIC to send an interrupt request to the host, to clear an interrupt request from the host, and to monitor the HPI. The host uses HPIC to configure and monitor the HPI, to send an interrupt request to the CPU, and to clear an interrupt request from the CPU.

Data flow between the host and the HPI uses a temporary storage register, the 32-bit data register (HPID). Data arriving from the host is held in HPID until the data can be stored elsewhere in the processor. Data to be sent to the host is held in HPID until the HPI is ready to perform the transfer. When address autoincrementing is used, read and write FIFOs are used to store burst data. If autoincrementing is not used, the FIFO memory acts as a single register (only one location is used).

Figure 20-1. HPI Block Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

20.1.4 Industry Standard(s) Compliance Statement

The HPI is not an industry standard interface that is developed and monitored by an international organization. It is a generic parallel interface that can be configured to gluelessly interface to a variety of parallel devices.

20.1.5 Terminology Used in This Document

The following is a brief explanation of some terms used in this document:

Term	Meaning
host	External host device
HPI DMA logic	Logic used to communicate between the HPI and the DMA system that moves data to and from memory. This is independent of the EDMA system on the processor
processor	Entire system-on-chip

Copyright © 2013–2016, Texas Instruments Incorporated

20.2 Architecture

20.2.1 Clock Control

For detailed information on the PLLs and clock distribution on the processor, see the *Phase-Locked Loop Controller (PLLC)* chapter.

20.2.2 Memory Map

The HPI can be used by the host to access on-chip device memory, peripheral, and memory-mapped registers. See your device-specific data manual for more detailed information.

20.2.3 Signal Descriptions

Table 20-1 shows the a description of the HPI signals.

Table 20-1. HPI Pins

Pin	Type	Host Connection	Function
UHPI_HCNTL[1:0]	I	Address or control pins	**HPI access control inputs**. The HPI latches the logic levels of these pins on the falling edge of internal $\overline{\text{HSTRB}}$ (for details about internal $\overline{\text{HSTRB}}$, see Section 20.2.6.4). The four binary states of these pins determine the access type of the current transfer (HPIC, HPIA, HPID with and without autoincrementing).
$\overline{\text{UHPI_HCS}}$	I	Chip select pin	**HPI chip select**. $\overline{\text{UHPI_HCS}}$ must be low for the HPI to be selected by the host. $\overline{\text{UHPI_HCS}}$ can be kept low between accesses. $\overline{\text{UHPI_HCS}}$ normally precedes an active $\overline{\text{UHPI_HDS}}$ (data strobe) signal, but can be connected to a $\overline{\text{UHPI_HDS}}$ pin for simultaneous select and strobe activity.
UHPI_HR/$\overline{\text{W}}$	I	R/W strobe pin	**HPI read/write**. On the falling edge of internal $\overline{\text{HSTRB}}$, UHPI_HR/$\overline{\text{W}}$ indicates whether the current access is to be a read or write operation. Driving UHPI_HR/$\overline{\text{W}}$ high indicates the transfer is a read from the HPI, while driving UHPI_HR/$\overline{\text{W}}$ low indicates a write to the HPI.
UHPI_HHWIL	I	Address or control pins	**Halfword identification line**. The host uses UHPI_HHWIL to identify the first and second halfwords of the host cycle. UHPI_HHWIL must be driven low for the first halfword and high for the second halfword.
$\overline{\text{UHPI_HAS}}$	I	None	**Address strobe**. Connect to logic high.
$\overline{\text{UHPI_HINT}}$	O/Z	Interrupt pin	**Host Interrupt**. The CPU can interrupt the host processor by writing a 1 to the HINT bit of HPIC. Before subsequent $\overline{\text{UHPI_HINT}}$ interrupts can occur, the host must acknowledge interrupts by writing a 1 to the HINT bit. This pin is active-low (that is, when an interrupt is asserted from the host, the state of this signal is low) and inverted from the HINT bit value in HPIC.
$\overline{\text{UHPI_HDS1}}$ and $\overline{\text{UHPI_HDS2}}$	I	Read strobe and write strobe pins or any data strobe pin	**HPI data strobe pins**. These pins are used for strobing data in and out of the HPI (for data strobing details, see Section 20.2.6.4). The direction of the data transfer depends on the logic level of the UHPI_HR/$\overline{\text{W}}$ signal. The $\overline{\text{UHPI_HDS}}$ signals are also used to latch control information on the falling edge. During an HPID write access, data is latched into the HPID register on the rising edge of $\overline{\text{UHPI_HDS}}$. During read operations, these pins act as output-enable pins of the host data bus.
UHPI_HD[15:0]	I/O/Z	Data bus	**HPI data bus**. The HPI data bus carries the address and data to/from the HPI.
$\overline{\text{UHPI_HRDY}}$	O/Z	Asynchronous ready pin	**HPI-ready signal**. When the HPI drives $\overline{\text{UHPI_HRDY}}$ low, the host has permission to complete the current host cycle. When the HPI drives $\overline{\text{UHPI_HRDY}}$ high, the HPI is not ready for the current host cycle to complete.

Copyright © 2013–2016, Texas Instruments Incorporated

20.2.4 Pin Multiplexing and General-Purpose I/O Control Blocks

Extensive pin multiplexing is used to accommodate the largest number of peripheral functions in the smallest possible package. Pin multiplexing is controlled using a combination of hardware configuration at device reset and software programmable register settings. See your device-specific data manual to determine how pin multiplexing affects the HPI.

The HPI supports general-purpose I/O (GPIO) capability on all pins. All HPI pins may be enabled for GPIO mode when the HPI is disabled via the HPIENA bit in the chip configuration 1 register (CFGCHIP1) in the *System Configuration (SYSCFG) Module* chapter.

When the HPI is enabled, the pins not being used for host accesses may be configured as general-purpose I/O.

20.2.4.1 Treatment of Optional Pins when Configured as General-Purpose I/O

Certain pins are optional, but if used to interface to the external hosts, the pins are inputs to the HPI. For the purpose of host accesses, the HPI treats these pins as if they were driven to the values listed in Table 20-2.

Table 20-2. Value on Optional Pins when Configured as General-Purpose I/O

Pin	GPIO Enable Bit(s)	When Enabled as GPIO, Treated as Driven:
UHPI_HD[15:8]	GPIOEN8	0
UHPI_HD[7:0]	GPIOEN7	0
UHPI_HCNTL0	GPIOEN1	1
UHPI_HCNTL1	GPIOEN1	1
$\overline{\text{UHPI_HAS}}$	GPIOEN2	1

20.2.4.2 General-Purpose I/O Programmer's Model

For each HPI pin, there are three bits that control this pin as general-purpose I/O (GPIO):

- Enable: GPIO_EN.GPIOEN[xx]
- Direction: GPIO_DIR*n*.DIR[yy]
- Data: GPIO_DAT*n*.DIR[yy]

For example, the $\overline{\text{UHPI_HAS}}$ pin is enabled with the GPIO_EN.GPIOEN2 bit. In the default setting, the GPIO_EN.GPIOEN2 bit is cleared to 0; therefore, the $\overline{\text{UHPI_HAS}}$ pin functions as the host address strobe.

Enabling this pin for GPIO, by setting the GPIO_EN.GPIOEN2 bit to 1 does two things:

- Transfers control of the $\overline{\text{UHPI_HAS}}$ pin to GPIO direction and data bits.
- Drives a 1 into the $\overline{\text{UHPI_HAS}}$ pin input of the external host interface block (regardless of the actual pin value).

Once enabled as GPIO, the direction of the $\overline{\text{UHPI_HAS}}$ pin is controlled by the GPIO_DIR2.HASZ bit. If this bit is set to 1, the pin will be driven as an output; if this bit is cleared to 0, the pin will be an input.

Once the direction of the $\overline{\text{UHPI_HAS}}$ pin is set, the data value is either written to or read from the GPIO_DAT2.HASZ bit. If the pin was configured as an output, then writing to this bit determines the value to drive out the pin. Reading from this bit will return the value written.

When the GPIO_DIR2.HASZ bit is cleared to 0, configuring the $\overline{\text{UHPI_HAS}}$ pin as an input, writing to the GPIO_DAT2.HASZ bit has no effect. Reading from this bit will return the value on the $\overline{\text{UHPI_HAS}}$ pin.

Copyright © 2013–2016, Texas Instruments Incorporated

NOTE: Note that you cannot preload a value into the DAT field before configuring the pin as an output. This means that when switching the pin from input to output, the pin will initially drive the last value input back out on the pin. Then the DAT bit can be written to change the value on the pin. If the intermediate value between writing to DIR and writing to DAT will cause a problem at the system level, it is suggested to use another general-purpose I/O pin on the device.

20.2.5 Protocol Description

The HPI does not conform to any industry standard protocol.

20.2.6 Operation

20.2.6.1 Using the Address Registers

The HPI contains two 32-bit address registers: one for read operations (HPIAR) and one for write operations (HPIAW). These roles are unchanging from the viewpoint of the HPI logic. The HPI DMA logic gets the address from HPIAR when reading from processor resources (see Section 20.2.2) and gets the address from HPIAW when writing to processor resources (see Section 20.2.2).

However, unlike the HPI logic, the host can choose how to interact with the two HPI address registers. Using the DUALHPIA bit in the HPI control register (HPIC), the host determines whether HPIAR and HPIAW act as a single 32-bit register (single-HPIA mode) or as two independent 32-bit registers (dual-HPIA mode).

Note that the addresses loaded into the HPI address registers must be byte addresses, and must be 32-bit word aligned (with the least-significant two bits equal to zero).

20.2.6.1.1 Single-HPIA Mode

When DUALHPIA = 0 in HPIC, HPIAR and HPIAW become a single HPI address register (HPIA) from the perspective of the host. In this mode:

- A host HPIA write cycle (UHPI_HCNTL[1:0] = 10b, UHPI_HR/$\overline{W}$ = 0) updates HPIAR and HPIAW with the same value.
- Both HPI address registers are incremented during autoincrement read/write cycles (UHPI_HCNTL[1:0] = 01b).
- An HPIA read cycle (UHPI_HCNTL[1:0] = 10b, UHPI_HR/$\overline{W}$ = 1) returns the content of HPIAR, which should be identical to the content of HPIAW.

To maintain consistency between the contents of HPIAR and HPIAW, the host should always reinitialize the HPI address registers after changing the state of the DUALHPIA bit. In addition, when DUALHPIA = 0, the host must always reinitialize the HPI address registers when it changes the data direction (from an HPID read cycle to an HPID write cycle, or conversely). Otherwise, the memory location accessed by the HPI DMA logic might not be the location intended by the host.

Copyright © 2013–2016, Texas Instruments Incorporated

20.2.6.1.2 Dual-HPIA Mode

When DUALHPIA = 1 in HPIC, HPIAR and HPIAW are two independent HPI address registers from the perspective of the host. In this mode:

- A host HPIA access (UHPI_HCNTL[1:0] = 10b) reads/updates either HPIAR or HPIAW, depending on the value of the HPIA read/write select (HPIASEL) bit in HPIC. This bit is programmed by the host. While HPIASEL = 1, only HPIAR is read or updated by the host. While HPIASEL = 0, only HPIAW is read or updated by the host. The HPIASEL bit is only meaningful in the dual-HPIA mode.

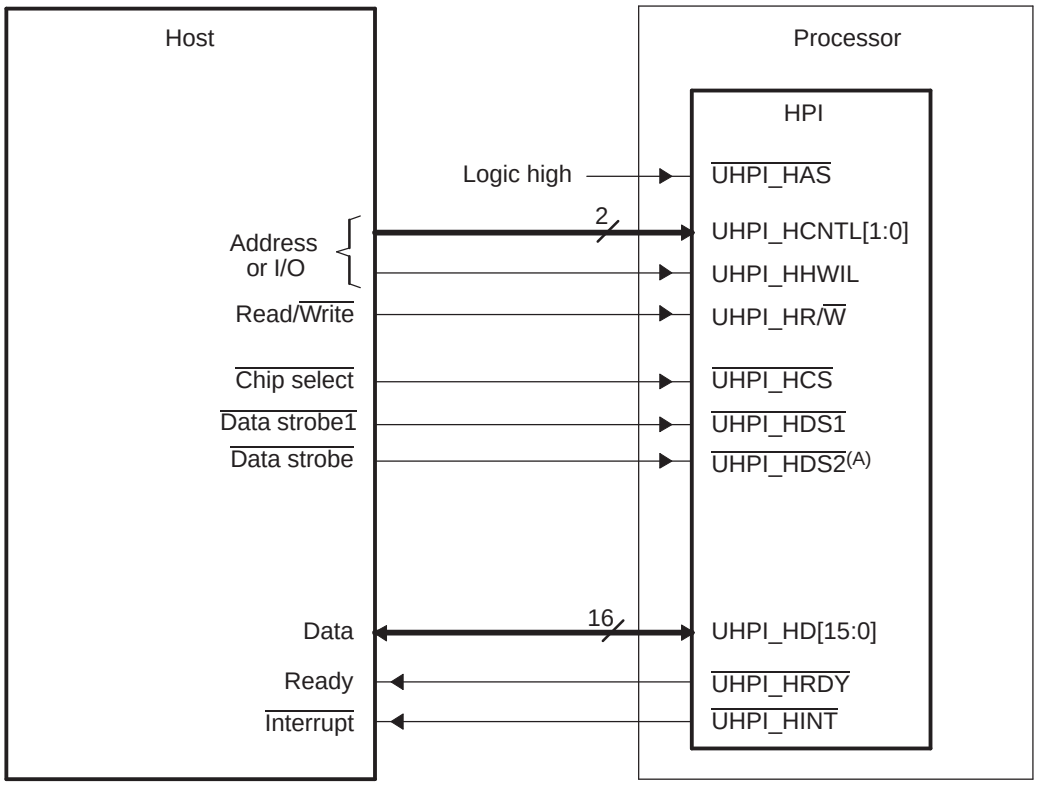
NOTE: The HPIASEL bit does not affect the HPI DMA logic. Regardless of the value of HPIASEL, the HPI DMA logic uses HPIAR when reading from memory and HPIAW when writing to memory.

- A host HPID access with autoincrementing (UHPI_HCNTL[1:0] = 01b) causes only the relevant HPIA value to be incremented to the next consecutive memory address. In an autoincrement read cycle, HPIAR is incremented after it has been used to perform the current read from memory. In an autoincrement write cycle, HPIAW is incremented after it has been used for the write operation.

20.2.6.2 Host-HPI Signal Connections

Figure 20-2 shows an example of a signal connection between the HPI and a host.

Figure 20-2. Example of Host-Processor Signal Connections



A Data strobing options are given in Section 20.2.6.4

Copyright © 2013–2016, Texas Instruments Incorporated

20.2.6.3 HPI Configuration and Data Flow

The host accomplishes a multiplexed access in the following manner:

1. The host writes to the HPI control register (HPIC) to properly configure the HPI. Typically, this means programming the halfword order bit (HWOB) and the HPIA-related bits (DUALHPIA and HPIASEL). This step is normally performed once before the initial data access.
2. The host writes the desired internal processor memory address to an address register (HPIAR and/or HPIAW). For an introduction to the two HPI address registers and the two ways the host can interact with them, see Section 20.2.6.1.
3. The host reads from or writes to the data register (HPID). Data transfers between HPID and the internal memory of the processor are handled by the HPI DMA logic and are transparent to the CPU.

Each step of the access uses the same bus. Therefore, the host must drive the appropriate levels on the UHPI_HCNTL1 and UHPI_HCNTL0 signals to indicate which register is to be accessed. The host must also drive the appropriate level on the UHPI_HR/$\overline{W}$ signal to indicate the data direction (read or write) and must drive other control signals as appropriate. When HPI resources are temporarily busy or unavailable, the HPI can communicate this to the host by deasserting the HPI-ready ($\overline{UHPI_HRDY}$) output signal.

When performing an access, the HPI first latches the levels on UHPI_HCNTL[1:0], UHPI_HR/$\overline{W}$, and other control signals. This latching can occur on the falling edge of the internal strobe signal (for details, see Section 20.2.6.4). After the control information is latched, the HPI initiates an access based on the control signals.

If the host wants to read data from processor resources (see Section 20.2.2), the HPI DMA logic reads the resource address from HPIAR and retrieves the data from the addressed memory location. When the data has been placed in HPID, the HPI drives the data onto its UHPI_HD bus. The $\overline{UHPI_HRDY}$ signal informs the host whether the data on the UHPI_HD bus is valid ($\overline{UHPI_HRDY}$ low) or not valid yet ($\overline{UHPI_HRDY}$ high). When the data is valid, the host should latch the data and drive the connected data strobe ($\overline{UHPI_HDS1}$ or $\overline{UHPI_HDS2}$) inactive, which, in turn, will cause the internal strobe (internal $\overline{HSTRB}$) signal to transition from low to high.

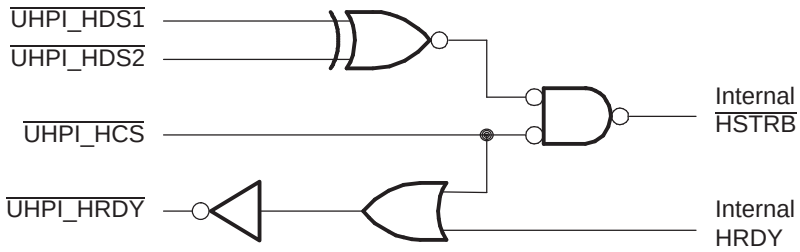
If the host wants to write data to processor resources (see Section 20.2.2), the operation is similar. After the host determines that the HPI is ready to latch the data ($\overline{UHPI_HRDY}$ is low), it must cause internal $\overline{HSTRB}$ to transition from low to high, which causes the data to be latched into HPID. Once the data is in HPID, the HPI DMA logic reads the memory address from HPIAW and transfers the data from HPID to the addressed memory location.

Copyright © 2013–2016, Texas Instruments Incorporated

20.2.6.4 UHPI_HDS2, UHPI_HDS1, and UHPI_HCS: Data Strobing and Chip Selection

As shown in Figure 20-3, the strobing logic is a function of three key inputs: the chip select pin (UHPI_HCS) and two data strobe signals (UHPI_HDS1 and UHPI_HDS2). The internal strobe signal, which is referred to as internal HSTRB throughout this document, functions as the actual strobe signal inside the HPI. UHPI_HCS must be low (HPI selected) during strobe activity on the UHPI_HDS pins. If UHPI_HCS remains high (HPI not selected), activity on the UHPI_HDS pins is ignored.

Figure 20-3. HPI Strobe and Select Logic



Strobe connections between the host and the HPI depend in part on the number and types of strobe pins available on the host. Table 20-3 describes some options for connecting to the UHPI_HDS pins.

Notice in Figure 20-3 that UHPI_HRDY is also gated by UHPI_HCS. If UHPI_HCS goes high (HPI not selected), UHPI_HRDY goes low, regardless of whether the current internal transfer is completed in the processor.

NOTE: The UHPI_HCS input and one UHPI_HDS strobe input can be tied together and driven with a single strobe signal from the host. This technique selects the HPI and provides the strobe, simultaneously. When using this method, be aware that UHPI_HRDY is gated by UHPI_HCS as previously described.

It is not recommended to tie both UHPI_HDS1 and UHPI_HDS2 to static logic levels and use UHPI_HCS as a strobe.

Table 20-3. Options for Connecting Host and HPI Data Strobe Pins

Available Host Data Strobe Pins	Connections to HPI Data Strobe Pins
Host has separate read and write strobe pins, both active-low	Connect one strobe pin to UHPI_HDS1 and the other to UHPI_HDS2 (1). Since such a host might not provide a R/W line, take care to satisfy UHPI_HR/W timings as stated in your device-specific data manual. This could possibly be done using a host address line.
Host has separate read and write strobe pins, both active-high	Connect one strobe pin to UHPI_HDS1 and the other to UHPI_HDS2 (1). Since such a host might not provide a R/W line, take care to satisfy UHPI_HR/W timings as stated in your device-specific data manual. This could possibly be done using a host address line.
Host has one active-low strobe pin	Connect the strobe pin to UHPI_HDS1 or UHPI_HDS2, and connect the other pin to logic-level 1.
Host has one active-high strobe pin	Connect the strobe pin to UHPI_HDS1 or UHPI_HDS2, and connect the other strobe pin to logic-level 0.

(1) The UHPI_HR/W signal could be driven by a host address line in this case.

Copyright © 2013–2016, Texas Instruments Incorporated

20.2.6.5 UHPI_HCNTL[1:0] and UHPI_HR/$\overline{W}$: Indicating the Cycle Type

The cycle type consists of:

- The access type that the host selects by driving the appropriate levels on the UHPI_HCNTL[1:0] pins of the HPI. Table 20-4 describes the four available access types.
- The transfer direction that the host selects with the UHPI_HR/$\overline{W}$ pin. The host must drive the UHPI_HR/$\overline{W}$ signal high (read) or low (write).

A summary of cycle types is in Table 20-5. The HPI samples the UHPI_HCNTL levels at the falling edge of the internal strobe signal $\overline{HSTRB}$.

Table 20-4. Access Types Selectable With the UHPI_HCNTL Signals

UHPI_HCNTL1	UHPI_HCNTL0	Access Type
0	0	**HPIC access.** The host requests to access the HPI control register (HPIC).
0	1	**HPID access with autoincrementing.** The host requests to access the HPI data register (HPID) and to have the appropriate HPI address register (HPIAR and/or HPIAW) automatically incremented by 1 after the access.
1	0	**HPIA access.** The host requests to access the appropriate HPI address register (HPIAR and/or HPIAW).
1	1	**HPID access without autoincrementing.** The host requests to access the HPI data register (HPID) but requests no automatic post-increment of the HPI address register.

Table 20-5. Cycle Types Selectable With the UHPI_HCNTL and UHPI_HR/$\overline{W}$ Signals

UHPI_HCNTL1	UHPI_HCNTL0	UHPI_HR/$\overline{W}$	Cycle Type
0	0	0	HPIC write cycle
0	0	1	HPIC read cycle
0	1	0	HPID write cycle with autoincrementing
0	1	1	HPID read cycle with autoincrementing
1	0	0	HPIA write cycle
1	0	1	HPIA read cycle
1	1	0	HPID write cycle without autoincrementing
1	1	1	HPID read cycle without autoincrementing

20.2.6.6 UHPI_HHWIL: Identifying the First and Second Halfwords in Multiplexed Mode Transfers

Each host cycle consists of two consecutive halfword transfers. For each transfer, the host must specify the cycle type with UHPI_HCNTL[1:0] and UHPI_HR/$\overline{W}$, and the host must use UHPI_HHWIL to indicate whether the first or second halfword is being transferred. For HPID and HPIA accesses, UHPI_HHWIL must always be driven low for the first halfword transfer and high for the second halfword transfer. Results are undefined if the sequence is broken. For examples of using UHPI_HHWIL, see Section 20.2.6.7.

When the host sends the two halfwords of a 32-bit word in this manner, the host can send the most-significant and the least-significant halfwords of the word in either order (most-significant halfword first or most-significant halfword second). However, the host must inform the HPI of the selected order before beginning the host cycle. This is done by programming the halfword order (HWOB) bit in HPIC. Although HWOB is written at bit 0 in HPIC, its current value is readable at both bit 0 and bit 8 (HWOBSTAT). Thus, the host can determine the current halfword order configuration by checking the least-significant bit of either half of HPIC.

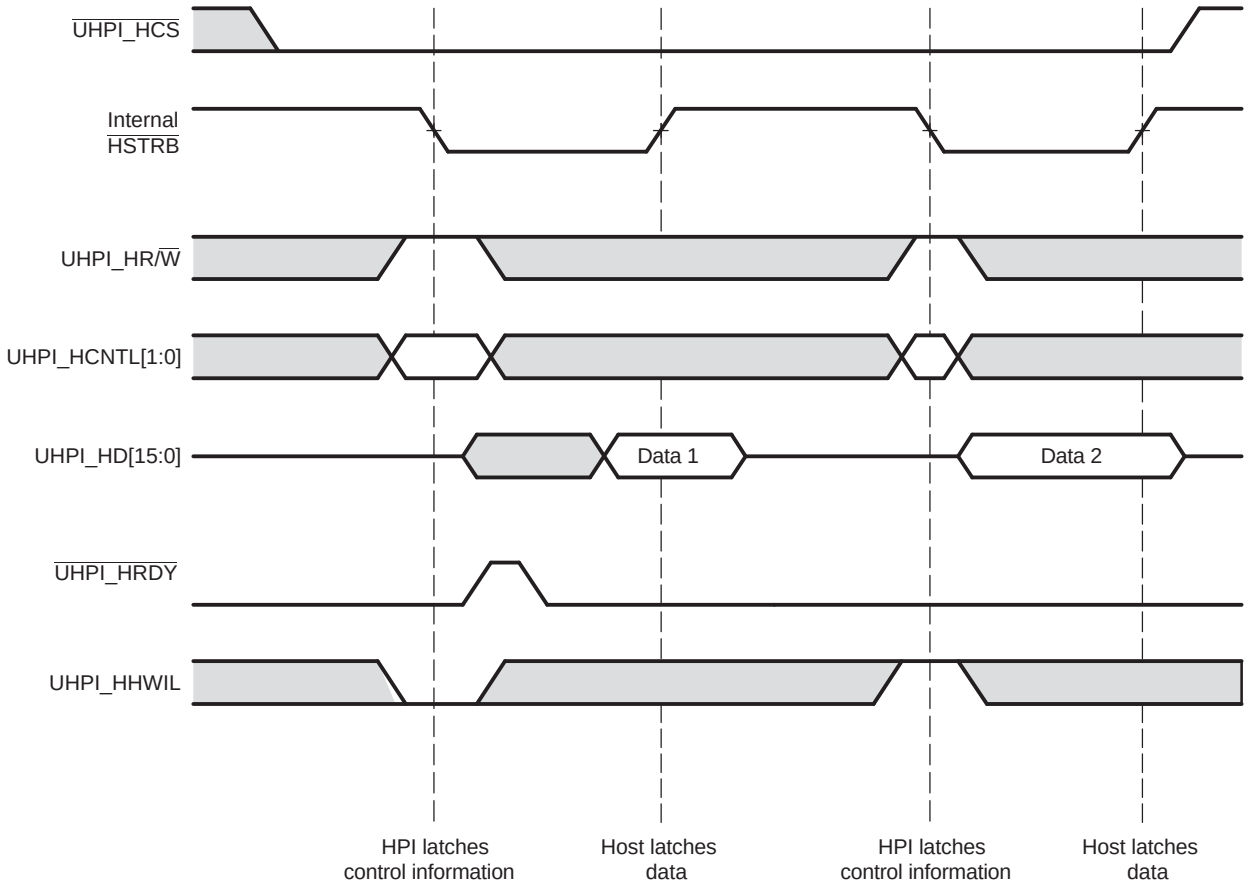
Copyright © 2013–2016, Texas Instruments Incorporated

There is one case when the HPI does not require a dual halfword access with UHPI_HHWIL low for the first halfword and UHPI_HHWIL high for the second halfword. This is the case when accessing the HPIC register. When accessing HPIC, the state of UHPI_HHWIL is ignored and the same 16-bit HPIC register is accessed regardless of whether the host performs a single or dual access. For an example timing diagram of this case, see Section 20.2.6.8.

20.2.6.7 Performing a Multiplexed Access

Figure 20-2 shows an example of signal connections for multiplexed transfers. Figure 20-4 and Figure 20-5 show typical HPI signal activity when performing a read and write transfer, respectively. In these cases, the falling edge of internal $\overline{\text{HSTRB}}$ is used to latch the UHPI_HCNTL[1:0], UHPI_HR/$\overline{\text{W}}$, and UHPI_HHWIL states into the HPI. Internal $\overline{\text{HSTRB}}$ is derived from $\overline{\text{UHPI_HCS}}$, $\overline{\text{UHPI_HDS1}}$, and $\overline{\text{UHPI_HDS2}}$ as described in Section 20.2.6.4.

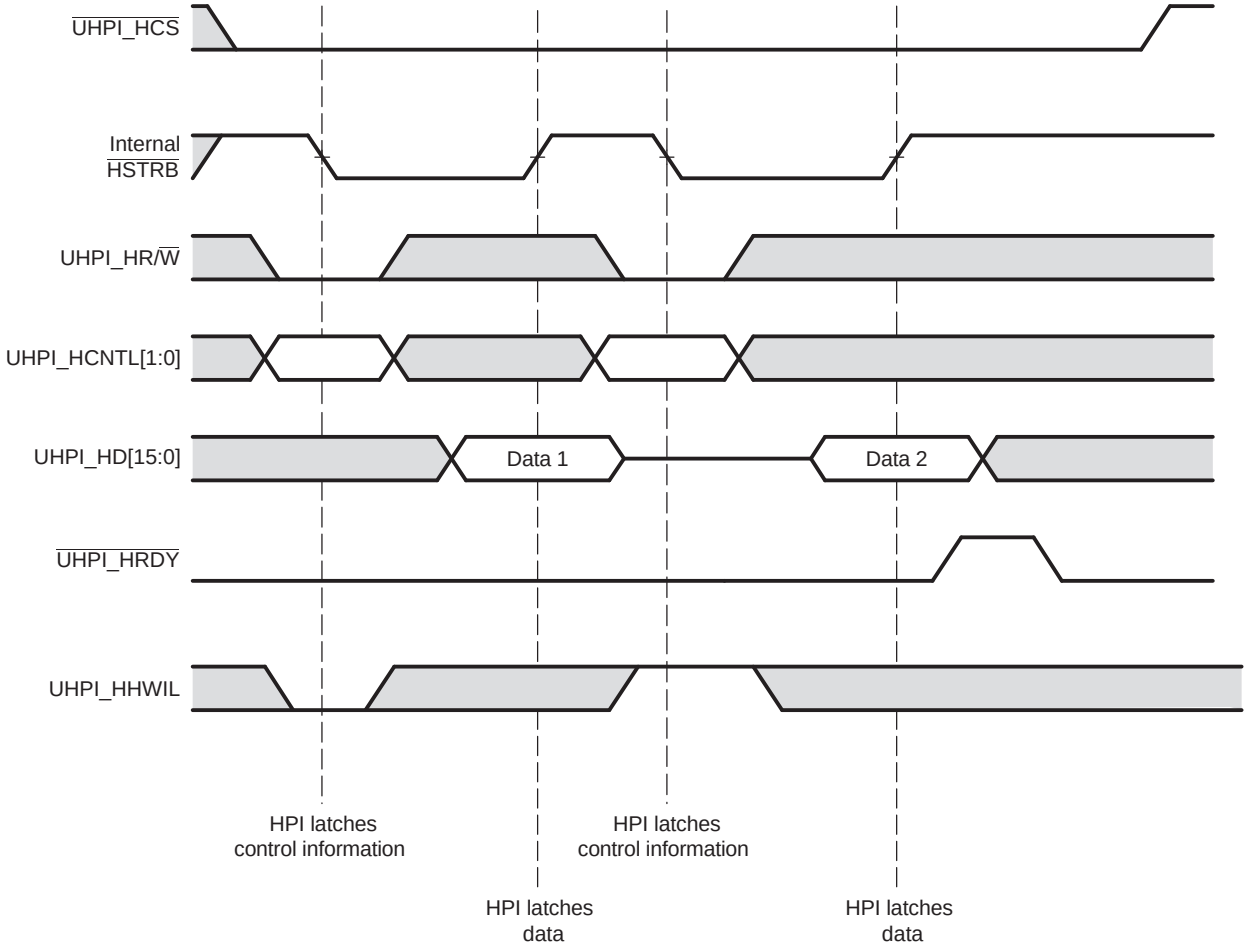
Figure 20-4. Multiplexed-Mode Host Read Cycle



NOTE: Depending on the type of write operation (HPID without autoincrementing, HPIA, HPIC, or HPID with autoincrementing) and the state of the FIFO, transitions on $\overline{\text{UHPI_HRDY}}$ may or may not occur. For more information, see Section 20.2.6.9.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 20-5. Multiplexed-Mode Host Write Cycle



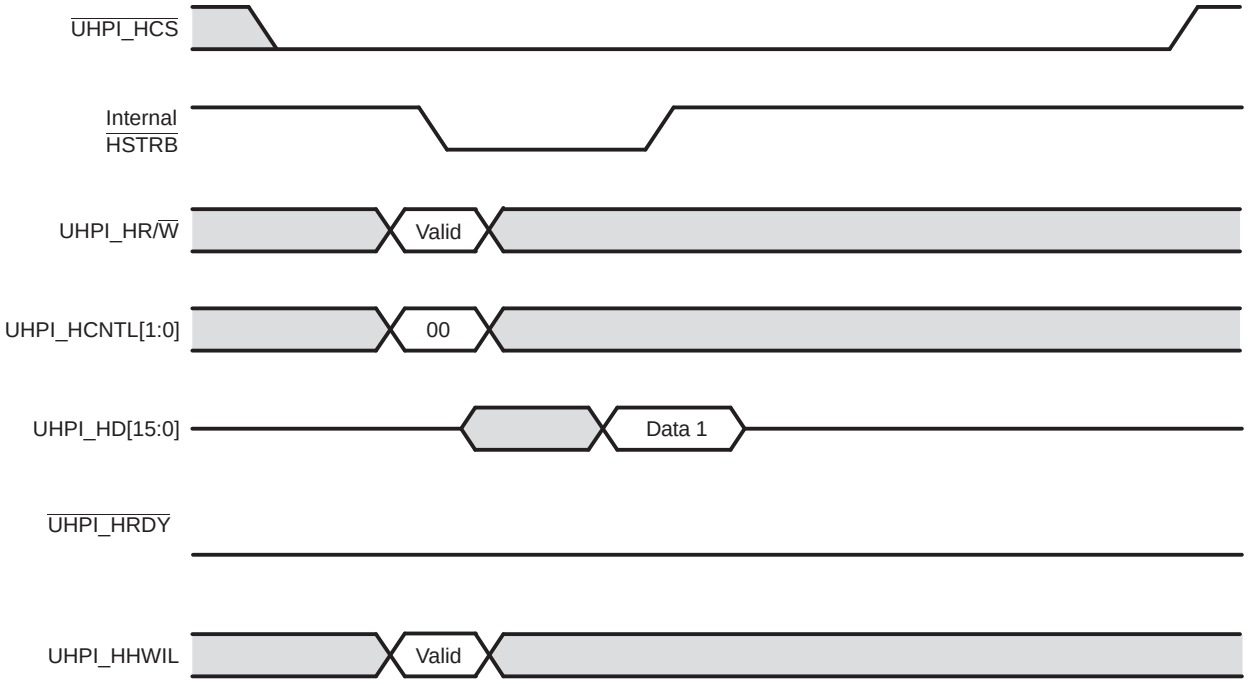
NOTE: Depending on the type of write operation (HPID without autoincrementing, HPIA, HPIC, or HPID with autoincrementing) and the state of the FIFO, transitions on UHPI_HRDY may or may not occur. For more information, see Section 20.2.6.9.

Copyright © 2013–2016, Texas Instruments Incorporated

20.2.6.8 Single-Halfword HPIC Cycle

Figure 20-6 shows the special case (see Section 20.2.6.6) when the host performs a single-halfword cycle to access the HPIC. The state of UHPI_HHWIL is ignored and if a dual-halfword access is performed, then the same HPIC register is accessed twice.

Figure 20-6. Multiplexed-Mode Single-Halfword HPIC Cycle (Read or Write)



20.2.6.9 Hardware Handshaking Using the HPI-Ready (UHPI_HRDY) Signal

The HPI uses its ready signal, UHPI_HRDY, to tell the host whether it is ready to complete an access. During a read cycle, the HPI is ready (UHPI_HRDY is low) when it has data available for the host. During a write cycle, the HPI is ready (UHPI_HRDY is low) when it is ready to latch data from the host. If the HPI is not ready, it can drive UHPI_HRDY high to insert wait states. These wait states indicate to the host that read data is not yet valid (read cycle) or that the HPI is not ready to latch write data (write cycle). The number of wait states that must be inserted by the HPI is dependent upon the state of the resource that is being accessed.

When the HPI is not ready to complete the current cycle (UHPI_HRDY is high), the host can begin a new host cycle by forcing the HPI to latch new control information. However, once the cycle has been initiated, the host must wait until UHPI_HRDY goes low before causing a rising edge on the internal strobe signal (internal HSTRB) to complete the cycle. If internal HSTRB goes high when the HPI is not ready, the cycle will be terminated with invalid data being returned (read cycle) or written (write cycle).

One reason the HPI may drive UHPI_HRDY high is a not-ready condition in one of its first-in, first-out buffers (FIFOs). For example, any HPID access that occurs while the write FIFO is full or the read FIFO is empty may result in some number of wait states being inserted by the HPI. The FIFOs are explained in Section 20.2.6.10.

The following sections describe the behavior of UHPI_HRDY during HPI register accesses. In all cases, the chip select signal, UHPI_HCS, must be asserted for UHPI_HRDY to go low.

Copyright © 2013–2016, Texas Instruments Incorporated

20.2.6.9.1 UHPI_HRDY Behavior During Multiplexed-Mode Read Operations

Figure 20-7 shows an HPIC (UHPI_HCNTL[1:0] = 00b) or HPIA (UHPI_HCNTL[1:0] = 10b) read cycle. Neither an HPIC read cycle nor an HPIA read cycle causes UHPI_HRDY to go high. For this type of access, the state of UHPI_HHWIL is ignored, so if a dual halfword access is performed, the same register will be accessed twice.

Figure 20-7. UHPI_HRDY Behavior During an HPIC or HPIA Read Cycle in the Multiplexed Mode

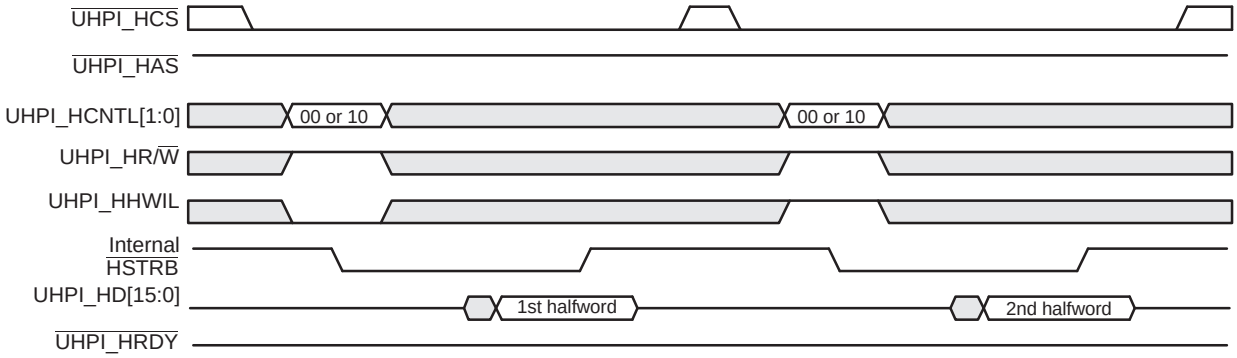


Figure 20-8 includes an HPID read cycle without autoincrementing. The host writes the memory address during the HPIA (UHPI_HCNTL[1:0] = 10b) write cycle, and the host reads the data during the HPID (UHPI_HCNTL[1:0] = 11b) read cycle. UHPI_HRDY goes high for each HPIA halfword access, but UHPI_HRDY goes high for only the first halfword access in each HPID read cycle.

Figure 20-8. UHPI_HRDY Behavior During a Data Read Operation in the Multiplexed Mode (Case 1: HPIA Write Cycle Followed by Nonautoincrement HPID Read Cycle)

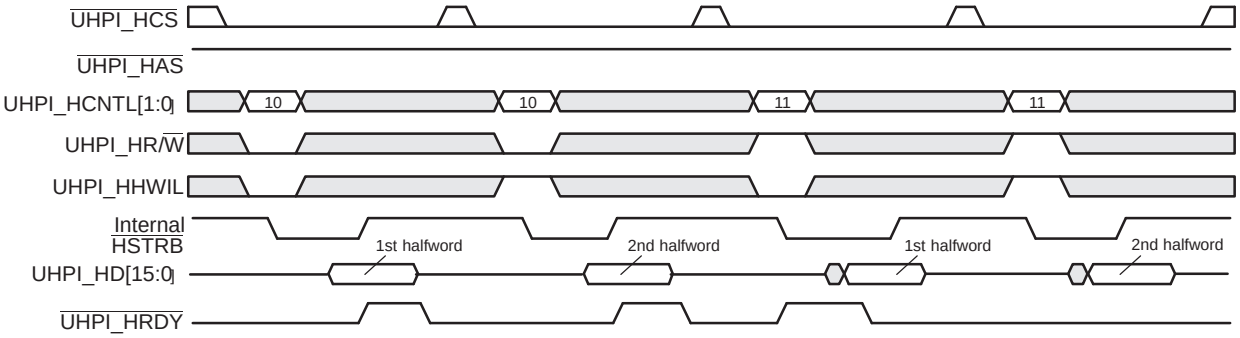
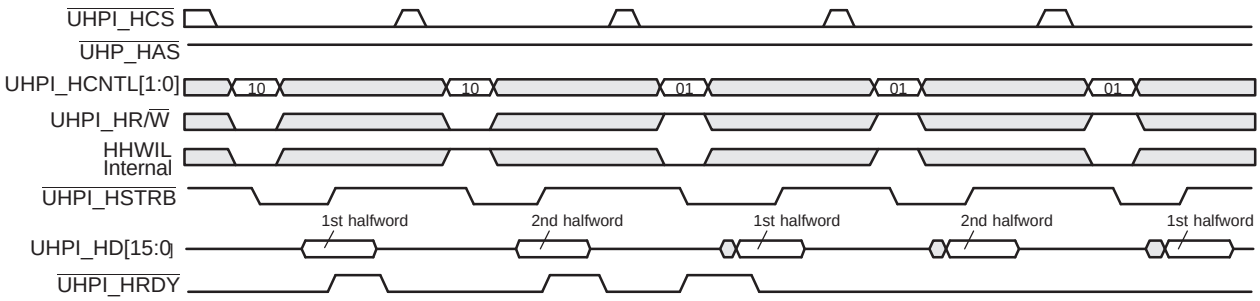


Figure 20-9 includes an autoincrement HPID read cycle. The host writes the memory address while asserting UHPI_HCNTL[1:0] = 10b and reads the data while asserting UHPI_HCNTL[1:0] = 01b. During the first HPID read cycle, UHPI_HRDY goes high for only the first halfword access, and subsequent HPID read cycles do not cause UHPI_HRDY to go high.

Figure 20-9. UHPI_HRDY Behavior During a Data Read Operation in the Multiplexed Mode (Case 2: HPIA Write Cycle Followed by Autoincrement HPID Read Cycles)



Copyright © 2013–2016, Texas Instruments Incorporated

20.2.6.9.2 UHPI_HRDY Behavior During Multiplexed-Mode Write Operations

Figure 20-10 shows an HPIC (UHPI_HCNTL[1:0] = 00b) write cycle operation. An HPIC write cycle does not cause UHPI_HRDY to go high and the state of UHPI_HHWIL is ignored. Firmware is not required to perform a dual access to access HPIC.

Figure 20-10. UHPI_HRDY Behavior During an HPIC Write Cycle in the Multiplexed Mode

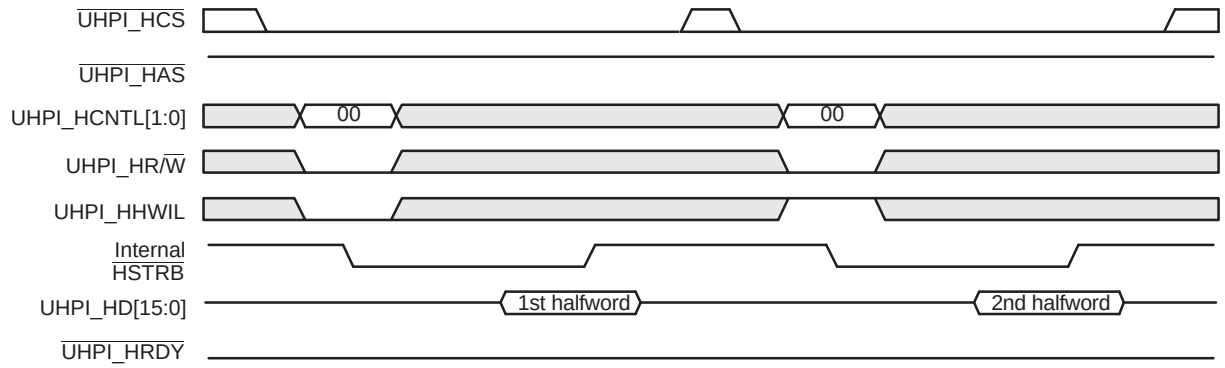


Figure 20-11 includes a HPID write cycle without autoincrementing. The host writes the memory address while UHPI_HCNTL[1:0] = 10b and writes the data while UHPI_HCNTL[1:0] = 11b. During the HPID write cycle, UHPI_HRDY goes high only for the second halfword access.

Figure 20-11. UHPI_HRDY Behavior During a Data Write Operation in the Multiplexed Mode (Case 1: No Autoincrementing)

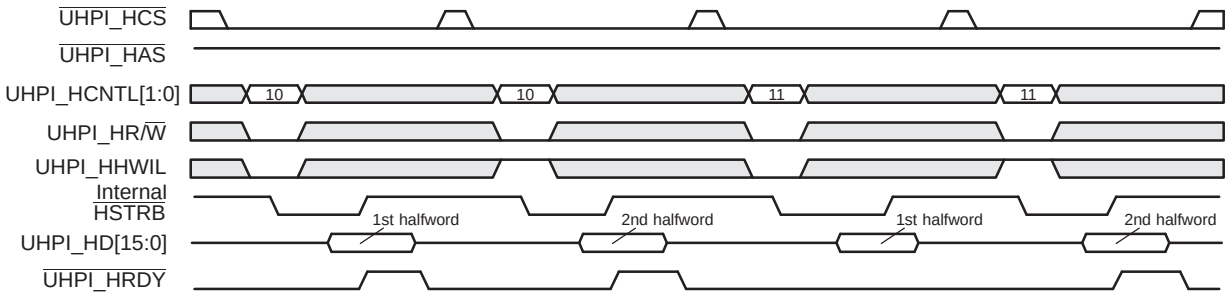
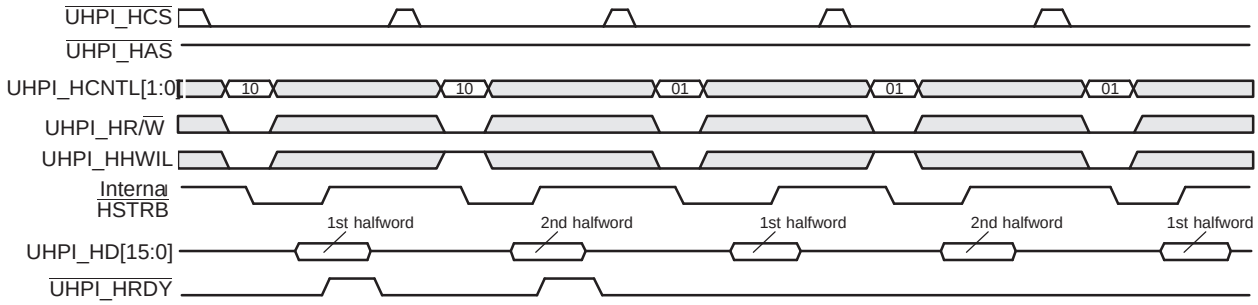


Figure 20-12 shows autoincrement HPID write cycles when the write FIFO is empty prior to the HPIA write. The host writes the memory address while UHPI_HCNTL[1:0] = 10b and writes the data while UHPI_HCNTL[1:0] = 01b. UHPI_HRDY does not go high during any of the HPID write cycles until the FIFO is full.

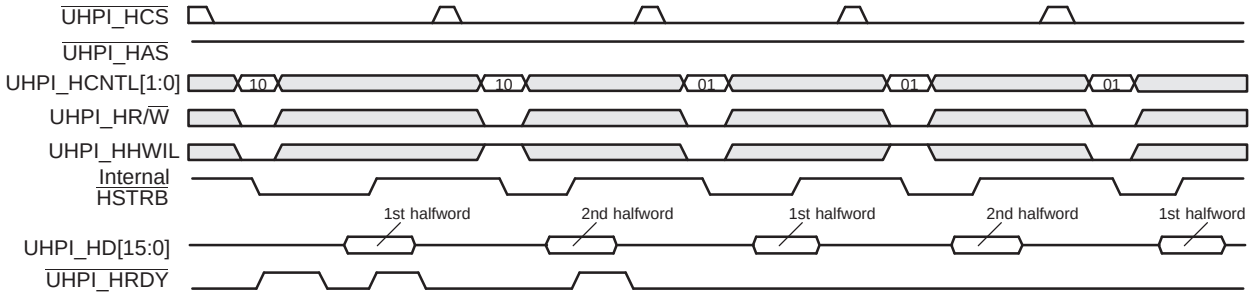
Figure 20-12. UHPI_HRDY Behavior During a Data Write Operation in the Multiplexed Mode (Case 2: Autoincrementing Selected, FIFO Empty Before Write)



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 20-13 shows a case similar to that of Figure 20-12. However, in the case of Figure 20-13, the write FIFO is not empty when the HPIA access is made. UHPI_HRDY goes high twice for the first halfword access of the HPIA write cycle. The first UHPI_HRDY high period is due to the nonempty FIFO. The data currently in the FIFO must first be written to the memory. This results in UHPI_HRDY going high immediately after the falling edge of the data strobe (HSTRB). The second and third UHPI_HRDY high periods occur for the writes to the HPIA. UHPI_HRDY remains low for the HPID accesses.

Figure 20-13. UHPI_HRDY Behavior During a Data Write Operation in the Multiplexed Mode (Case 3: Autoincrementing Selected, FIFO Not Empty Before Write)



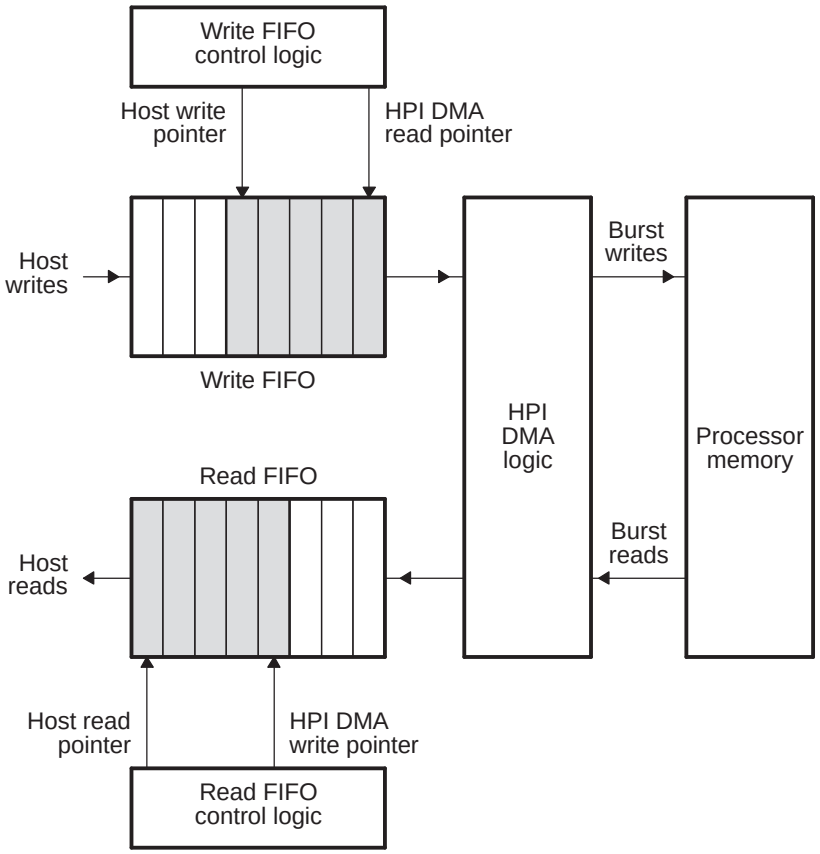
Copyright © 2013–2016, Texas Instruments Incorporated

20.2.6.10 FIFOs and Bursting

The HPI data register (HPID) is a port through which the host accesses two first-in, first-out buffers (FIFOs). As shown in Figure 20-14, a read FIFO supports host read cycles, and a write FIFO supports host write cycles. Both read and write FIFOs are 8-words deep (each word is 32 bits). If the host is performing multiple reads or writes to consecutive memory addresses (autoincrement HPID cycles), the FIFOs are used for bursting. The HPI DMA logic reads or writes a burst of four words at a time when accessing one of the FIFOs.

Bursting is essentially invisible to the host because the host interface signaling is not affected. Its benefit to the host is that the $\overline{\text{UHPI_HRDY}}$ signal is deasserted less often when there are multiple reads or writes to consecutive addresses.

Figure 20-14. FIFOs in the HPI



Copyright © 2013–2016, Texas Instruments Incorporated

20.2.6.10.1 Read Bursting

When the host writes to the read address register (HPIAR), the read FIFO is flushed. Any host read data that was in the read FIFO is discarded (the read FIFO pointers are reset). If an HPI DMA write to the read FIFO is in progress at the time of a flush request, the HPI allows this write to complete and then performs the flush.

Read bursting can begin in one of two ways: the host initiates an HPID read cycle with autoincrementing, or the host initiates issues a FETCH command (writes 1 to the FETCH bit in HPIC).

If the host initiates an HPID read cycle with autoincrementing, the HPI DMA logic performs two 4-word burst operations to fill the read FIFO. The host is initially held off by the deassertion of the $\overline{\text{UHPI_HRDY}}$ signal until data is available to be read from the read FIFO. Once data is available in the read FIFO, the host can read data from the read FIFO by performing subsequent reads of HPID with autoincrementing. Once the initial read has been performed, the HPI DMA logic continues to perform 4-word burst operations to consecutive memory addresses every time there are four empty word locations in the read FIFO. The HPI DMA logic continues to prefetch data to keep the read FIFO full, until the occurrence of an event that causes a read FIFO flush (see Section 20.2.6.10.3).

As mentioned, the second way that read bursting may begin is with a FETCH command. The host should always precede the FETCH command with the initialization of the HPIAR register or a nonautoincrement access, so that the read FIFO is flushed beforehand. When the host initiates a FETCH command, the HPI DMA logic begins to prefetch data to keep the read FIFO full, as described in the previous paragraph. The FETCH bit in HPIC does not actually store the value that is written to it; rather, the decoding of a host write of 1 to this bit is considered a FETCH command.

The FETCH command can be helpful if the host wants to minimize a stall condition on the interface. The host can initiate prefetching by writing 1 to the FETCH bit and later perform a read. The host can make use of the time it takes to load the read FIFO with read data, during which the HPI was not ready, by using the CPU to service other tasks.

Both types of continuous or burst reads described in the previous paragraphs begin with a write to the HPI address register, which causes a read FIFO flush. This is the typical way of initiating read cycles, because the initial read address needs to be specified.

NOTE: An HPID read cycle without autoincrementing does not initiate any prefetching activity. Instead, it causes the read FIFO to be flushed and causes the HPI DMA logic to perform a single-word read from the processor memory. As soon as the host activates a read cycle without autoincrementing, prefetching activity ceases until the occurrence of a FETCH command or an autoincrement read cycle.

Copyright © 2013–2016, Texas Instruments Incorporated

20.2.6.10.2 Write Bursting

A write to the write address register (HPIAW) causes the write FIFO to be flushed. This means that any write data in the write FIFO is forced to its destination in the processor memory (the HPI DMA logic performs burst operations until the write FIFO is empty). When the FIFO has been flushed, the only action that will cause the HPI DMA logic to perform burst writes is a host write to HPID with autoincrementing. The initial host-write data is stored in the write FIFO. An HPI DMA write is not requested until there are four words in the write FIFO. As soon as four words have been written to the FIFO via HPID write cycles with autoincrementing, the HPI DMA logic performs a 4-word burst operation to the processor memory. The burst operations continue as long as there are at least four words in the FIFO. If the FIFO becomes full (eight words are waiting in the FIFO), the HPI holds off the host by deasserting $\overline{\text{UHPI_HRDY}}$ until at least one empty word location is available in the FIFO.

Because excessive time might pass between consecutive burst operations, the HPI has a time-out counter. If there are fewer than four words in the write FIFO and the time-out counter expires, the HPI DMA logic empties the FIFO immediately by performing a 2-word or 3-word burst, or a single-word write, as necessary. Every time new data is written to the write FIFO, the time-out counter is automatically reset to begin its count again. The time-out period is set to a value of 160. For more detailed information about the time-out period, see your device-specific data manual.

NOTE: An HPID write cycle without autoincrementing does not initiate any bursting activity. Instead, it causes the write FIFO to be flushed and causes the HPI DMA logic to perform a single-word write to the processor memory. As soon as the host activates a write cycle without autoincrementing, bursting activity ceases until the occurrence of an autoincrement write cycle. A nonautoincrement write cycle always should be preceded by the initialization of HPIAW or by another nonautoincrement access, so that the write FIFO is flushed beforehand.

Copyright © 2013–2016, Texas Instruments Incorporated

20.2.6.10.3 FIFO Flush Conditions

When specific conditions occur within the HPI, the read or write FIFO must be flushed to prevent the reading of stale data from the FIFOs. When a read FIFO flush condition occurs, all current host accesses and direct memory accesses (DMAs) to the read FIFO are allowed to complete. This includes DMAs that have been requested but not yet initiated. The read FIFO pointers are then reset, causing any read data to be discarded.

Similarly, when a write FIFO flush condition occurs, all current host accesses and DMAs to the write FIFO are allowed to complete. This includes DMAs that have been requested but not yet initiated. All posted writes in the FIFO are then forced to completion with a final burst or single-word write, as necessary.

If the host initiates an HPID host cycle during a FIFO flush, the cycle is held off with the deassertion of $\overline{\text{UHPI_HRDY}}$ until the flush is complete and the FIFO is ready to be accessed.

The following conditions cause the read and write FIFOs to be flushed:

- Read FIFO flush conditions:
  - A value from the host is written to the read address register (HPIAR).
  - The host performs an HPID read cycle without autoincrementing.
- Write FIFO flush conditions:
  - A value from the host is written to the write address register (HPIAW).
  - The host performs an HPID write cycle without autoincrementing.
  - The write-burst time-out counter expires.

When operating with DUALHPIA = 0, any read or write flush condition causes both read and write FIFOs to be flushed. In addition, the following scenarios cause both FIFOs to be flushed when DUALHPIA = 0:

- The host performs a write to the HPIA register.
- The host performs an HPID write cycle with autoincrementing while the read FIFO is not empty (the read FIFO still contains data from prefetching or an HPID read cycle with autoincrementing).
- The host performs an HPID read cycle with autoincrementing while the write FIFO is not empty (there is still posted write data in the write FIFO).

This is useful in providing protection against reading stale data by reading a memory address when a previous write cycle has not been completed at the same address. Similarly, this protects against overwriting data at a memory address when a previous read cycle has not been completed at the same address.

When operating with DUALHPIA = 1 (HPIAR and HPIAW are independent), there is no such protection. However, when DUALHPIA = 1, data flow can occur in both directions without flushing both FIFOs simultaneously, thereby improving HPI bandwidth.

20.2.6.10.4 FIFO Behavior When a Hardware Reset or Software Reset Occurs

A hardware reset (RESET pin driven low) or an HPI software reset causes the FIFOs to be reset. The FIFO pointers are cleared, so that all data in the FIFOs are discarded. In addition, all associated FIFO logic is reset.

If a host cycle is active when a hardware or HPI software reset occurs, the $\overline{\text{UHPI_HRDY}}$ signal is asserted (driven low), allowing the host to complete the cycle. When the cycle is complete, $\overline{\text{UHPI_HRDY}}$ is deasserted (driven high). Any access interrupted by a reset may result in corrupted read data or a lost write data (if the write does not actually update the intended memory or register). Although data may be lost, the host interface protocol is not violated. While either of reset condition is true, and the host is idle (internal $\overline{\text{HSTRB}}$ is held high), the FIFOs are held in reset, and host transactions are held off with an inactive $\overline{\text{UHPI_HRDY}}$ signal.

Copyright © 2013–2016, Texas Instruments Incorporated

20.2.7 Reset Considerations

The HPI has two reset sources: software reset and hardware reset.

20.2.7.1 Software Reset Considerations

The HPI is not affected by a software reset issued by the emulator.

20.2.7.2 Hardware Reset Considerations

When the entire processor is reset with the RESET pin:

- If the internal strobe signal, internal $\overline{\text{HSTRB}}$, is high (host is inactive), $\overline{\text{UHPI_HRDY}}$ is driven low and remains low until the reset condition is over.
- If internal $\overline{\text{HSTRB}}$ is low (host cycle is active), $\overline{\text{UHPI_HRDY}}$ is driven high, allowing the host to complete the cycle. When internal $\overline{\text{HSTRB}}$ goes high (cycle is complete), $\overline{\text{UHPI_HRDY}}$ is driven low and remains low until the reset condition is over. If the active cycle was a write cycle, the memory or register may not have been correctly updated. If the active cycle was a read cycle, the fetched value may not be valid.
- The HPI registers are reset to their default values (see Section 20.3).
- The read and write FIFOs and the associated FIFO logic are reset (this includes a flush of the FIFOs).
- Host-to-CPU and CPU-to-host interrupts are cleared.

20.2.8 Initialization

The following steps are required to configure the HPI after a hardware reset:

1. Perform the necessary device pin multiplexing setup (see your device-specific data manual).
2. Configure the HPIENA and HPIBYTEAD bits in the chip configuration 1 register (CFGCHIP1) in the *System Configuration (SYSCFG) Module* chapter.
3. Choose how HPIAR and HPIAW will be controlled by configuring the DUALHPIA bit in HPIC.
4. Choose how halfword ordering will be handled by configuring the HWOB bit in HPIC.
5. Choose how the HPI will respond to emulation suspend events by configuring the FREE and SOFT bits in PWREMU_MGMT.
6. Choose the desired initial addresses and write the addresses to HPIAW and HPIAR, appropriately.
7. Release the HPI logic from reset by clearing the HPIRST bit in HPIC.

The HPI is now ready to perform data transactions.

Copyright © 2013–2016, Texas Instruments Incorporated

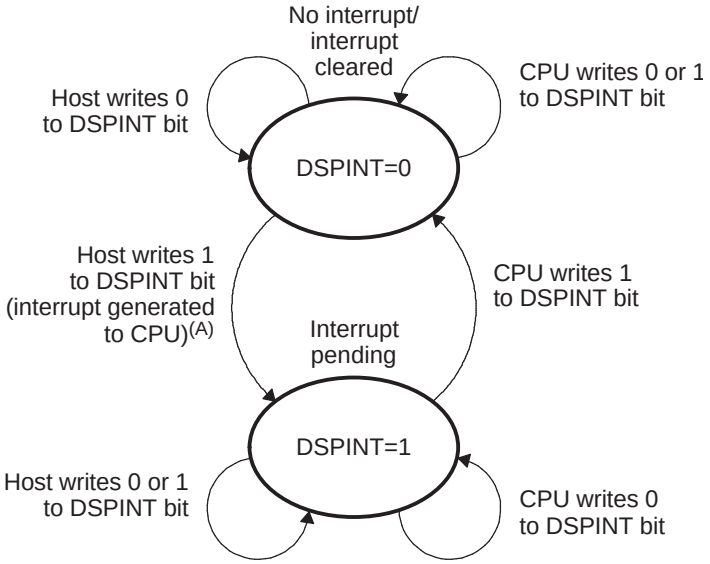
20.2.9 Interrupt Support

The host can interrupt the CPU via the DSPINT bit in HPIC, as described in Section 20.2.9.1. The CPU can send an interrupt to the host by using the HINT bit in HPIC, as described in Section 20.2.9.2.

20.2.9.1 DSPINT Bit: Host-to-CPU Interrupts

The DSPINT bit in HPIC allows the host to send an interrupt request to the CPU. The use of the DSPINT bit is summarized in Figure 20-15.

Figure 20-15. Host-to-CPU Interrupt State Diagram



A When the DSPINT bit transitions from 0 to 1, an interrupt is generated to the CPU. No new interrupt can be generated until the CPU has cleared the bit (DSPINT = 0).

To interrupt the CPU, the host must:

1. Drive both UHPI_HCNTL1 and UHPI_HCNTL0 low to request a write to HPIC.
2. Write 1 to the DSPINT bit in HPIC.

When the host sets the DSPINT bit, the HPI generates an interrupt pulse to the CPU. If this maskable interrupt is properly enabled in the CPU, the CPU executes the corresponding interrupt service routine (ISR).

Before the host can use DSPINT to generate a subsequent interrupt to the CPU, the CPU must acknowledge the current interrupt by writing a 1 to the DSPINT bit. When the CPU writes 1, DSPINT is forced to 0. The host should verify that DSPINT = 0 before generating subsequent interrupts. While DSPINT = 1, host writes to the DSPINT bit do not generate an interrupt pulse.

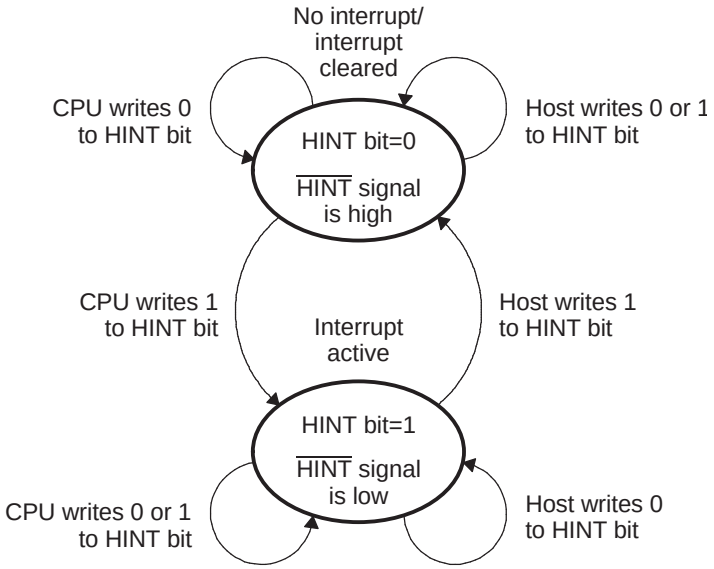
Writes of 0 have no effect. A hardware reset immediately clears DSPINT and thus clears an active host-to-CPU interrupt.

Copyright © 2013–2016, Texas Instruments Incorporated

20.2.9.2 HINT Bit: CPU-to-Host Interrupts

The HINT bit in HPIC allows the CPU to send an interrupt request to the host. The use of the HINT bit is summarized in Figure 20-16.

Figure 20-16. CPU-to-Host Interrupt State Diagram



If the CPU writes 1 to the HINT bit of HPIC, the HPI drives the $\overline{\text{UHPI_HINT}}$ signal low, indicating an interrupt condition to the host. Before the CPU can use the HINT bit generate a subsequent interrupt to host, the host must acknowledge the current interrupt by writing 1 to the HINT bit. When the host does this, the HPI clears the HINT bit (HINT = 0), and this drives the $\overline{\text{UHPI_HINT}}$ signal high. The CPU should read HPIC and make sure HINT = 0 before generating subsequent interrupts.

Writes of 0 have no effect. A hardware reset immediately clears the HINT bit and thus clears an active CPU-to-host interrupt.

20.2.10 EDMA Event Support

The HPI does not provide synchronization events to the EDMA system. Memory accesses from the HPI are handled automatically, independent of the EDMA controller. The HPI controller has its own dedicated DMA and its operation and configuration are transparent.

20.2.11 Power Management

The HPI peripheral can be placed in reduced-power modes to conserve power during periods of low activity. The power management of the HPI peripheral is controlled by the processor Power and Sleep Controller (PSC). The PSC acts as a master controller for power management for all of the peripherals on the device. For detailed information on power management procedures using the PSC, see the *Power and Sleep Controller (PSC)* chapter.

Copyright © 2013–2016, Texas Instruments Incorporated

20.2.12 Emulation Considerations

The FREE and SOFT bits in the power and emulation management register (PWREMU_MGMT) determine the response of the HPI to an emulation suspend condition. If FREE = 1, the HPI is not affected, and the SOFT bit has no effect. If FREE = 0 and SOFT = 0, the HPI is not affected. If FREE = 0 and SOFT = 1:

- The HPI DMA logic halts after the current host and HPI DMA operations are completed.
- The external host interface functions as normal throughout the emulation suspend condition. The host may access the control register (HPIC). The host may also access the HPIA registers and may perform data reads until the read FIFO is empty or data writes until the write FIFO is full. As in normal operation, $\overline{\text{UHPI_HRDY}}$ is driven low during a host cycle that cannot be completed due to the write FIFO being full or the read FIFO being empty. If this occurs, $\overline{\text{UHPI_HRDY}}$ continues to be driven low, holding off the host, until the emulation suspend condition is over, and the FIFOs are serviced by the HPI DMA logic, allowing the host cycle to complete.
- When the emulation suspend condition is over, the appropriate requests by the HPI DMA logic are made to process any posted host writes in the write FIFO or to fill the read FIFO as necessary. HPI operation then continues as normal.

20.3 Registers

Table 20-6 lists the memory-mapped registers for the HPI. See your device-specific data manual for the memory addresses of these registers.

Table 20-6. HPI Registers

Offset	Acronym	Register Description	Section
0	REVID	Revision Identification Register	Section 20.3.1
4h	PWREMU_MGMT	Power and Emulation Management Register	Section 20.3.2
Ch	GPIO_EN	GPIO Enable Register	Section 20.3.3
10h	GPIO_DIR1	GPIO Direction 1 Register	Section 20.3.4
14h	GPIO_DAT1	GPIO Data 1 Register	Section 20.3.5
18h	GPIO_DIR2	GPIO Direction 2 Register	Section 20.3.6
1Ch	GPIO_DAT2	GPIO Data 2 Register	Section 20.3.7
30h	HPIC	Host Port Interface Control Register	Section 20.3.8
34h	HPIAW	Host Port Interface Write Address Register	Section 20.3.9
38h	HPIAR	Host Port Interface Read Address Register	Section 20.3.10

Copyright © 2013–2016, Texas Instruments Incorporated

20.3.1 Revision Identification Register (REVID)

The revision identification register (REVID) contains identification data for the peripheral. REVID is shown in Figure 20-17 and described in Table 20-7.

Figure 20-17. Revision Identification Register (REVID)

31	0
REV	
R-4421 210Ah	

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

Table 20-7. Revision Identification Register (REVID) Field Descriptions

Bit	Field	Value	Description
31-0	REV	0-4421 210Ah	Revision identification

20.3.2 Power and Emulation Management Register (PWREMU_MGMT)

The power and emulation management register (PWREMU_MGMT) determines the emulation mode of the HPI. PWREMU_MGMT is shown in Figure 20-18 and described in Table 20-8.

Figure 20-18. Power and Emulation Management Register (PWREMU_MGMT)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		SOFT	FREE
R-0		R/W-0	R/W-0

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

Table 20-8. Power and Emulation Management Register (PWREMU_MGMT) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	SOFT		Determines emulation mode functionality of the HPI. When the FREE bit is cleared to 0, the SOFT bit selects the HPI mode.
		0	Upon emulation suspend, the HPI operation is not affected.
		1	In response to an emulation suspend event, the HPI logic halts after the current HPI transaction is completed.
0	FREE		Free run emulation control. Determines emulation mode functionality of the HPI. When the FREE bit is cleared to 0, the SOFT bit selects the HPI mode.
		0	The SOFT bit selects the HPI mode.
		1	The HPI runs free regardless of the SOFT bit.

Copyright © 2013–2016, Texas Instruments Incorporated

20.3.3 GPIO Enable Register (GPIO_EN)

The GPIO enable register (GPIO_EN) enables the pin for general-purpose I/O. GPIO_EN is shown in Figure 20-19 and described in Table 20-9.

Figure 20-19. GPIO Enable Register (GPIO_EN)

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							GPIOEN8
R-0							R/W-0

7	6	5	4	3	2	1	0
GPIOEN7	GPIOEN6	GPIOEN5	GPIOEN4	Reserved	GPIOEN2	GPIOEN1	GPIOEN0
R/W-0	R/W-0	R/W-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 20-9. GPIO Enable Register (GPIO_EN) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	GPIOEN8		Enable as GPIO for UHPI_HD[15:8] pins
		0	Disable pins for GPIO. Pins functions as HPI signal.
		1	Enable pins for GPIO.
7	GPIOEN7		Enable as GPIO for UHPI_HD[7:0] pins.
		0	Disable pins for GPIO. Pins functions as HPI signal.
		1	Enable pins for GPIO.
6	GPIOEN6		Enable as GPIO for $\overline{\text{UHPI_HINT}}$ pin.
		0	Disable pin for GPIO. Pin functions as HPI signal.
		1	Enable pin for GPIO.
5	GPIOEN5		Enable as GPIO for UHPI_HRDY pin.
		0	Disable pin for GPIO. Pin functions as HPI signal.
		1	Enable pin for GPIO.
4	GPIOEN4		Enable as GPIO for UHPI_HHWIL pin.
		0	Disable pin for GPIO. Pin functions as HPI signal.
		1	Enable pin for GPIO.
3	Reserved	0	Reserved
2	GPIOEN2		Enable as GPIO for $\overline{\text{UHPI_HAS}}$ pin.
		0	Disable pin for GPIO. Pin functions as HPI signal.
		1	Enable pin for GPIO.
1	GPIOEN1		Enable as GPIO for UHPI_HCNTL[1:0] pins.
		0	Disable pins for GPIO. Pins functions as HPI signal.
		1	Enable pins for GPIO.
0	GPIOEN0		Enable as GPIO for $\overline{\text{UHPI_HCS}}$, $\overline{\text{UHPI_HDS1}}$, $\overline{\text{UHPI_HDS2}}$, UHPI_HR/$\overline{\text{W}}$ pins.
		0	Disable pins for GPIO. Pins functions as HPI signal.
		1	Enable pins for GPIO.

Copyright © 2013–2016, Texas Instruments Incorporated

20.3.4 GPIO Direction 1 Register (GPIO_DIR1)

The GPIO direction 1 register (GPIO_DIR1) determines if the UHPI_HD*n* pin is an input or an output. GPIO_DIR1 is shown in Figure 20-20 and described in Table 20-10.

Figure 20-20. GPIO Direction 1 Register (GPIO_DIR1)

31															16
Reserved															
R-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HD15	HD14	HD13	HD12	HD11	HD10	HD9	HD8	HD7	HD6	HD5	HD4	HD3	HD2	HD1	HD0
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	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 20-10. GPIO Direction 1 Register (GPIO_DIR1) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	HD*n*		Direction control for UHPI_HD*n* pin.
		0	UHPI_HD*n* pin is an input.
		1	UHPI_HD*n* pin is an output.

20.3.5 GPIO Data 1 Register (GPIO_DAT1)

The GPIO data 1 register (GPIO_DAT1) determines the value driven on the corresponding UHPI_HD*n* pin, if the pin is configured as an output (GPIO_DIR1.HD*n* = 1). Writes do not affect pins not configured as GPIO outputs. The bits in GPIO_DAT1 are set or cleared by writing directly to this register. A read of GPIO_DAT1 returns the value of the register bit (HD*n*) not the value at the UHPI_HD*n* pin (that might be configured as an input). GPIO_DAT1 is shown in Figure 20-21 and described in Table 20-11.

Figure 20-21. GPIO Data 1 Register (GPIO_DAT1)

31															16
Reserved															
R-0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HD15	HD14	HD13	HD12	HD11	HD10	HD9	HD8	HD7	HD6	HD5	HD4	HD3	HD2	HD1	HD0
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	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 20-11. GPIO Data 1 Register (GPIO_DAT1) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	HD*n*	0-1	Data read from/written to UHPI_HD*n* pin.

Copyright © 2013–2016, Texas Instruments Incorporated

20.3.6 GPIO Direction 2 Register (GPIO_DIR2)

The GPIO direction 2 register (GPIO_DIR2) determines if the HPI pin is an input or an output. GPIO_DIR2 is shown in Figure 20-22 and described in Table 20-12.

Figure 20-22. GPIO Direction 2 Register (GPIO_DIR2)

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						HRDY	HINTZ
R-0						R/W-0	R/W-0

7	6	5	4	3	2	1	0
HCNTL0	HCNTL1	HHWIL	HRW	HDS2Z	HDS1Z	HCSZ	HASZ
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 20-12. GPIO Direction 2 Register (GPIO_DIR2) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9	HRDY		Direction control for UHPI_HRDY pin.
		0	UHPI_HRDY pin is an input.
		1	UHPI_HRDY pin is an output.
8	HINTZ		Direction control for $\overline{\text{UHPI_HINT}}$ pin.
		0	$\overline{\text{UHPI_HINT}}$ pin is an input.
		1	$\overline{\text{UHPI_HINT}}$ pin is an output.
7	HCNTL0		Direction control for UHPI_HCNTL0 pin.
		0	UHPI_HCNTL0 pin is an input.
		1	UHPI_HCNTL0 pin is an output.
6	HCNTL1		Direction control for UHPI_HCNTL1 pin.
		0	UHPI_HCNTL1 pin is an input.
		1	UHPI_HCNTL1 pin is an output.
5	HHWIL		Direction control for UHPI_HHWIL pin.
		0	UHPI_HHWIL pin is an input.
		1	UHPI_HHWIL pin is an output.
4	HRW		Direction control for UHPI_HR/$\overline{\text{W}}$ pin.
		0	UHPI_HR/$\overline{\text{W}}$ pin is an input.
		1	UHPI_HR/$\overline{\text{W}}$ pin is an output.
3	HDS2Z		Direction control for $\overline{\text{UHPI_HDS2}}$ pin.
		0	$\overline{\text{UHPI_HDS2}}$ pin is an input.
		1	$\overline{\text{UHPI_HDS2}}$ pin is an output.
2	HDS1Z		Direction control for $\overline{\text{UHPI_HDS1}}$ pin.
		0	$\overline{\text{UHPI_HDS1}}$ pin is an input.
		1	$\overline{\text{UHPI_HDS1}}$ pin is an output.
1	HCSZ		Direction control for $\overline{\text{UHPI_HCS}}$ pin.
		0	$\overline{\text{UHPI_HCS}}$ pin is an input.
		1	$\overline{\text{UHPI_HCS}}$ pin is an output.
0	HASZ		Direction control for $\overline{\text{UHPI_HAS}}$ pin.
		0	$\overline{\text{UHPI_HAS}}$ pin is an input.
		1	$\overline{\text{UHPI_HAS}}$ pin is an output.

Copyright © 2013–2016, Texas Instruments Incorporated

20.3.7 GPIO Data 2 Register (GPIO_DAT2)

The GPIO data 2 register (GPIO_DAT2) determines the value driven on the corresponding HPI pin, if the pin is configured as an output (GPIO_DIR2 bit = 1). Writes do not affect pins not configured as GPIO outputs. The bits in GPIO_DAT2 are set or cleared by writing directly to this register. A read of GPIO_DAT2 returns the value of the register bit not the value at the HPI pin (that might be configured as an input). GPIO_DAT2 is shown in Figure 20-23 and described in Table 20-13.

Figure 20-23. GPIO Data 2 Register (GPIO_DAT2)

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						HRDY	HINTZ
R-0						R/W-0	R/W-0

7	6	5	4	3	2	1	0
HCNTL0	HCNTL1	HHWIL	HRW	HDS2Z	HDS1Z	HCSZ	HASZ
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 20-13. GPIO Data 2 Register (GPIO_DAT2) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9	HRDY	0-1	Data read from/written to UHPI_HRDY pin.
8	HINTZ	0-1	Data read from/written to $\overline{\text{UHPI_HINT}}$ pin.
7	HCNTL0	0-1	Data read from/written to UHPI_HCNTL0 pin.
6	HCNTL1	0-1	Data read from/written to UHPI_HCNTL1 pin.
5	HHWIL	0-1	Data read from/written to UHPI_HHWIL pin.
4	HRW	0-1	Data read from/written to UHPI_HR/$\overline{\text{W}}$ pin.
3	HDS2Z	0-1	Data read from/written to $\overline{\text{UHPI_HDS2}}$ pin.
2	HDS1Z	0-1	Data read from/written to $\overline{\text{UHPI_HDS1}}$ pin.
1	HCSZ	0-1	Data read from/written to $\overline{\text{UHPI_HCS}}$ pin.
0	HASZ	0-1	Data read from/written to $\overline{\text{UHPI_HAS}}$ pin.

Copyright © 2013–2016, Texas Instruments Incorporated

20.3.8 Host Port Interface Control Register (HPIC)

The host port interface control register (HPIC) stores configuration and control information for the HPI. As shown in Figure 20-24 and Figure 20-25 and described in Table 20-14, the host and CPU do not have the same access permissions. The host has full read/write access; the CPU has primarily read-only access, but with the exception that the CPU can write 1 to the HINT bit to generate an interrupt to the host.

Figure 20-24. Host Port Interface Control Register (HPIC)–Host Access Permissions

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				HPIASEL	Reserved	DUALHPIA	HWOBSTAT
R-0				R/W-0	R/W-0	R/W-0	R-0

7	6	5	4	3	2	1	0
HPIRST	Reserved		FETCH	Reserved	HINT	DSPINT	HWOB
R-1	R-2h		R/W-0	R-1	R/W-1 (Host) R/W1C-0 (CPU)	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing 0 has no effect); -*n* = value after reset

Figure 20-25. Host Port Interface Control Register (HPIC)–CPU Access Permissions

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				HPIASEL	Reserved	DUALHPIA	HWOBSTAT
R-0				R-0	R-0	R-0	R-0

7	6	5	4	3	2	1	0
HPIRST	Reserved		FETCH	Reserved	HINT	DSPINT	HWOB
R/W-1	R-2h		R-0	R-1	R/W-1 (Host) R/W1C-0 (CPU)	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing 0 has no effect); -*n* = value after reset

Copyright © 2013–2016, Texas Instruments Incorporated

Table 20-14. Host Port Interface Control Register (HPIC) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reserved
11	HPIASEL		HPI address register select bit. When DUALHPIA = 1, the HPIASEL bit is used to select the HPI address register to be accessed.
		0	Selects the HPI write address register (HPIAW).
		1	Selects the HPI read address register (HPIAR).
10	Reserved	0	Reserved. Always write 0 to this bit.
9	DUALHPIA		Dual HPIA mode configuration bit. The CPU can access both HPI address registers separately, regardless of the DUALHPIA setting. (Regardless of this bit, dual HPIA mode is implied when the CPU has ownership of the HPI address registers).
		0	The two HPI address registers (HPIAW and HPIAR) operate as a single HPI address register in terms of host accesses.
		1	Dual HPIA mode operation is enabled.
8	HWOBSTAT		HWOB status. The value of the HWOB bit is also stored in this bit position. A write to the HWOB bit also updates HWOBSTAT.
		0	HWOB bit is logic 0.
		1	HWOB bit is logic 1.
7	HPIRST		HPI reset. Some HPI logic is held in reset when the HPIRST bit is set. The HPIRST bit must be cleared to 0 before data transactions can take place.
		0	HPI is released from reset.
		1	HPI is held in reset.
6-5	Reserved	2h	Reserved
4	FETCH		Host data fetch request bit. Only the host may write to FETCH. When a host writes a 1 to FETCH, a request is posted in the HPI to prefetch data into the read FIFO. Host and CPU reads of FETCH return a 0.
3	Reserved	1	Reserved
2	HINT		Processor-to-host interrupt. The CPU writes a 1 to HINT to generate a host interrupt. HINT has an inverted logic level to the $\overline{\text{UHPI_HINT}}$ pin. The host must write a 1 to HINT to clear the $\overline{\text{UHPI_HINT}}$ pin; writing a 0 to HINT by the host or processor has no effect.
		0	No effect.
		1	A CPU write generates a host interrupt ($\overline{\text{UHPI_HINT}}$ signal goes low). A host write sets the $\overline{\text{UHPI_HINT}}$ signal high (clears the interrupt).
1	DSPINT		Host-to-processor interrupt. The host writes a 1 to DSPINT to generate a processor interrupt; writing a 0 to DSPINT by the host or processor has no effect.
		0	No effect.
		1	A host write generates a processor interrupt.
0	HWOB		Halfword ordering bit. HWOB affects both data and address transfers. HWOB must be initialized before the first data or address register access.
		0	First halfword is most significant.
		1	First halfword is least significant.

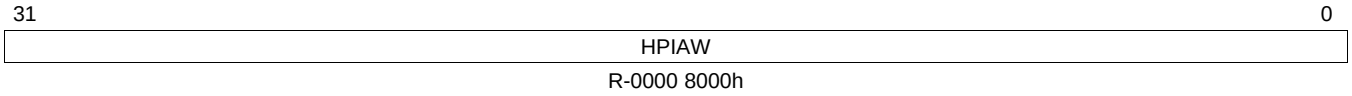
Copyright © 2013–2016, Texas Instruments Incorporated

20.3.9 Host Port Interface Write Address Register (HPIAW)

The HPI contains two 32-bit address registers: one for read operations (HPIAR) and one for write operations (HPIAW). The host port interface write address register (HPIAW) is shown in Figure 20-26 and described in Table 20-15. The HPI can be configured such that HPIAR and HPIAW act as a single 32-bit HPIA (single-HPIA mode) or as two separate 32-bit HPIAs (dual-HPIA mode) from the perspective of the host. For details about these HPIA modes, see Section 20.2.6.1.

Note that the addresses loaded into the HPI address registers can be configured by the HPIBYTEAD bit in the chip configuration 1 register (CFGCHIP1) of the system configuration module. If byte address is selected (HPIBYTEAD = 1), the address must be 32-bit word aligned (with the least-significant two bits equal to zero).

Figure 20-26. Host Port Interface Write Address Register (HPIAW)



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

Table 20-15. Host Port Interface Write Address Register (HPIAW) Field Descriptions

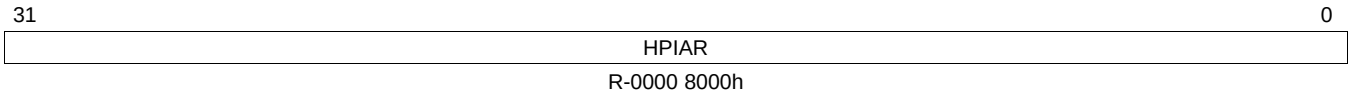
Bit	Field	Value	Description
31-0	HPIAW	0-FFFF FFFFh	Host port interface write address.

20.3.10 Host Port Interface Read Address Register (HPIAR)

The HPI contains two 32-bit address registers: one for read operations (HPIAR) and one for write operations (HPIAW). The host port interface read address register (HPIAR) is shown in Figure 20-27 and described in Table 20-16. The HPI can be configured such that HPIAR and HPIAW act as a single 32-bit HPIA (single-HPIA mode) or as two separate 32-bit HPIAs (dual-HPIA mode) from the perspective of the host. For details about these HPIA modes, see Section 20.2.6.1.

Note that the addresses loaded into the HPI address registers can be configured by the HPIBYTEAD bit in the chip configuration 1 register (CFGCHIP1) of the system configuration module. If byte address is selected (HPIBYTEAD = 1), the address must be 32-bit word aligned (with the least-significant two bits equal to zero).

Figure 20-27. Host Port Interface Read Address Register (HPIAR)



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

Table 20-16. Host Port Interface Read Address Register (HPIAR) Field Descriptions

Bit	Field	Value	Description
31-0	HPIAR	0-FFFF FFFFh	Host port interface read address.

Copyright © 2013–2016, Texas Instruments Incorporated

Inter-Integrated Circuit (I2C) Module

This chapter describes the inter-integrated circuit (I2C) peripheral. The scope of this chapter assumes that you are familiar with the Philips Semiconductors Inter-IC bus (I2C-bus) specification version 2.1.

Topic **Page**

21.1 Introduction **850**
21.2 Architecture **852**
21.3 Registers **864**

Copyright © 2013–2016, Texas Instruments Incorporated

21.1 Introduction

21.1.1 Purpose of the Peripheral

The I2C peripheral provides an interface between the SoC and other devices that are compliant with the I2C-bus specification and connected by way of an I2C-bus. External components that are attached to this two-wire serial bus can transmit and receive data that is up to eight bits wide both to and from the SoC through the I2C peripheral.

21.1.2 Features

The I2C peripheral has the following features:

- Compliance with the Philips Semiconductors I2C-bus specification (version 2.1):
  - Support for byte format transfer
  - 7-bit and 10-bit addressing modes
  - General call
  - START byte mode
  - Support for multiple master-transmitters and slave-receivers mode
  - Support for multiple slave-transmitters and master-receivers mode
  - Combined master transmit/receive and receive/transmit mode
  - I2C data transfer rate of from 10 kbps up to 400 kbps (Philips I2C rate)
- 2-bit to 8-bit format transfer
- Free data format mode
- One read DMA event and one write DMA event that the DMA can use
- Seven interrupts that the CPU can use
- Peripheral enable/disable capability

21.1.2.1 Features Not Supported

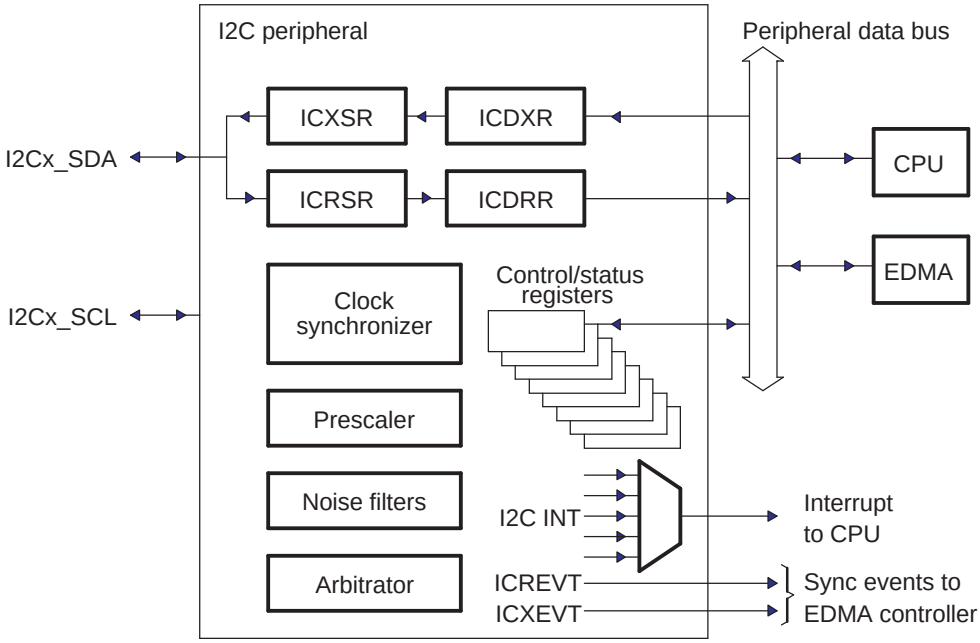
- High-speed mode
- CBUS-compatibility mode
- The combined format in 10-bit addressing mode (the I2C sends the slave address the second byte every time it sends the slave address the first byte).

Copyright © 2013–2016, Texas Instruments Incorporated

21.1.3 Functional Block Diagram

A block diagram of the I2C peripheral is shown in Figure 23-1. Refer to Section 21.2 for detailed information about the architecture of the I2C peripheral.

Figure 21-1. I2C Peripheral Block Diagram



21.1.4 Industry Standard(s) Compliance Statement

The I2C peripheral is compliant with the Philips Semiconductors Inter-IC bus (I2C-bus) specification version 2.1.

Copyright © 2013–2016, Texas Instruments Incorporated

21.2 Architecture

The I2C peripheral consists of the following primary blocks:

- A serial interface: one data pin (I2Cx_SDA) and one clock pin (I2Cx_SCL)
- Data registers to temporarily hold receive data and transmit data traveling between the I2Cx_SDA pin and the CPU or the EDMA controller
- Control and status registers
- A peripheral data bus interface to enable the CPU and the EDMA controller to access the I2C peripheral registers
- A clock synchronizer to synchronize the I2C input clock (from the processor clock generator) and the clock on the I2Cx_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 peripheral
- A noise filter on each of the two pins, I2Cx_SDA and I2Cx_SCL
- An arbitrator to handle arbitration between the I2C peripheral (when it is a master) and another master
- Interrupt generation logic, so that an interrupt can be sent to the CPU
- EDMA event generation logic, so that activity in the EDMA controller can be synchronized to data reception and data transmission in the I2C peripheral

Figure 23-1 shows the four registers used for transmission and reception. The CPU or the EDMA controller writes data for transmission to ICDXR and reads received data from ICDRR. When the I2C peripheral is configured as a transmitter, data written to ICDXR is copied to ICXSR and shifted out on the I2Cx_SDA pin one bit a time. When the I2C peripheral is configured as a receiver, received data is shifted into ICRSR and then copied to ICDRR.

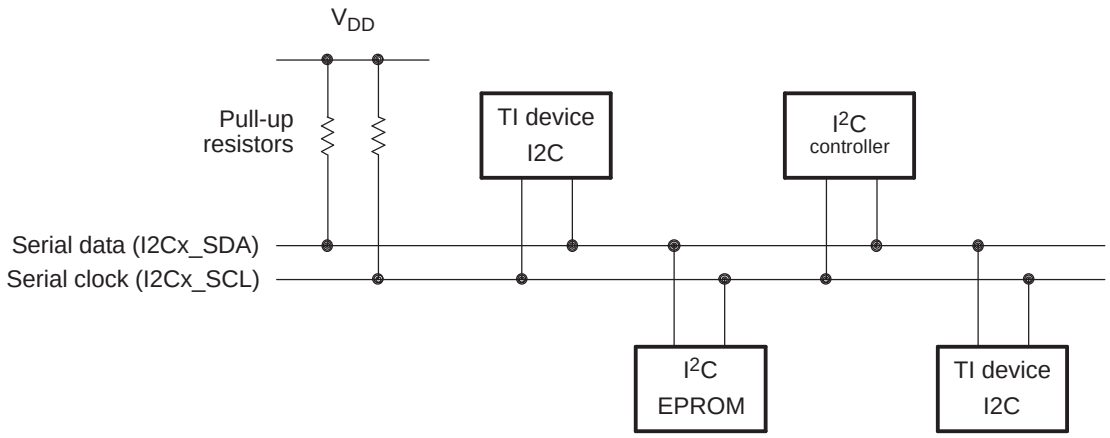
21.2.1 Bus Structure

Figure 23-1 shows how the I2C peripheral is connected to the I2C bus. The I2C bus is a multi-master bus that supports a multi-master mode. This allows more than one device capable of controlling the bus that is connected to it. A unique address recognizes each I2C device. Each I2C device can operate as either transmitter or receiver, depending on the function of the device. Devices that are connected to the I2C bus can be considered a master or slave when performing data transfers, in addition to being a transmitter or receiver.

NOTE: A master device is the device that initiates a data transfer on the bus and generates the clock signals to permit that transfer. Any device that is addressed by this master is considered a slave during this transfer.

An example of multiple I2C modules that are connected for a two-way transfer from one device to other devices is shown in Figure 21-2.

Figure 21-2. Multiple I2C Modules Connected



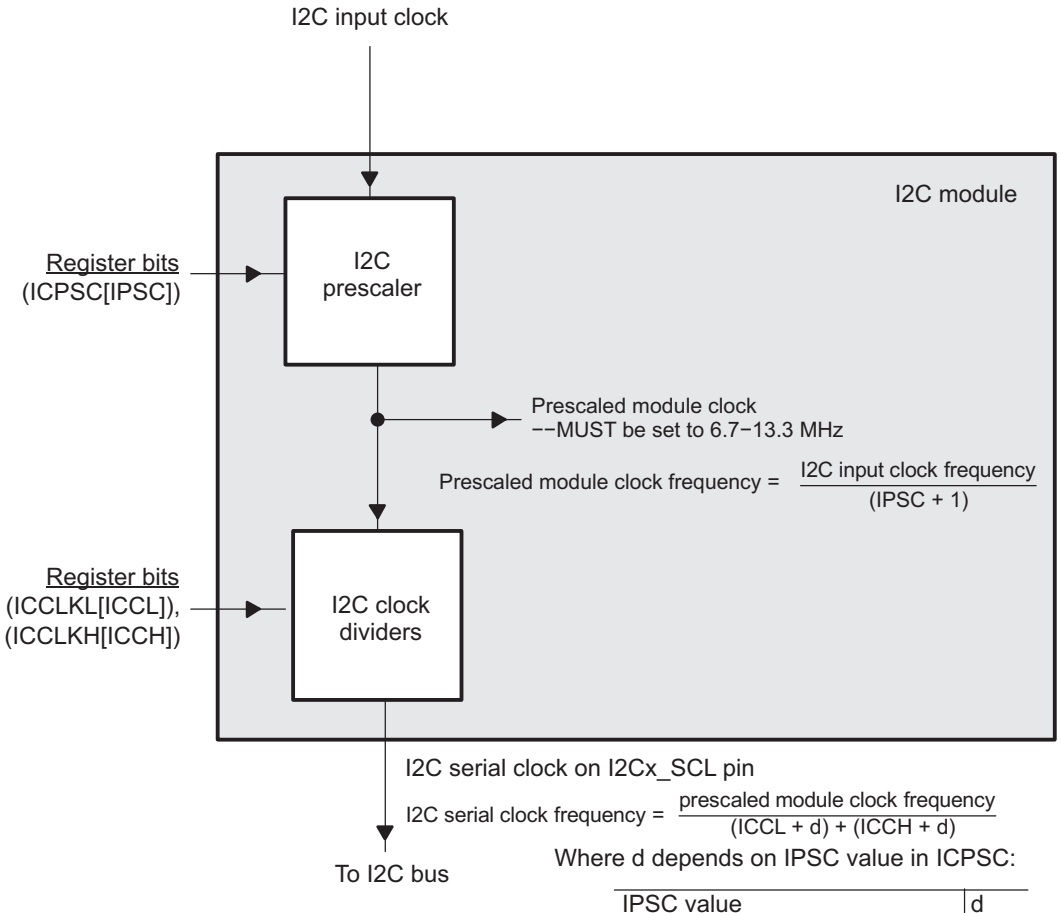
Copyright © 2013–2016, Texas Instruments Incorporated

21.2.2 Clock Generation

As shown in Figure 21-3, I2C input clock is fed to the I2C module. A programmable prescaler (IPSC bit in ICPSC) in the I2C module divides down the I2C input clock to produce a prescaled module clock. The prescaled module clock must be operated within the range of 6.7 to 13.3 MHz. The I2C clock dividers divide-down the high (ICCH bit in ICCLKH) and low portions (ICCL bit in ICCLKL) of the prescaled module clock signal to produce the I2C serial clock, which appears on the I2Cx_SCL pin when the I2C module is configured to be a master on the I2C bus.

The prescaler (IPSC bit in ICPSC) must only be initialized while the I2C module is in the reset state (IRS = 0 in ICMDR). The prescaled frequency only takes effect when the IRS bit in ICMDR is changed to 1. Changing the IPSC bit in ICPSC while IRS = 1 in ICMDR has no effect. Likewise, you must configure the I2C clock dividers (ICCH bit in ICCLKH and ICCL bit in ICCLKL) while the I2C module is still in reset (IRS = 0 in ICMDR).

Figure 21-3. Clocking Diagram for the I2C Peripheral



IPSC value	d
0	7
1	6
2h−FFh	5

CAUTION

Prescaled Module Clock Frequency Range:

The I2C module must be operated with a prescaled module clock frequency of 6.7 to 13.3 MHz. The I2C prescaler register (ICPSC) must be configured to this frequency range.

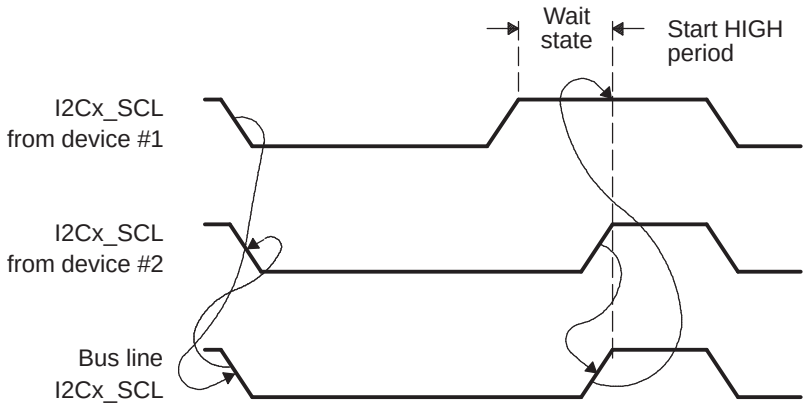
Copyright © 2013–2016, Texas Instruments Incorporated

21.2.3 Clock Synchronization

Only one master device generates the clock signal (I2Cx_SCL) under normal conditions. However, there are two or more masters during the arbitration procedure; and, you must synchronize the clock so that you can compare the data output. Figure 21-4 illustrates the clock synchronization. The wired-AND property of I2Cx_SCL means that a device that first generates a low period on I2Cx_SCL (device #1) 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 I2Cx_SCL is held low by the device with the longest low period. The other devices that finish their low periods must wait for I2Cx_SCL to be released before starting their high periods. A synchronized signal on I2Cx_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. This way, a slave slows down a fast master and the slow device creates enough time to store a received data word or to prepare a data word that you are going to transmit.

Figure 21-4. Synchronization of Two I2C Clock Generators During Arbitration



21.2.4 Signal Descriptions

The I2C peripheral has a serial data pin (I2Cx_SDA) and a serial clock pin (I2Cx_SCL) for data communication, as shown in Figure 23-1. These two pins carry information between the device and other devices that are connected to the I2C-bus. The I2Cx_SDA and I2Cx_SCL pins both are bi-directional. 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.

See your device-specific data manual for additional timing and electrical specifications for these pins.

21.2.4.1 Input and Output Voltage Levels

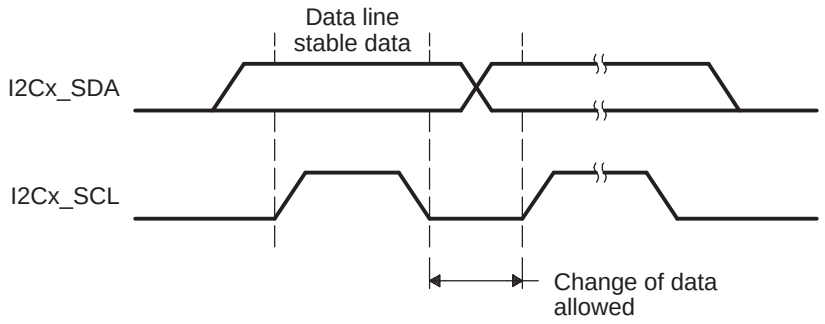
The master device generates one clock pulse for each data bit that is transferred. Due to a variety of different technology devices that can be connected to the I2C-bus, the levels of logic 0 (low) and logic 1 (high) are not fixed and depend on the associated power supply level. See your device-specific data manual for more information.

Copyright © 2013–2016, Texas Instruments Incorporated

21.2.4.2 Data Validity

The data on I2Cx_SDA must be stable during the high period of the clock (see Figure 21-5). The high or low state of the data line, I2Cx_SDA, can change only when the clock signal on I2Cx_SCL is low.

Figure 21-5. Bit Transfer on the I2C-Bus



21.2.5 START and STOP Conditions

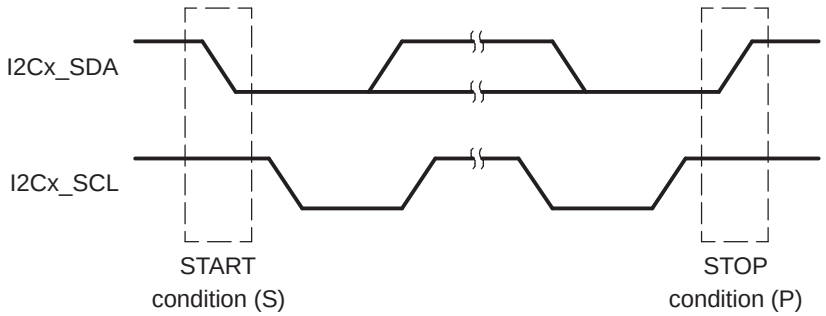
The I2C peripheral can generate START and STOP conditions when the peripheral is configured to be a master on the I2C-bus, as shown in Figure 21-6:

- The START condition is defined as a high-to-low transition on the I2Cx_SDA line while I2Cx_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 I2Cx_SDA line while I2Cx_SCL is high. A master drives this condition to indicate the end of a data transfer.

The I2C-bus is considered busy after a START condition and before a subsequent STOP condition. The bus busy (BB) bit of ICSTR is 1. The bus is considered free between a STOP condition and the next START condition. The BB is 0.

The master mode (MST) bit and the START condition (STT) bit in ICMDR must both be 1 for the I2C peripheral to start a data transfer with a START condition. The STOP condition (STP) bit must be set to 1 for the I2C peripheral to end a data transfer with a STOP condition. A repeated START condition generates when BB is set to 1 and STT is also set to 1. See Section 21.3.9 for a description of ICMDR (including the MST, STT, and STP bits).

Figure 21-6. I2C Peripheral START and STOP Conditions



Copyright © 2013–2016, Texas Instruments Incorporated

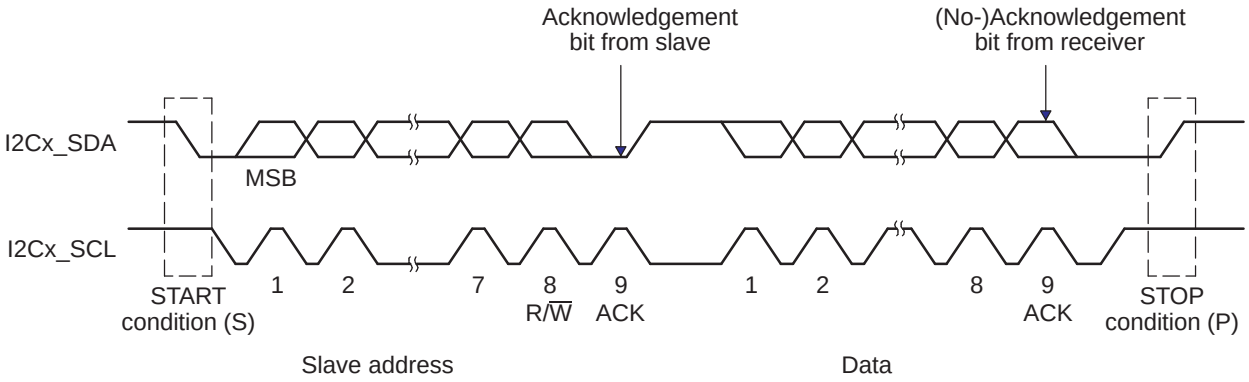
21.2.6 Serial Data Formats

Figure 21-7 shows an example of a data transfer on the I2C-bus. The I2C peripheral supports 1-bit to 8-bit data values. Figure 21-7 is shown in an 8-bit data format (BC = 000 in ICMDR). Each bit put on the I2Cx_SDA line is equivalent to one pulse on the I2Cx_SCL line. The data is always transferred with the most-significant bit (MSB) first. The number of data values that can be transmitted or received is unrestricted; however, the transmitters and receivers must agree on the number of data values being transferred.

The I2C peripheral supports the following data formats:

- 7-bit addressing mode
- 10-bit addressing mode
- Free data format mode

Figure 21-7. I2C Peripheral Data Transfer



21.2.6.1 7-Bit Addressing Format

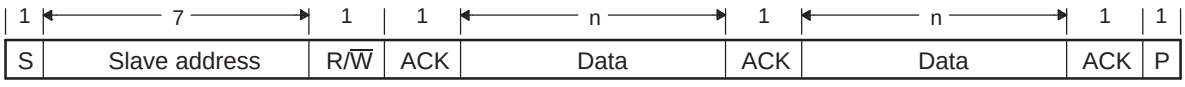
In the 7-bit addressing format (Figure 21-8), the first byte after a START condition (S) consists of a 7-bit slave address followed by a R/$\overline{W}$ bit. The R/$\overline{W}$ bit determines the direction of the data.

- 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 the R/$\overline{W}$ bit. If the slave inserts the ACK bit, *n* bits of data from the transmitter (master or slave, depending on the R/$\overline{W}$ bit) follow it. *n* is a number from 1 to 8 that the bit count (BC) bits of ICMDR determine. The receiver inserts an ACK bit after the data bits have been transferred.

Write a 0 to the expanded address enable (XA) bit of ICMDR to select the 7-bit addressing format.

Figure 21-8. I2C Peripheral 7-Bit Addressing Format (FDF = 0, XA = 0 in ICMDR)



n = The number of data bits (from 1 to 8) specified by the bit count (BC) field of ICM DR.

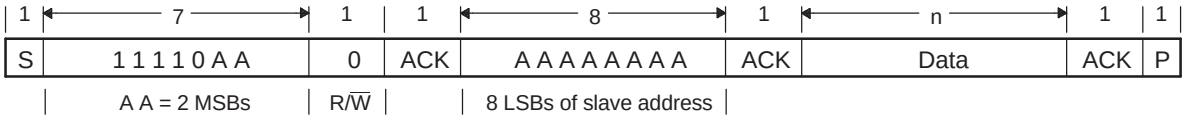
Copyright © 2013–2016, Texas Instruments Incorporated

21.2.6.2 10-Bit Addressing Format

The 10-bit addressing format (Figure 21-9) is like 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 (ACK) 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 information about using 10-bit addressing, see the Philips Semiconductors I2C-bus specification.)

Write 1 to the XA bit of ICMDR to select the 10-bit addressing format.

Figure 21-9. I2C Peripheral 10-Bit Addressing Format With Master-Transmitter Writing to Slave-Receiver (FDF = 0, XA = 1 in ICMDR)



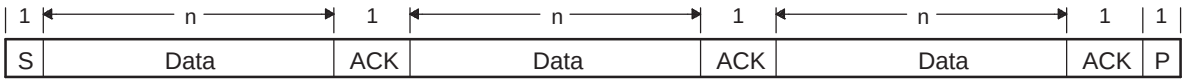
n = The number of data bits (from 1 to 8) specified by the bit count (BC) field of ICMDR.

21.2.6.3 Free Data Format

In the free data format (Figure 21-10), the first bits after a START condition (S) are a data word. An ACK bit is inserted after each data word. The data word can be from 1 to 8 bits, depending on the bit count (BC) bits of ICMDR. 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 ICMDR.

Figure 21-10. I2C Peripheral Free Data Format (FDF = 1 in ICMDR)

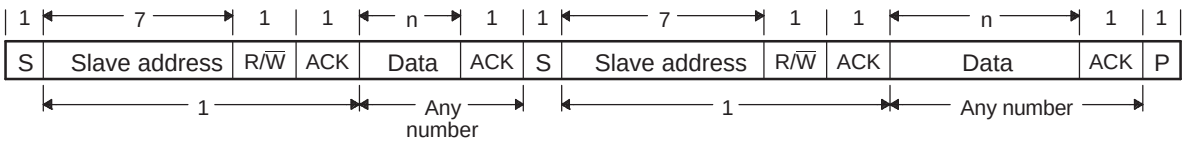


n = The number of data bits (from 1 to 8) specified by the bit count (BC) field of ICMDR.

21.2.6.4 Using a Repeated START Condition

The repeated START condition can be used with the 7-bit addressing, 10-bit addressing, and free data formats. The 7-bit addressing format using a repeated START condition (S) is shown in Figure 21-11. At the end of each data word, the master can drive another START condition. Using this capability, a master can transmit/receive any number of data words before driving a STOP condition. The length of a data word can be from 1 to 8 bits and is selected with the bit count (BC) bits of ICMDR.

Figure 21-11. I2C Peripheral 7-Bit Addressing Format With Repeated START Condition (FDF = 0, XA = 0 in ICMDR)



n = The number of data bits (from 1 to 8) specified by the bit count (BC) field of ICMDR.

Copyright © 2013–2016, Texas Instruments Incorporated

21.2.7 Operating Modes

The I2C peripheral has four basic operating modes to support data transfers as a master and as a slave. See Table 21-1 for the names and descriptions of the modes.

If the I2C peripheral 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 peripheral must remain a master-transmitter. In order to receive data from a slave, the I2C peripheral must be changed to the master-receiver mode.

If the I2C peripheral 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 peripheral, the peripheral must remain a slave-receiver. If the master has requested data from the I2C peripheral, the peripheral must be changed to the slave-transmitter mode.

Table 21-1. Operating Modes of the I2C Peripheral

Operating Mode	Description
Slave-receiver mode	The I2C peripheral is a slave and receives data from a master. All slave modules begin in this mode. In this mode, serial data bits received on I2Cx_SDA are shifted in with the clock pulses that are generated by the master. As a slave, the I2C peripheral does not generate the clock signal, but it can hold I2Cx_SCL low while the intervention of the processor is required (RSFULL = 1 in ICSTR) after data has been received.
Slave-transmitter mode	The I2C peripheral is a slave and transmits data to a master. This mode can only be entered from the slave-receiver mode; the I2C peripheral must first receive a command from the master. When you are using any of the 7-bit/10-bit addressing formats, the I2C peripheral enters its slave-transmitter mode if the slave address is the same as its own address (in ICOAR) and the master has transmitted R/$\overline{W}$ = 1. As a slave-transmitter, the I2C peripheral then shifts the serial data out on I2Cx_SDA with the clock pulses that are generated by the master. While a slave, the I2C peripheral does not generate the clock signal, but it can hold I2Cx_SCL low while the intervention of the processor is required (XSMT = 0 in ICSTR) after data has been transmitted.
Master-receiver mode	The I2C peripheral is a master and receives data from a slave. This mode can only be entered from the master-transmitter mode; the I2C peripheral must first transmit a command to the slave. When you are using any of the 7-bit/10-bit addressing formats, the I2C peripheral enters its master-receiver mode after transmitting the slave address and R/$\overline{W}$ = 1. Serial data bits on I2Cx_SDA are shifted into the I2C peripheral with the clock pulses generated by the I2C peripheral on I2Cx_SCL. The clock pulses are inhibited and I2Cx_SCL is held low when the intervention of the processor is required (RSFULL = 1 in ICSTR) after data has been received.
Master-transmitter mode	The I2C peripheral is a master and transmits control information and data to a slave. All master modules begin in this mode. In this mode, data assembled in any of the 7-bit/10-bit addressing formats is shifted out on I2Cx_SDA. The bit shifting is synchronized with the clock pulses generated by the I2C peripheral on I2Cx_SCL. The clock pulses are inhibited and I2Cx_SCL is held low when the intervention of the processor is required (XSMT = 0 in ICSTR) after data has been transmitted.

Copyright © 2013–2016, Texas Instruments Incorporated

21.2.8 NACK Bit Generation

When the I2C peripheral is a receiver (master or slave), it can acknowledge or ignore bits sent by the transmitter. To ignore any new bits, the I2C peripheral must send a no-acknowledge (NACK) bit during the acknowledge cycle on the bus. Table 21-2 summarizes the various ways the I2C peripheral sends a NACK bit.

Table 21-2. Ways to Generate a NACK Bit

I2C Peripheral Condition	NACK Bit Generation	
	Basic	**Optional**
Slave-receiver mode	• Disable data transfers (STT = 0 in ICSTR). • Allow an overrun condition (RSFULL = 1 in ICSTR). • Reset the peripheral (IRS = 0 in ICMDR) .	Set the NACKMOD bit of ICMDR before the rising edge of the last data bit you intend to receive.
Master-receiver mode AND Repeat mode (RM = 1 in ICMDR)	• Generate a STOP condition (STOP = 1 in ICMDR). • Reset the peripheral (IRS = 0 in ICMDR).	Set the NACKMOD bit of ICMDR before the rising edge of the last data bit you intend to receive.
Master-receiver mode AND Nonrepeat mode (RM = 0 in ICMDR)	• If STP = 1 in ICMDR, allow the internal data counter to count down to 0 and force a STOP condition. • If STP = 0, make STP = 1 to generate a STOP condition. • Reset the peripheral (IRS = 0 in ICMDR).	Set the NACKMOD bit of ICMDR before the rising edge of the last data bit you intend to receive.

Copyright © 2013–2016, Texas Instruments Incorporated

21.2.9 Arbitration

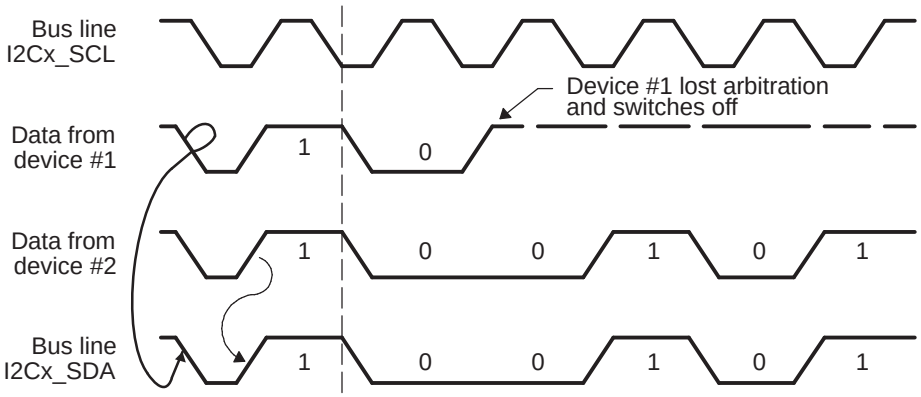
If two or more master-transmitters simultaneously start a transmission on the same bus, an arbitration procedure is invoked. The arbitration procedure uses the data presented on the serial data bus (I2Cx_SDA) by the competing transmitters. Figure 21-12 illustrates the arbitration procedure between two devices. The first master-transmitter, which drives I2Cx_SDA high, is overruled by another master-transmitter that drives I2Cx_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 peripheral is the losing master, it switches to the slave-receiver mode, sets the arbitration lost (AL) flag, and generates the arbitration-lost interrupt.

If during a serial transfer the arbitration procedure is still in progress when a repeated START condition or a STOP condition is transmitted to I2Cx_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

Figure 21-12. Arbitration Procedure Between Two Master-Transmitters



Copyright © 2013–2016, Texas Instruments Incorporated

21.2.10 Reset Considerations

The I2C peripheral has two reset sources: software reset and hardware reset.

21.2.10.1 Software Reset Considerations

To reset the I2C peripheral, write 0 to the I2C reset (IRS) bit in the I2C mode register (ICMDR). All status bits in the I2C interrupt status register (ICSTR) are forced to their default values, and the I2C peripheral remains disabled until IRS is changed to 1. The I2Cx_SDA and I2Cx_SCL pins are in the high-impedance state.

NOTE: If the IRS bit is cleared to 0 during a transfer, this can cause the I2C bus to hang (I2Cx_SDA and I2Cx_SCL are in the high-impedance state).

21.2.10.2 Hardware Reset Considerations

When a hardware reset occurs, all the registers of the I2C peripheral are set to their default values and the I2C peripheral remains disabled until the I2C reset (IRS) bit in the I2C mode register (ICMDR) is changed to 1.

NOTE: The IRS bit must be cleared to 0 while you configure/reconfigure the I2C peripheral. Forcing IRS to 0 can be used to save power and to clear error conditions.

21.2.11 Initialization

Proper I2C initialization is required prior to starting communication with other I2C device(s). Unless a fully fledged driver is in place, you need to determine the required I2C configuration needed (for example, Master Receiver, etc.) and configure the I2C controller with the desired settings. Enabling the I2C clock should be the first task. Then the I2C controller is placed in reset. You now are ready to configure the I2C controller. Once configuration is done, you need to enable the I2C controller by releasing the controller from reset. Prior to starting communication, you need to make sure that all status bits are cleared and no pending interrupts exist. Once the bus is determined to be available (the bus is not busy), the I2C is ready to proceed with the desired communication.

Copyright © 2013–2016, Texas Instruments Incorporated

21.2.11.1 Configuring the I2C in Master Receiver Mode and Servicing Receive Data via CPU

The following initialization procedure is for the I2C controller configured in Master Receiver mode. The CPU is used to move data from the I2C receive register to CPU memory (memory accessible by the CPU).

1. Enable I2C clock from the Power and Sleep Controller, if it is driven by the Power and Sleep Controller (see the *Power and Sleep Controller (PSC)* chapter).
2. Place I2C in reset (clear IRS = 0 in ICMDR).
3. Configure ICMDR:
   - Configure I2C as Master (MST = 1).
   - Indicate the I2C configuration to be used; for example, Data Receiver (TRX = 0)
   - Indicate 7-bit addressing is to be used (XA = 0).
   - Disable repeat mode (RM = 0).
   - Disable loopback mode (DLB = 0).
   - Disable free data format (FDF = 0).
   - Optional: Disable start byte mode if addressing a fully fledged I2C device (STB = 0).
   - Set number of bits to transfer to be 8 bits (BC = 0).
4. Configure Slave Address: the I2C device this I2C master would be addressing (ICSAR = 7BIT ADDRESS).
5. Configure the peripheral clock operation frequency (ICPSC). This value should be selected in such a way that the frequency is between 6.7 and 13.3 MHz.
6. Configure I2C master clock frequency:
   - Configure the low-time divider value (ICCLKL).
   - Configure the high-time divider value (ICCLKH).
7. Make sure the interrupt status register (ICSTR) is cleared:
   - Read ICSTR and write it back (write 1 to clear) ICSTR = ICSTR
   - Read ICIVR until it is 0.
8. Take I2C controller out of reset: enable I2C controller (set IRS bit = 1 in ICMDR).
9. Wait until bus busy bit is cleared (BB = 0 in ICSTR).
10. Generate a START event, followed by Slave Address, etc. (set STT = 1 in ICMDR).
11. Wait until data is received (ICRRDY = 1 in ICSTR).
12. Read data:
    - If ICRRDY = 1 in ICSTR, then read ICDRR.
    - Perform the previous two steps until receiving one byte short of the entire byte expecting to receive.
13. Configure the I2C controller not to generate an ACK on the next/final byte reception: set NACKMOD bit for the I2C to generate a NACK on the last byte received (set NACKMOD = 1 in ICMDR).
14. End transfer/release bus when transfer is done. Generate a STOP event (set STP = 1 in ICMDR).

21.2.12 Interrupt Support

The I2C peripheral is capable of interrupting the CPU. The CPU can determine which I2C events caused the interrupt by reading the I2C interrupt vector register (ICIVR). ICIVR contains a binary-coded interrupt vector type to indicate which interrupt has occurred. Reading ICIVR clears the interrupt flag; if other interrupts are pending, a new interrupt is generated. If there is more than one pending interrupt flag, reading ICIVR clears the highest-priority interrupt flag.

Copyright © 2013–2016, Texas Instruments Incorporated

21.2.12.1 Interrupt Events and Requests

The I2C peripheral can generate the interrupts described in Table 21-3. Each interrupt has a flag bit in the I2C interrupt status register (ICSTR) and a mask bit in the interrupt mask register (ICIMR). When one of the specified events occurs, its flag bit is set. If the corresponding mask bit is 0, the interrupt request is blocked; if the mask bit is 1, the request is forwarded to the CPU as an I2C interrupt.

Table 21-3. Descriptions of the I2C Interrupt Events

I2C Interrupt	Initiating Event
Arbitration-lost interrupt (AL)	Generated when the I2C arbitration procedure is lost or illegal START/STOP conditions occur
No-acknowledge interrupt (NACK)	Generated when the master I2C does not receive any acknowledge from the receiver
Registers-ready-for-access interrupt (ARDY)	Generated by the I2C when the previously programmed address, data and command have been performed and the status bits have been updated. This interrupt is used to let the controlling processor know that the I2C registers are ready to be accessed.
Receive interrupt/status (ICRINT and ICRRDY)	Generated when the received data in the receive-shift register (ICRSR) has been copied into the ICDRR. The ICRRDY bit can also be polled by the CPU to read the received data in the ICDRR.
Transmit interrupt/status (ICXINT and ICXRDY)	Generated when the transmitted data has been copied from ICDXR to the transmit-shift register (ICXSR) and shifted out on the I2Cx_SDA pin. This bit can also polled by the CPU to write the next transmitted data into the ICDXR.
Stop-Condition-Detection interrupt (SCD)	Generated when a STOP condition has been detected
Address-as-Slave interrupt (AAS)	Generated when the I2C has recognized its own slave address or an address of all (8) zeros.

21.2.13 DMA Events Generated by the I2C Peripheral

For the EDMA controller to handle transmit and receive data, the I2C peripheral generates the following two EDMA events. Activity in EDMA channels can be synchronized to these events.

- Receive event (ICREVT): When receive data has been copied from the receive shift register (ICRSR) to the data receive register (ICDRR), the I2C peripheral sends an REVT signal to the EDMA controller. In response, the EDMA controller can read the data from ICDRR.
- Transmit event (ICXEVT): When transmit data has been copied from the data transmit register (ICDXR) to the transmit shift register (ICXSR), the I2C peripheral sends an XEVT signal to the EDMA controller. In response, the EDMA controller can write the next transmit data value to ICDXR.

21.2.14 Power Management

The I2C peripheral can be placed in reduced-power modes to conserve power during periods of low activity. The power management of the I2C peripheral is controlled by the processor Power and Sleep Controller (PSC). The PSC acts as a master controller for power management for all of the peripherals on the device. For detailed information on power management procedures using the PSC, see the *Power and Sleep Controller (PSC)* chapter.

21.2.15 Emulation Considerations

The response of the I2C events to emulation suspend events (such as halts and breakpoints) is controlled by the FREE bit in the I2C mode register (ICMDR). The I2C peripheral either stops exchanging data (FREE = 0) or continues to run (FREE = 1) when an emulation suspend event occurs. How the I2C peripheral terminates data transactions is affected by whether the I2C peripheral is acting as a master or a slave. For more information, see the description of the FREE bit in ICMDR (see Section 21.3.9).

Copyright © 2013–2016, Texas Instruments Incorporated

21.3 Registers

Table 27-8 lists the memory-mapped registers for the inter-integrated circuit (I2C) peripheral. See your device-specific data manual for the memory address of these registers. All other register offset addresses not listed in Table 27-8 should be considered as reserved locations and the register contents should not be modified.

Table 21-4. Inter-Integrated Circuit (I2C) Registers

Offset	Acronym	Register Description	Section
0h	ICOAR	I2C Own Address Register	Section 21.3.1
4h	ICIMR	I2C Interrupt Mask Register	Section 21.3.2
8h	ICSTR	I2C Interrupt Status Register	Section 21.3.3
Ch	ICCLKL	I2C Clock Low-Time Divider Register	Section 21.3.4
10h	ICCLKH	I2C Clock High-Time Divider Register	Section 21.3.4
14h	ICCNT	I2C Data Count Register	Section 21.3.5
18h	ICDRR	I2C Data Receive Register	Section 21.3.6
1Ch	ICSAR	I2C Slave Address Register	Section 21.3.7
20h	ICDXR	I2C Data Transmit Register	Section 21.3.8
24h	ICMDR	I2C Mode Register	Section 21.3.9
28h	ICIVR	I2C Interrupt Vector Register	Section 21.3.10
2Ch	ICEMDR	I2C Extended Mode Register	Section 21.3.11
30h	ICPSC	I2C Prescaler Register	Section 21.3.12
34h	REVID1	I2C Revision Identification Register 1	Section 21.3.13
38h	REVID2	I2C Revision Identification Register 2	Section 21.3.13
3Ch	ICDMAC	I2C DMA Control Register	Section 21.3.15
48h	ICPFUNC	I2C Pin Function Register	Section 21.3.16
4Ch	ICPDIR	I2C Pin Direction Register	Section 21.3.17
50h	ICPDIN	I2C Pin Data In Register	Section 21.3.18
54h	ICPDOUT	I2C Pin Data Out Register	Section 21.3.19
58h	ICPDSET	I2C Pin Data Set Register	Section 21.3.20
5Ch	ICPDCLR	I2C Pin Data Clear Register	Section 21.3.21

Copyright © 2013–2016, Texas Instruments Incorporated

21.3.1 I2C Own Address Register (ICOAR)

The I2C own address register (ICOAR) is used 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 ICMDR), only bits 6-0 are used; bits 9-7 are ignored. ICOAR is shown in Figure 21-13 and described in Table 21-5.

Figure 21-13. I2C Own Address Register (ICOAR)

31	16
Reserved	
R-0	

15	10	9	0
Reserved		OADDR	
R-0		R/W-0	

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

Table 21-5. I2C Own Address Register (ICOAR) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
9-0	OADDR	0-3FFh	Own slave address. Provides the slave address of the I2C. In 7-bit addressing mode (XA = 0 in ICMDR): bits 6-0 provide the 7-bit slave address of the I2C. Bits 9-7 are ignored. In 10-bit addressing mode (XA = 1 in ICMDR): bits 9-0 provide the 10-bit slave address of the I2C.

Copyright © 2013–2016, Texas Instruments Incorporated

21.3.2 I2C Interrupt Mask Register (ICIMR)

The I2C interrupt mask register (ICIMR) is used to individually enable or disable I2C interrupt requests. ICIMR is shown in Figure 21-14 and described Table 21-6.

Figure 21-14. I2C Interrupt Mask Register (ICIMR)

31							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
Reserved	AAS	SCD	ICXRDY	ICRRDY	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 21-6. I2C Interrupt Mask Register (ICIMR) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
6	AAS		Address-as-slave interrupt enable bit.
		0	Interrupt request is disabled.
		1	Interrupt request is enabled.
5	SCD		Stop condition detected interrupt enable bit.
		0	Interrupt request is disabled.
		1	Interrupt request is enabled.
4	ICXRDY		Transmit-data-ready interrupt enable bit.
		0	Interrupt request is disabled.
		1	Interrupt request is enabled.
3	ICRRDY		Receive-data-ready interrupt enable bit.
		0	Interrupt request is disabled.
		1	Interrupt request is enabled.
2	ARDY		Register-access-ready interrupt enable bit.
		0	Interrupt request is disabled.
		1	Interrupt request is enabled.
1	NACK		No-acknowledgment interrupt enable bit.
		0	Interrupt request is disabled.
		1	Interrupt request is enabled.
0	AL		Arbitration-lost interrupt enable bit
		0	Interrupt request is disabled.
		1	Interrupt request is enabled.

Copyright © 2013–2016, Texas Instruments Incorporated

21.3.3 I2C Interrupt Status Register (ICSTR)

The I2C interrupt status register (ICSTR) is used to determine which interrupt has occurred and to read status information. ICSTR is shown in Figure 21-15 and described in Table 21-7.

Figure 21-15. I2C Interrupt Status Register (ICSTR)

31							16
Reserved							
R-0							
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/W1C-0	R-0	R-1	R-0	R-0
7	6	5	4	3	2	1	0
Reserved		SCD	ICXRDY	ICRRDY	ARDY	NACK	AL
R-0		R/W1C-0	R/W1C-1	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing 0 has no effect); -*n* = value after reset

Table 21-7. I2C Interrupt Status Register (ICSTR) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
14	SDIR		Slave direction bit. In digital-loopback mode (DLB), the SDIR bit is cleared to 0.
		0	I2C is acting as a master-transmitter/receiver or a slave-receiver. SDIR is cleared by one of the following events: • A STOP or a START condition. • SDIR is manually cleared. To clear this bit, write a 1 to it.
		1	I2C is acting as a slave-transmitter.
13	NACKSNT		No-acknowledgment sent bit. NACKSNT bit is used when the I2C 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 21.3.9).
		0	NACK is not sent. NACKSNT is cleared by one of the following events: • It is manually cleared. To clear this bit, write a 1 to it. • The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when the processor is reset).
		1	NACK is sent. A no-acknowledge bit was sent during the acknowledge cycle on the I2C-bus.
12	BB		Bus busy bit. BB bit indicates whether the I2C-bus is busy or is free for another data transfer. In the master mode, BB is controlled by the software.
		0	Bus is free. BB is cleared by one of the following events: • The I2C receives or transmits a STOP bit (bus free). • BB is manually cleared. To clear this bit, write a 1 to it. • The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when the processor is reset).
		1	Bus is busy. When the STT bit in ICMDR is set to 1, a restart condition is generated. BB is set by one of the following events: • The I2C has received or transmitted a START bit on the bus. • I2Cx_SCL is in a low state and the IRS bit in ICMDR is 0.
11	RSFULL		Receive shift register full bit. RSFULL indicates an overrun condition during reception. Overrun occurs when the receive shift register (ICRSR) is full with new data but the previous data has not been read from the data receive register (ICDRR). The new data will not be copied to ICDRR until the previous data is read. As new bits arrive from the I2Cx_SDA pin, they overwrite the bits in ICRSR.
		0	No overrun is detected. RSFULL is cleared by one of the following events: • ICDRR is read. • The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when the processor is reset).
		1	Overrun is detected.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 21-7. I2C Interrupt Status Register (ICSTR) Field Descriptions (continued)

Bit	Field	Value	Description
10	XSMT		Transmit shift register empty bit. XSMT indicates that the transmitter has experienced underflow. Underflow occurs when the transmit shift register (ICXSR) is empty but the data transmit register (ICDXR) has not been loaded since the last ICDXR-to-ICXSR transfer. The next ICDXR-to-ICXSR transfer will not occur until new data is in ICDXR. If new data is not transferred in time, the previous data may be re-transmitted on the I2Cx_SDA pin.
		0	Underflow is detected.
		1	No underflow is detected. XSMT is set by one of the following events: • Data is written to ICDXR. • The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when the processor is reset).
9	AAS		Addressed-as-slave bit.
		0	The AAS bit has been cleared by a repeated START condition or by a STOP condition.
		1	AAS is set by one of the following events: • I2C has recognized its own slave address or an address of all zeros (general call). • The first data word has been received in the free data format (FDF = 1 in ICMDR).
8	AD0		Address 0 bit.
		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 indicates when a STOP condition has been detected on the I2C bus. The STOP condition could be generated by the I2C or by another I2C device connected to the bus.
		0	No STOP condition has been detected. SCD is cleared by one of the following events: • By reading the INTCODE bits in ICIVR as 110b. • SCD is manually cleared. To clear this bit, write a 1 to it.
		1	A STOP condition has been detected.
4	ICXRDY		Transmit-data-ready interrupt flag bit. ICXRDY indicates that the data transmit register (ICDXR) is ready to accept new data because the previous data has been copied from ICDXR to the transmit shift register (ICXSR). The CPU can poll ICXRDY or use the XRDY interrupt request.
		0	ICDXR is not ready. ICXRDY is cleared by one of the following events: • Data is written to ICDXR. • ICXRDY is manually cleared. To clear this bit, write a 1 to it.
		1	ICDXR is ready. Data has been copied from ICDXR to ICXSR. ICXRDY is forced to 1 when the I2C is reset.
3	ICRRDY		Receive-data-ready interrupt flag bit. ICRRDY indicates that the data receive register (ICDRR) is ready to be read because data has been copied from the receive shift register (ICRSR) to ICDRR. The CPU can poll ICRRDY or use the RRDY interrupt request.
		0	ICDRR is not ready. ICRRDY is cleared by one of the following events: • ICDRR is read. • ICRRDY is manually cleared. To clear this bit, write a 1 to it. • The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when the processor is reset).
		1	ICDRR is ready. Data has been copied from ICRSR to ICDRR.
2	ARDY		Register-access-ready interrupt flag bit (only applicable when the I2C is in the master mode). ARDY indicates that the I2C 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.
		0	The registers are not ready to be accessed. ARDY is cleared by one of the following events: • The I2C starts using the current register contents. • ARDY is manually cleared. To clear this bit, write a 1 to it. • The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when the processor is reset).
		1	The registers are ready to be accessed. This bit is set after the slave address appears on the I2C bus. • In the nonrepeat mode (RM = 0 in ICMDR): If STP = 0 in ICMDR, ARDY is set when the internal data counter counts down to 0. If STP = 1, ARDY is not affected (instead, the I2C generates a STOP condition when the counter reaches 0). • In the repeat mode (RM = 1): ARDY is set at the end of each data word transmitted from ICDXR.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 21-7. I2C Interrupt Status Register (ICSTR) Field Descriptions (continued)

Bit	Field	Value	Description
1	NACK		No-acknowledgment interrupt flag bit. NACK applies when the I2C is a transmitter (master or slave). NACK indicates whether the I2C 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.
		0	ACK received/NACK is not received. NACK is cleared by 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 vector register (ICIVR) when the register contains the code for a NACK interrupt. • The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when the processor is reset).
		1	NACK bit is received. The hardware detects that a no-acknowledge (NACK) bit has been received. **Note:** While the I2C performs a general call transfer, NACK is 1, even if one or more slaves send acknowledgment.
0	AL		Arbitration-lost interrupt flag bit (only applicable when the I2C is a master-transmitter). AL primarily indicates when the I2C has lost an arbitration contest with another master-transmitter. The CPU can poll AL or use the AL interrupt request.
		0	Arbitration is not lost. AL is cleared by one of the following events: • AL is manually cleared. To clear this bit, write a 1 to it. • The CPU reads the interrupt vector register (ICIVR) when the register contains the code for an AL interrupt. • The I2C is reset (either when 0 is written to the IRS bit of ICMDR or when the processor is reset).
		1	Arbitration is lost. AL is set by one of the following events: • The I2C senses that it has lost an arbitration with two or more competing transmitters that started a transmission almost simultaneously. • The I2C attempts to start a transfer while the BB (bus busy) bit is set to 1. When AL is set to 1, the MST and STP bits of ICMDR are cleared, and the I2C becomes a slave-receiver.

Copyright © 2013–2016, Texas Instruments Incorporated

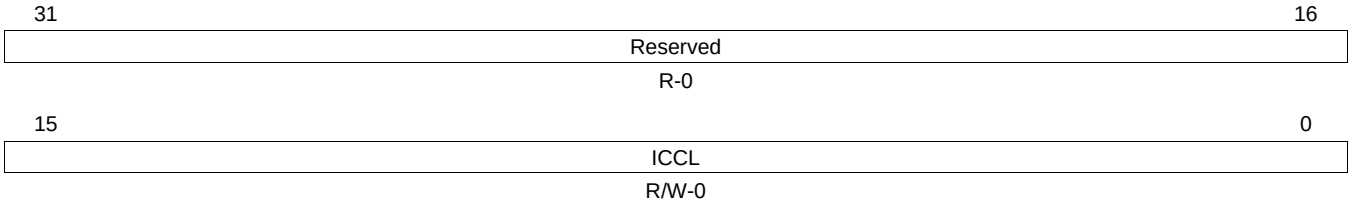
21.3.4 I2C Clock Divider Registers (ICCLKL and ICCLKH)

When the I2C is a master, the prescaled module clock is divided down for use as the I2C serial clock on the I2Cx_SCL pin. The shape of the I2C serial clock depends on two divide-down values, ICCL and ICCH. For detailed information on how these values are programmed, see Section 21.2.2.

21.3.4.1 I2C Clock Low-Time Divider Register (ICCLKL)

For each I2C serial clock cycle, ICCL in the I2C clock low-time divider register (ICCLKL) determines the amount of time the signal is low. ICCLKL must be configured while the I2C is still in reset (IRS = 0 in ICMDR). ICCLKL is shown in Figure 21-16 and described in Table 21-8.

Figure 21-16. I2C Clock Low-Time Divider Register (ICCLKL)



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

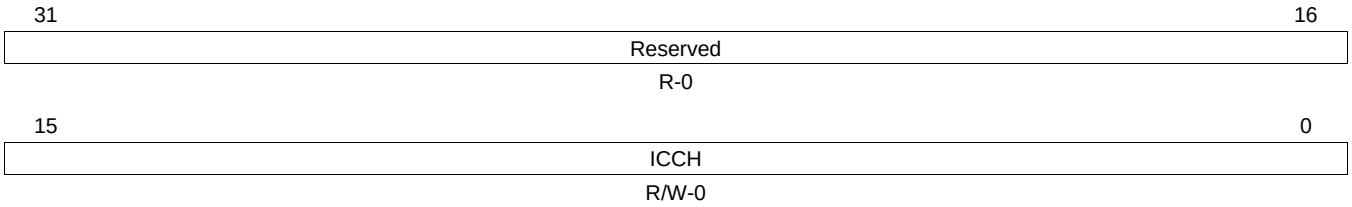
Table 21-8. I2C Clock Low-Time Divider Register (ICCLKL) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
15-0	ICCL	0-FFFFh	Clock low-time divide-down value of 1-65536. The period of the module clock is multiplied by (ICCL + d) to produce the low-time duration of the I2C serial on the I2Cx_SCL pin.

21.3.4.2 I2C Clock High-Time Divider Register (ICCLKH)

For each I2C serial clock cycle, ICCH in the I2C clock high-time divider register (ICCLKH) determines the amount of time the signal is high. ICCLKH must be configured while the I2C is still in reset (IRS = 0 in ICMDR). ICCLKH is shown in Figure 21-17 and described in Table 21-9.

Figure 21-17. I2C Clock High-Time Divider Register (ICCLKH)



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

Table 21-9. I2C Clock High-Time Divider Register (ICCLKH) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
15-0	ICCH	0-FFFFh	Clock high-time divide-down value of 1-65536. The period of the module clock is multiplied by (ICCH + d) to produce the high-time duration of the I2C serial on the I2Cx_SCL pin.

Copyright © 2013–2016, Texas Instruments Incorporated

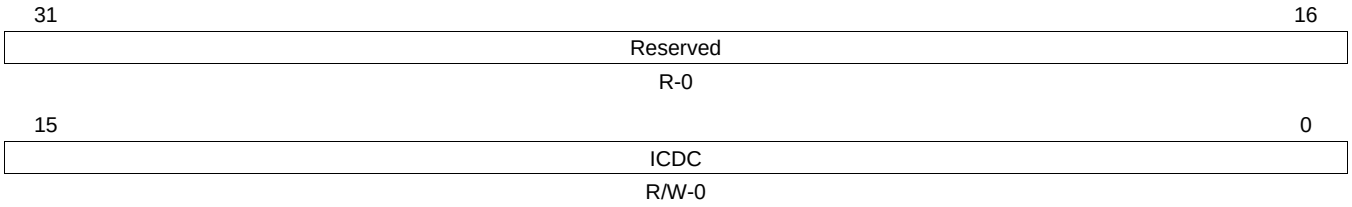
21.3.5 I2C Data Count Register (ICCNT)

The I2C data count register (ICCNT) is used to indicate how many data words to transfer when the I2C is configured as a master-transmitter-receiver (MST = 1 and TRX = 1/0 in ICMDR) and the repeat mode is off (RM = 0 in ICMDR). In the repeat mode (RM = 1), ICCNT is not used.

The value written to ICCNT is copied to an internal data counter. The internal data counter is decremented by 1 for each data word transferred (ICCNT remains unchanged). If a STOP condition is requested (STP = 1 in ICMDR), the I2C terminates the transfer with a STOP condition when the countdown is complete (that is, when the last data word has been transferred).

ICCNT is shown in Figure 21-18 and described in Table 21-10.

Figure 21-18. I2C Data Count Register (ICCNT)



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

Table 21-10. I2C Data Count Register (ICCNT) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
15-0	ICDC	0-FFFFh	Data count value. When RM = 0 in ICMDR, ICDC indicates the number of data words to transfer in the nonrepeat mode. When RM = 1 in ICMDR, the value in ICCNT is a don't care. If STP = 1 in ICMDR, a STOP condition is generated when the internal data counter counts down to 0.
		0	The start value loaded to the internal data counter is 65536.
		1h-FFFFh	The start value loaded to internal data counter is 1-65535.

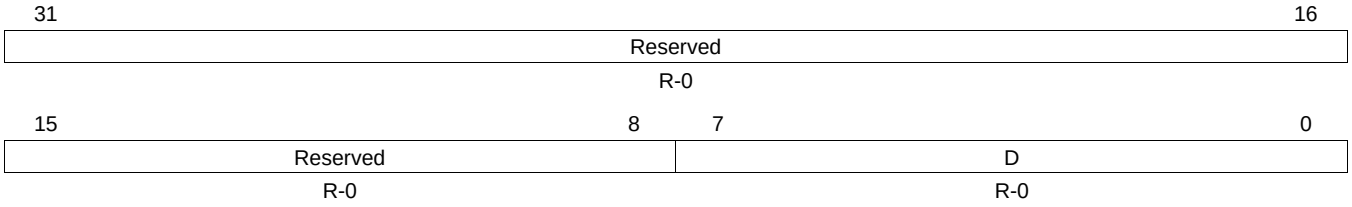
Copyright © 2013–2016, Texas Instruments Incorporated

21.3.6 I2C Data Receive Register (ICDRR)

The I2C data receive register (ICDRR) is used to read the receive data. The ICDRR can receive a data value of up to 8 bits; data values with fewer than 8 bits are right-aligned in the D bits and the remaining D bits are undefined. The number of data bits is selected by the bit count bits (BC) of ICMDR. The I2C receive shift register (ICRSR) shifts in the received data from the I2Cx_SDA pin. Once data is complete, the I2C copies the contents of ICRSR into ICDRR. The CPU and the EDMA controller cannot access ICRSR.

ICDRR is shown in Figure 21-19 and described in Table 21-11.

Figure 21-19. I2C Data Receive Register (ICDRR)



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

Table 21-11. I2C Data Receive Register (ICDRR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
7-0	D	0-FFh	Receive data.

Copyright © 2013–2016, Texas Instruments Incorporated

21.3.7 I2C Slave Address Register (ICSAR)

The I2C slave address register (ICSAR) contains a 7-bit or 10-bit slave address. When the I2C is not using the free data format (FDF = 0 in ICMDR), 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 ICMDR), only bits 6-0 of ICSAR are used; bits 9-7 are ignored.

ICSAR is shown in Figure 21-20 and described in Table 21-12.

Figure 21-20. I2C Slave Address Register (ICSAR)

31	16
Reserved	
R-0	

15	10	9	0
Reserved		SADDR	
R-0		R/W-3FFh	

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

Table 21-12. I2C Slave Address Register (ICSAR) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
9-0	SADDR	0-3FFh	Slave address. Provides the slave address of the I2C. In 7-bit addressing mode (XA = 0 in ICMDR): bits 6-0 provide the 7-bit slave address that the I2C transmits when it is in the master-transmitter mode. Bits 9-7 are ignored. In 10-bit addressing mode (XA = 1 in ICMDR): Bits 9-0 provide the 10-bit slave address that the I2C transmits when it is in the master-transmitter mode.

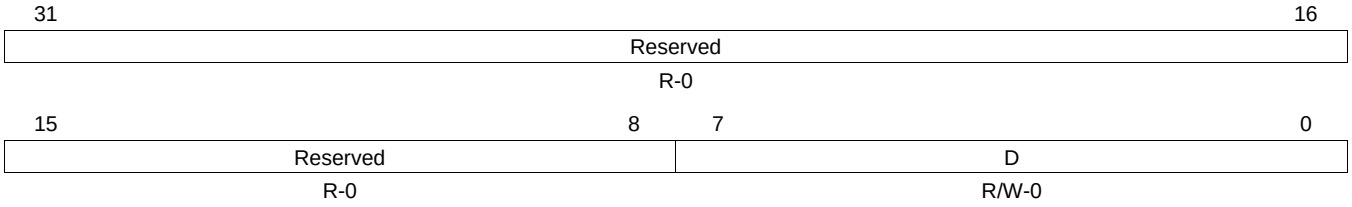
Copyright © 2013–2016, Texas Instruments Incorporated

21.3.8 I2C Data Transmit Register (ICDXR)

The CPU or EDMA writes transmit data to the I2C data transmit register (ICDXR). The ICDXR can accept a data value of up to 8 bits. When writing a data value with fewer than 8 bits, the written data must be right-aligned in the D bits. The number of data bits is selected by the bit count bits (BC) of ICMDR. Once data is written to ICDXR, the I2C copies the contents of ICDXR into the I2C transmit shift register (ICXSR). The ICXSR shifts out the transmit data from the I2Cx_SDA pin. The CPU and the EDMA controller cannot access ICXSR.

ICDXR is shown in Figure 21-21 and described in Table 21-13.

Figure 21-21. I2C Data Transmit Register (ICDXR)



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

Table 21-13. I2C Data Transmit Register (ICDXR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
7-0	D	0-FFh	Transmit data.

Copyright © 2013–2016, Texas Instruments Incorporated

21.3.9 I2C Mode Register (ICMDR)

The I2C mode register (ICMDR) contains the control bits of the I2C. ICMDR is shown in shown in Figure 21-22 and described in Table 21-14.

Figure 21-22. I2C Mode Register (ICMDR)

31							16
Reserved							
R-0							
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-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 21-14. I2C Mode Register (ICMDR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
15	NACKMOD		No-acknowledge (NACK) mode bit (only applicable when the I2C is a receiver).
		0	In slave-receiver mode: The I2C sends an acknowledge (ACK) bit to the transmitter during the each acknowledge cycle on the bus. The I2C only sends a no-acknowledge (NACK) bit if you set the NACKMOD bit. In master-receiver mode: The I2C sends an ACK bit during each acknowledge cycle until the internal data counter counts down to 0. When the counter reaches 0, the I2C 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 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. 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 emulation mode bit is used to determine the state of the I2C when a breakpoint is encountered in the high-level language debugger.
		0	When I2C is master: If I2Cx_SCL is low when the breakpoint occurs, the I2C stops immediately and keeps driving I2Cx_SCL low, whether the I2C is the transmitter or the receiver. If I2Cx_SCL is high, the I2C waits until I2Cx_SCL becomes low and then stops. When I2C is slave: A breakpoint forces the I2C to stop when the current transmission/reception is complete.
		1	The I2C runs free; that is, it continues to operate when a breakpoint occurs.
13	STT		START condition bit (only applicable when the I2C is a master). The RM, STT, and STP bits determine when the I2C starts and stops data transmissions (see Table 21-15). Note that the STT and STP bits can be used to terminate the repeat mode.
		0	In master mode, STT is automatically cleared after the START condition has been generated. In slave mode, if STT is 0, the I2C does not monitor the bus for commands from a master. As a result, the I2C performs no data transfers.
		1	In master mode, setting STT to 1 causes the I2C to generate a START condition on the I2C-bus. In slave mode, if STT is 1, the I2C monitors the bus and transmits/receives data in response to commands from a master.
12	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
11	STP		STOP condition bit (only applicable when the I2C is a master). The RM, STT, and STP bits determine when the I2C starts and stops data transmissions (see Table 21-15). Note that the STT and STP bits can be used to terminate the repeat mode.
		0	STP is automatically cleared after the STOP condition has been generated.
		1	STP has been set to generate a STOP condition when the internal data counter of the I2C counts down to 0.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 21-14. I2C Mode Register (ICMDR) Field Descriptions (continued)

Bit	Field	Value	Description
10	MST		Master mode bit. MST determines whether the I2C 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. See Table 21-16.
		0	Slave mode. The I2C is a slave and receives the serial clock from the master.
		1	Master mode. The I2C is a master and generates the serial clock on the I2Cx_SCL pin.
9	TRX		Transmitter mode bit. When relevant, TRX selects whether the I2C is in the transmitter mode or the receiver mode. Table 21-16 summarizes when TRX is used and when it is a don't care.
		0	Receiver mode. The I2C is a receiver and receives data on the I2Cx_SDA pin.
		1	Transmitter mode. The I2C is a transmitter and transmits data on the I2Cx_SDA pin.
8	XA		Expanded address enable bit.
		0	7-bit addressing mode (normal address mode). The I2C transmits 7-bit slave addresses (from bits 6-0 of ICSAR), and its own slave address has 7 bits (bits 6-0 of ICOAR).
		1	10-bit addressing mode (expanded address mode). The I2C transmits 10-bit slave addresses (from bits 9-0 of ICSAR), and its own slave address has 10 bits (bits 9-0 of ICOAR).
7	RM		Repeat mode bit (only applicable when the I2C is a master). The RM, STT, and STP bits determine when the I2C starts and stops data transmissions (see Table 21-15). If the I2C is configured in slave mode, the RM bit is don't care.
		0	Nonrepeat mode. The value in the data count register (ICCNT) determines how many data words are received/transmitted by the I2C.
		1	Repeat mode. Data words are continuously received/transmitted by the I2C until the STP bit is manually set to 1, regardless of the value in ICCNT.
6	DLB		Digital loopback mode bit (only applicable when the I2C is a master-transmitter). This bit disables or enables the digital loopback mode of the I2C. The effects of this bit are shown in Figure 21-23. Note that DLB mode in the free data format mode (DLB = 1 and FDF = 1) is not supported.
		0	Digital loopback mode is disabled.
		1	Digital loopback mode is enabled. In this mode, the MST bit must be set to 1 and data transmitted out of ICDXR is received in ICDRR after n clock cycles by an internal path, where: n = ((I2C input clock frequency/prescaled module clock frequency) × 8) The transmit clock is also the receive clock. The address transmitted on the I2Cx_SDA pin is the address in ICOAR.
5	IRS		I2C reset bit. Note that if IRS is reset during a transfer, it can cause the I2C bus to hang (I2Cx_SDA and I2Cx_SCL are in a high-impedance state).
		0	The I2C is in reset/disabled. When this bit is cleared to 0, all status bits (in ICSTR) are set to their default values.
		1	The I2C is enabled.
4	STB		START byte mode bit (only applicable when the I2C is a master). As described in version 2.1 of the Philips 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 is a slave, the I2C ignores a START byte from a master, regardless of the value of the STB bit.
		0	The I2C is not in the START byte mode.
		1	The I2C is in the START byte mode. When you set the START condition bit (STT), the I2C 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 The I2C sends the slave address that is in ICSAR.
3	FDF		Free data format mode bit. Note that DLB mode in the free data format mode (DLB = 1 and FDF = 1) is not supported. See Table 21-16.
		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.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 21-14. I2C Mode Register (ICMDR) Field Descriptions (continued)

Bit	Field	Value	Description
2-0	BC	0-7h	Bit count bits. BC defines the number of bits (1 to 8) in the next data word that is to be received or transmitted by the I2C. The number of bits selected with BC must match the data size of the other device. Note that when BC = 0, a data word has 8 bits. If the bit count is less than 8, receive data is right aligned in the D bits of ICDRR and the remaining D bits are undefined. Also, transmit data written to ICDXR must be right aligned.
		0	8 bits per data word
		1h	1 bit per data word
		2h	2 bits per data word
		3h	3 bits per data word
		4h	4 bits per data word
		5h	5 bits per data word
		6h	6 bits per data word
		7h	7 bits per data word

Table 21-15. Master-Transmitter/Receiver Bus Activity Defined by RM, STT, and STP Bits

ICMDR Bit				
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 words (*n* = value in ICCNT)
0	1	1	S-A-D..(*n*)..D-P	START condition, slave address, *n* data words, STOP condition (*n* = value in ICCNT)
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

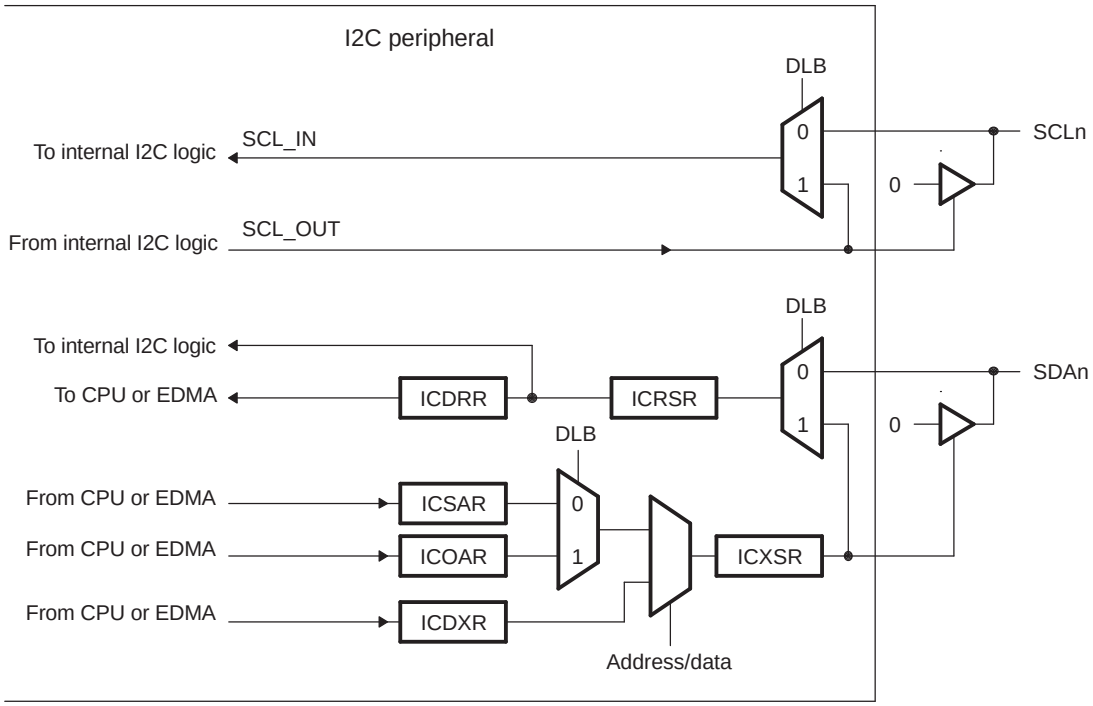
(1) A = Address; D = Data word; P = STOP condition; S = START condition

Table 21-16. How the MST and FDF Bits Affect the Role of TRX Bit

ICMDR Bit			
MST	FDF	I2C State	Function of TRX Bit
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 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 transmitter and receiver be fixed. TRX identifies the role of the I2C: TRX = 0: The I2C is a receiver. TRX = 1: The I2C is a transmitter.
1	0	In master mode but not free data format mode	TRX identifies the role of the I2C: TRX = 0: The I2C is a receiver. TRX = 1: The I2C is a transmitter.
1	1	In master mode and free data format mode	The free data format mode requires that the transmitter and receiver be fixed. TRX identifies the role of the I2C: TRX = 0: The I2C is a receiver. TRX = 1: The I2C is a transmitter.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 21-23. Block Diagram Showing the Effects of the Digital Loopback Mode (DLB) Bit



Copyright © 2013–2016, Texas Instruments Incorporated

21.3.10 I2C Interrupt Vector Register (ICIVR)

The I2C interrupt vector register (ICIVR) is used by the CPU to determine which event generated the I2C interrupt. Reading ICIVR clears the interrupt flag; if other interrupts are pending, a new interrupt is generated. If there are more than one interrupt flag, reading ICIVR clears the highest priority interrupt flag. Note that you must read (clear) ICIVR before doing another start; otherwise, ICIVR could contain an incorrect (old interrupt flags) value.

ICIVR is shown in Figure 21-24 and described in Table 21-17.

Figure 21-24. I2C Interrupt Vector Register (ICIVR)

31	16
Reserved	
R-0	

15	2	0
Reserved	INTCODE	
R-0	R-0	

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

Table 21-17. I2C Interrupt Vector Register (ICIVR) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
2-0	INTCODE	0-7h	Interrupt code bits. The binary code in INTCODE indicates which event generated an I2C interrupt.
		0	None
		1h	Arbitration-lost interrupt (AL). Highest priority if multiple I2C interrupts are pending.
		2h	No-acknowledgment interrupt (NACK)
		3h	Register-access-ready interrupt (ARDY)
		4h	Receive-data-ready interrupt (ICRRDY)
		5h	Transmit-data-ready interrupt (ICXRDY)
		6h	Stop condition detected interrupt (SCD)
		7h	Address-as-slave interrupt (AAS). Lowest priority if multiple I2C interrupts are pending.

Copyright © 2013–2016, Texas Instruments Incorporated

21.3.11 I2C Extended Mode Register (ICEMDR)

The I2C extended mode register (ICEMDR) is used to indicate which condition generates a transmit data ready interrupt. ICEMDR is shown in Figure 21-25 and described in Table 21-18.

Figure 21-25. I2C Extended Mode Register (ICEMDR)

31	16
Reserved	
R-0	

15	1	0
Reserved	IGNACK	BCM
R-0	R/W-0	R/W-1

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

Table 21-18. I2C Extended Mode Register (ICEMDR) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
1	IGNACK		Ignore NACK mode.
		0	Master transmitter operates normally, that is, it discontinues the data transfer and sets the ARDY and NACK bits in ICSTR when receiving a NACK from the slave.
		1	Master transmitter ignores a NACK from the slave.
0	BCM		Backward compatibility mode bit. Determines which condition generates a transmit data ready interrupt.
			The BCM bit only has an effect when the I2C is operating as a slave-transmitter.
		0	The transmit data ready interrupt is generated when the master requests more data by sending an acknowledge signal after the transmission of the last data.
		1	The transmit data ready interrupt is generated when the data in ICDXR is copied to ICXSR.

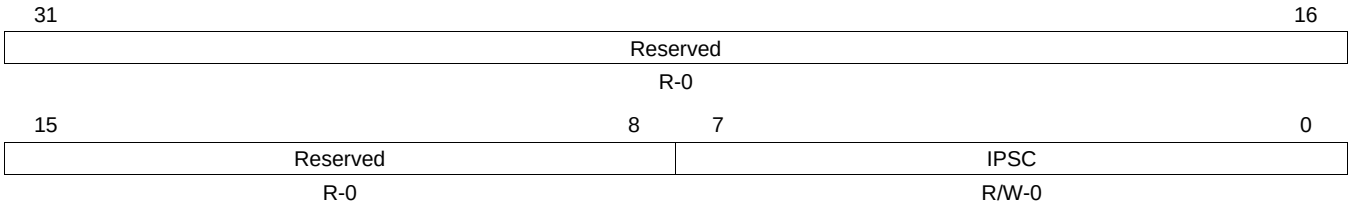
Copyright © 2013–2016, Texas Instruments Incorporated

21.3.12 I2C Prescaler Register (ICPSC)

The I2C prescaler register (ICPSC) is used for dividing down the I2C input clock to obtain the desired prescaled module clock for the operation of the I2C. The IPSC bits must be initialized while the I2C is in reset (IRS = 0 in ICMDR). The prescaled frequency takes effect only when the IRS bit is changed to 1. Changing the IPSC value while IRS = 1 has no effect.

ICPSC is shown in Figure 21-26 and described in Table 21-19.

Figure 21-26. I2C Prescaler Register (ICPSC)



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

Table 21-19. I2C Prescaler Register (ICPSC) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
7-0	IPSC	0-FFh	I2C prescaler divide-down value. IPSC determines how much the I2C input clock is divided to create the I2C prescaled module clock: I2C clock frequency = I2C input clock frequency/(IPSC + 1) **Note:** IPSC must be initialized while the I2C is in reset (IRS = 0 in ICMDR).

Copyright © 2013–2016, Texas Instruments Incorporated

21.3.13 I2C Revision Identification Register (REVID1)

The I2C revision identification register (REVID1) contains identification data for the peripheral. REVID1 is shown in Figure 21-27 and described in Table 21-20.

Figure 21-27. I2C Revision Identification Register 1 (REVID1)

31	0
REVID1	
R-4415h	

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

Table 21-20. I2C Revision Identification Register 1 (REVID1) Field Descriptions

Bit	Field	Value	Description
31-0	REVID1	4415h	Peripheral Identification Number

21.3.14 I2C Revision Identification Register (REVID2)

The I2C revision identification register (REVID2) contains identification data for the peripheral. REVID2 is shown in Figure 21-28 and described in Table 21-21.

Figure 21-28. I2C Revision Identification Register 2 (REVID2)

31	0
REVID2	
R-6h	

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

Table 21-21. I2C Revision Identification Register 2 (REVID2) Field Descriptions

Bit	Field	Value	Description
31-0	REVID2	6h	Peripheral Identification Number

Copyright © 2013–2016, Texas Instruments Incorporated

21.3.15 I2C DMA Control Register (ICDMAC)

The I2C DMA control register (ICDMAC) is used to control the transmit DMA event and receive DMA event pin to the system . ICDMAC is shown in Figure 21-29 and described in Table 21-22.

Figure 21-29. I2C DMA Control Register (ICDMAC)

31	16
Reserved	
R-0	

15	1	0
Reserved	TXDMAEN	RXDMAEN
R-0	R/W-1	R/W-1

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

Table 21-22. I2C DMA Control Register (ICDMAC) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
1	TXDMAEN		Transmit DMA enable. This bit controls the transmit DMA event pin to the system. Always set this bit to 1.
		0	DMA transmit event is disabled.
		1	DMA transmit event is enabled.
0	RXDMAEN		Receive DMA enable . This bit controls the receive DMA event pin to the system. Always set this bit to 1.
		0	DMA receive event is disabled.
		1	DMA receive event is enabled.

Copyright © 2013–2016, Texas Instruments Incorporated

21.3.16 I2C Pin Function Register (ICPFUNC)

The I2C pin function register (ICPFUNC) is used to configure the external I2C pins (I2Cx_SDA and I2Cx_SCL) as a I2C peripheral pin or a GPIO pin. ICPFUNC is shown in Figure 21-30 and described in Table 21-23.

Figure 21-30. I2C Pin Function Register (ICPFUNC)

31	16
Reserved	
R-0	

15	1	0
Reserved		PFUNC0
R-0		R/W-0

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

Table 21-23. I2C Pin Function Register (ICPFUNC) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
0	PFUNC0		Controls the function of the I2Cx_SCL and I2Cx_SDA pins.
		0	Pins function as I2Cx_SCL and I2Cx_SDA.
		1	Pins function as GPIO.
			Note: No hardware protection is required to disable the I2C function when the PFUNC0 bit and the IRS bit in the I2C mode register (ICMDR) are both set to 1. When PFUNC0 = 1 (GPIO mode), the sub-module that controls the I2C function receives the value 1 for I2Cx_SCL and I2Cx_SDA. The IRS bit can be set to 1 regardless of PFUNC0, and the I2C function works whenever the IRS bit is 1. You are expected to hold I2C in reset via the IRS bit when changing to/from GPIO mode via the PFUNC0 bit.

Copyright © 2013–2016, Texas Instruments Incorporated

21.3.17 I2C Pin Direction Register (ICPDIR)

The I2C pin direction register (ICPDIR) is used to configure each GPIO pin as either an input or an output. ICPDIR is shown in Figure 21-31 and described in Table 21-24.

Figure 21-31. I2C Pin Direction Register (ICPDIR)

31															16
Reserved															
R-0															

15													2	1	0
Reserved														PDIR1	PDIR0
R-0														R/W-0	R/W-0

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

Table 21-24. I2C Pin Direction Register (ICPDIR) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
1	PDIR1		Controls the direction of the I2Cx_SDA pin when configured as GPIO.
		0	I2Cx_SDA pin functions as input.
		1	I2Cx_SDA pin functions as output.
0	PDIR0		Controls the direction of the I2Cx_SCL pin when configured as GPIO.
		0	I2Cx_SCL pin functions as input.
		1	I2Cx_SCL pin functions as output.

Copyright © 2013–2016, Texas Instruments Incorporated

21.3.18 I2C Pin Data In Register (ICPDIN)

The I2C pin data in register (ICPDIN) holds the I/O state of each of the I2C pins (I2Cx_SDA and I2Cx_SCL); and should return the value from the pin's input buffer (with appropriate synchronization/DFT considerations). However, this register allows the actual value of the pin to be read regardless of the state of PFUNC or PDIR bits . ICPDIN is shown in Figure 21-32 and described in Table 21-25.

Figure 21-32. I2C Pin Data In Register (ICPDIN)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		PDIN1	PDIN0
R-0		R/W-0	R/W-0

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

Table 21-25. I2C Pin Data In Register (ICPDIN) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
1	PDIN1		Indicates the logic level present on the I2Cx_SDA pin.
			During reads:
		0	Logic-low present at I2Cx_SDA pin, regardless of PFUNC bit setting.
		1	Logic-high present at I2Cx_SDA pin, regardless of PFUNC bit setting.
			During writes: Writes have no effect.
0	PDIN0		Indicates the logic level present on the I2Cx_SCL pin.
			During reads:
		0	Logic-low present at I2Cx_SCL pin, regardless of PFUNC bit setting.
		1	Logic-high present at I2Cx_SCL pin, regardless of PFUNC bit setting.
			During writes: Writes have no effect.

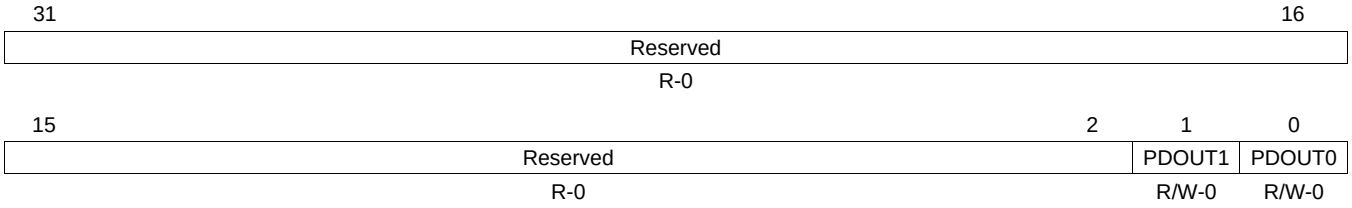
Copyright © 2013–2016, Texas Instruments Incorporated

21.3.19 I2C Pin Data Out Register (ICPDOUT)

The I2C pin data out register (ICPDOUT) has one bit for each of the GPIO pins. This bit holds a value for data out at all times, and may be read back at all times. The value held by this register is not affected by writing to the PDIR and PFUNC bits. However, the data value in this register is driven out onto the GPIO pin only if the PFUNC0 bit in ICPFUNC is set to 1 (I2Cx_SDA and I2Cx_SCL function as GPIO) and also the corresponding bit in ICPDIR is set to 1 (output).

ICPDOUT is shown in Figure 21-33 and described in Table 21-26.

Figure 21-33. I2C Pin Data Out Register (ICPDOUT)



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

Table 21-26. I2C Pin Data Out Register (ICPDOUT) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
1	PDOUT1		Controls the level driven on the I2Cx_SDA pin when configured as GPIO output. Note: If I2Cx_SDA is connected to an open-drain buffer at the chip level, the I2C cannot drive I2Cx_SDA to high. **During reads:** Reads return register values, not GPIO pin levels.
			During writes:
		0	I2Cx_SDA pin is driven low.
		1	I2Cx_SDA pin is driven high.
0	PDOUT0		Controls the level driven on the I2Cx_SCL pin when configured as GPIO output. Note: If I2Cx_SCL is connected to an open-drain buffer at the chip level, the I2C cannot drive I2Cx_SCL to high. **During reads:** Reads return register values, not GPIO pin levels.
			During writes:
		0	I2Cx_SCL pin is driven low.
		1	I2Cx_SCL pin is driven high.

Copyright © 2013–2016, Texas Instruments Incorporated

21.3.20 I2C Pin Data Set Register (ICPDSET)

The I2C pin data set register (ICPDSET) is an alias of the I2C pin data out register (ICPDOUT). Writing a 1 to a bit in ICPDSET sets the corresponding bit in ICPDOUT to a 1, while writing a 0 keeps the bit unchanged. ICPDSET is shown in Figure 21-34 and described in Table 21-27.

Figure 21-34. I2C Pin Data Set Register (ICPDSET)

31															16
Reserved															
R-0															
15													2	1	0
Reserved														PDSET1	PDSET0
R-0														R/W-0	R/W-0

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

Table 21-27. I2C Pin Data Set Register (ICPDSET) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
1	PDSET1		Used to set the PDOUT1 bit in the I2C pin data out register (ICPDOUT) that corresponds to the I2Cx_SDA GPIO pin. **During reads:** Reads return indeterminate values.
			During writes:
		0	No effect
		1	PDOUT1 bit is set to logic high.
0	PDSET0		Used to set the PDOUT0 bit in the I2C pin data out register (ICPDOUT) that corresponds to the I2Cx_SCL GPIO pin. **During reads:** Reads return indeterminate values.
			During writes:
		0	No effect
		1	PDOUT0 bit is set to logic high.

Copyright © 2013–2016, Texas Instruments Incorporated

21.3.21 I2C Pin Data Clear Register (ICPDCLR)

The I2C pin data clear register (ICPDCLR) is an alias of the I2C pin data out register (ICPDOUT). Writing a 1 to a bit in ICPDCLR clears the corresponding bit in ICPDOUT to a 0, while writing a 0 keeps the bit unchanged. ICPDCLR is shown in Figure 21-35 and described in Table 21-28.

Figure 21-35. I2C Pin Data Clear Register (ICPDCLR)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		PDCLR1	PDCLR0
R-0		R/W-0	R/W-0

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

Table 21-28. I2C Pin Data Clear Register (ICPDCLR) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	These reserved bit locations are always read as zeros. A value written to this field has no effect.
1	PDCLR1		Used to clear the PDOUT1 bit in the I2C pin data out register (ICPDOUT) that corresponds to the I2Cx_SDA GPIO pin.
			During reads: Reads return indeterminate values.
			During writes:
		0	No effect
		1	PDOUT1 bit is cleared to logic low.
0	PDCLR0		Used to clear the PDOUT0 bit in the I2C pin data out register (ICPDOUT) that corresponds to the I2Cx_SCL GPIO pin.
			During reads: Reads return indeterminate values.
			During writes:
		0	No effect
		1	PDOUT0 bit is cleared to logic low.

Copyright © 2013–2016, Texas Instruments Incorporated

Liquid Crystal Display Controller (LCDC)

The liquid crystal display controller (LCDC) is capable of supporting an asynchronous (memory-mapped) LCD interface and a synchronous (raster-type) LCD interface. This chapter describes the LCDC.

Topic **Page**

22.1 Introduction **891**
22.2 Architecture **892**
22.3 Registers **908**

Copyright © 2013–2016, Texas Instruments Incorporated

22.1 Introduction

22.1.1 Purpose of the Peripheral

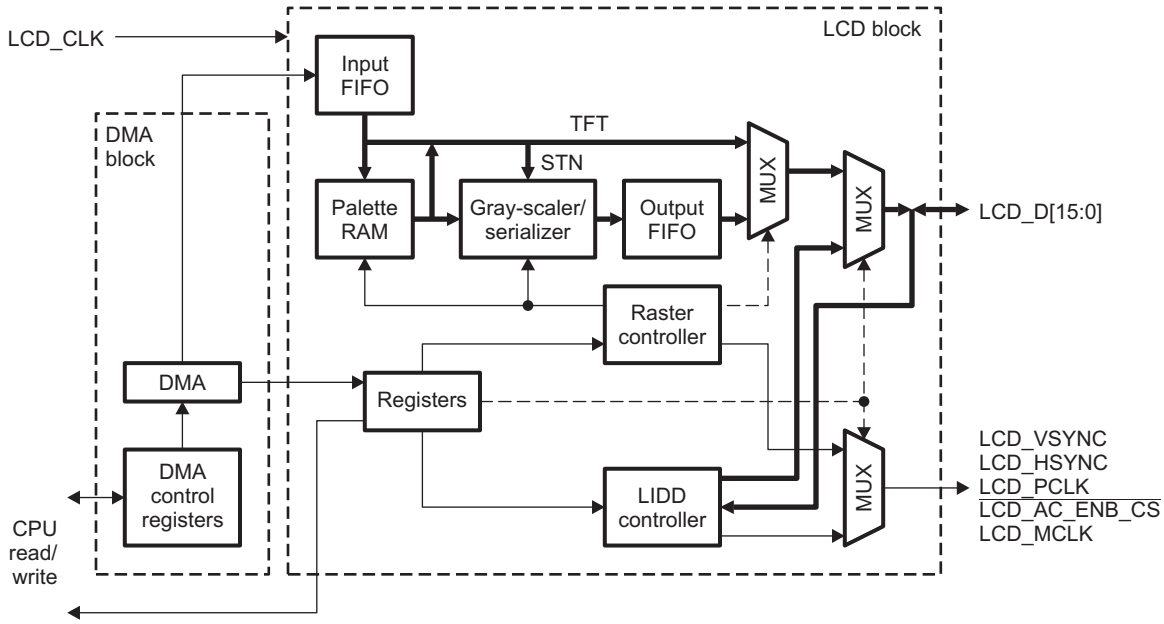
The LCD controller consists of two independent controllers, the Raster Controller and the LCD Interface Display Driver (LIDD) controller. Each controller operates independently from the other and only one of them is active at any given time.

- The Raster Controller handles the synchronous LCD interface. It provides timing and data for constant graphics refresh to a passive display. It supports a wide variety of monochrome and full-color display types and sizes by use of programmable timing controls, a built-in palette, and a gray-scale/serializer. Graphics data is processed and stored in frame buffers. A frame buffer is a contiguous memory block in the system. A built-in DMA engine supplies the graphics data to the Raster engine which, in turn, outputs to the external LCD device.
- The LIDD Controller supports the asynchronous LCD interface. It provides full-timing programmability of control signals and output data.

Figure 22-1 shows the LCD controller details. The raster and LIDD Controllers are responsible for generating the correct external timing. The DMA engine provides a constant flow of data from the frame buffer(s) to the external LCD panel via the Raster and LIDD Controllers. In addition, CPU access is provided to read and write registers.

The solid, thick lines in Figure 22-1 indicate the data path. The Raster Controller's data path is fairly complicated, for a thorough description of the Raster Controller data path, see Section 22.2.5.

Figure 22-1. LCD Controller



Copyright © 2013–2016, Texas Instruments Incorporated

22.1.2 Features

See your device-specific data manual to check the features supported by the LCD controller.

22.1.3 Terminology

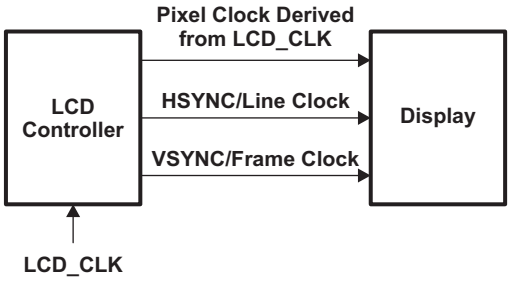
Term	Meaning
Passive (STN) device	Refers to the Super-Twisted Nematic (STN) display device.
Active (TFT) device	Refers to the Thin-Film Transistor (TFT) display device.
BPP	Bits per pixel; that is, the number of bits used for each pixel. In some documentation, this is also referred to as color depth.
RGB	Red, Green, Blue

22.2 Architecture

22.2.1 Clocking

This section details the various clocks and signals. Figure 22-2 shows input and output LCD controller clocks.

Figure 22-2. Input and Output Clocks



22.2.1.1 Pixel Clock

The pixel clock (LCD_PCLK) frequency is derived from LCD_CLK, the reference clock to this LCD module (see Figure 22-2). The pixel clock is used by the LCD display to clock the pixel data into the line shift register.

$$\text{LCD_PCLK} = \frac{\text{LCD_CLK}}{\text{CLKDIV}}$$

where CLKDIV is a field in the LCD_CTRL register and should not be 0 or 1.

- **Passive (STN) mode.** LCD_PCLK only transitions when valid data is available for output. It does not transition when the horizontal clock is asserted or during wait state insertion.
- **Active (TFT) mode.** LCD_PCLK continuously toggles as long as the Raster Controller is enabled.

Copyright © 2013–2016, Texas Instruments Incorporated

22.2.1.2 Horizontal Clock (LCD_HSYNC)

LCD_HSYNC toggles after all pixels in a horizontal line have been transmitted to the LCD and a programmable number of pixel clock wait states has elapsed both at the beginning and end of each line.

The RASTER_TIMING_0 register fully defines the behavior of this signal.

LCD_HSYNC can be programmed to be synchronized with the rising or falling edge of LCD_PCLK. The configuration field is bits 24 and 25 in the RASTER_TIMING_2 register.

- **Active (TFT) mode.** The horizontal clock or the line clock is also used by TFT displays as the horizontal synchronization signal (LCD_HSYNC).

  The timings of the horizontal clock(line clock) pins are programmable to support:
- Delay insertion both at the beginning and end of each line
- Line clock polarity.
- Line clock pulse width, driven on rising or falling edge of pixel clock.

22.2.1.3 Vertical Clock (LCD_VSYNC)

LCD_VSYNC toggles after all lines in a frame have been transmitted to the LCD and a programmable number of line clock cycles has elapsed both at the beginning and end of each frame.

The RASTER_TIMING_1 register fully defines the behavior of this signal.

LCD_VSYNC can be programmed to be synchronized with the rising or falling edge of LCD_PCLK. The configuration field is bits 24 and 25 in the RASTER_TIMING_2 register.

- **Passive (STN) mode.** The vertical clock; that is, the frame clock, toggles during the first line of the screen.
- **Active (TFT) mode.** The vertical clock, that is, the frame clock, is also used by TFT displays as the vertical synchronization signal (LCD_VSYNC).

  The timings of the vertical clock pins are programmable to support:
- Delay insertion both at the beginning and end of each frame
- Frame clock polarity

22.2.1.4 $\overline{\text{LCD_AC_ENB_CS}}$

- **Passive (STN) mode.** To prevent a dc charge within the screen pixels, the power and ground supplies of the display are periodically switched. The Raster Controller signals the LCD to switch the polarity by toggling this pin ($\overline{\text{LCD_AC_ENB_CS}}$).
- **Active (TFT) mode.** This signal acts as an output enable (OE) signal. It is used to signal the external LCD that the data is valid on the data bus (LCD_D[15:0]).

Copyright © 2013–2016, Texas Instruments Incorporated

22.2.2 LCD External I/O Signals

Table 22-1 shows the details of the LCD controller external signals.

Table 22-1. LCD External I/O Signals

Signal	Type	Description
LCD_VSYNC	OUT	Raster controller: Frame clock the LCD uses to signal the start of a new frame of pixels. Also used by TFT displays as the vertical synchronization signal. LIDD character: Register select (RS) LIDD graphics: Register select (RS), address bit 0 (A0), or command/data select (C/D)
LCD_HSYNC	OUT	Raster controller: Line clock the LCD uses to signal the end of a line of pixels that transfers line data from the shift register to the screen and to increment the line pointer(s). Also used by TFT displays as the horizontal synchronization signal. LIDD character: read or write enable LIDD graphics: • 6800 mode = read or write enable • 8080 mode = write strobe
LCD_PCLK	OUT	Raster controller: Pixel clock the LCD uses to clock the pixel data into the line shift register. In passive mode, the pixel clock transitions only when valid data is available on the data lines. In active mode, the pixel clock transitions continuously, and the ac-bias pin is used as an output enable to signal when data is available on the LCD pin. LIDD character: not used. LIDD graphics: • 6800 mode = enable strobe • 8080 mode = read strobe
$\overline{\text{LCD_AC_ENB_CS}}$	OUT	Raster controller: ac-bias used to signal the LCD to switch the polarity of the power supplies to the row and column axis of the screen to counteract DC offset. Used in TFT mode as the output enable to signal when data is latched from the data pins using the pixel clock. LIDD character: Primary enable strobe LIDD graphics: Chip select 0 ($\overline{\text{CS0}}$)
LCD_MCLK	OUT	Raster controller: not used. LIDD character: Secondary enable strobe LIDD graphics: Chip select 1 ($\overline{\text{CS1}}$)
LCD_D[15:0]	Raster: OUT	LCD data bus, providing a 4-, 8-, or 16-bit data path.
	LIDD: OUT/IN	Raster controller: For monochrome displays, each signal represents a pixel; for passive color displays, groupings of three signals represent one pixel (red, green, and blue). LCD_D[3:0] is used for monochrome displays of 2, 4, and 8 BPP; LCD_D[7:0] is used for color STN displays and LCD_D[15:0] is used for active (TFT) mode. LIDD character: Read and write the command and data registers. LIDD graphics: Read and write the command and data registers.

Copyright © 2013–2016, Texas Instruments Incorporated

22.2.3 DMA Engine

The DMA engine provides the capability to output graphics data to constantly refresh LCDs, without burdening the CPU, via interrupts or a firmware timer. It operates on one or two frame buffers, which are set up during initialization. Using two frame buffers (ping-pong buffers) enables the simultaneous operation of outputting the current video frame to the external display and updating the next video frame. The ping-pong buffering approach is preferred in most applications.

When the Raster Controller is used, the DMA engine reads data from a frame buffer and writes it to the input FIFO (as shown in Figure 22-1). The Raster Controller requests data from the FIFO for frame refresh; as a result, the DMA's job is to ensure that the FIFO is always kept full.

When the LIDD Controller is used, the DMA engine accesses the LIDD Controller's address and/or data registers.

To program DMA engine, configure the following registers, as shown in Table 22-2.

Table 22-2. Register Configuration for DMA Engine Programming

Register	Configuration
LCDDMA_CTRL	Configure DMA data format
LCDDMA_FB0_BASE LCDDMA_FB0_CEILING	Configure frame buffer 0
LCDDMA_FB1_BASE LCDDMA_FB1_CEILING	Configure frame buffer 1. (If only one frame buffer is used, these two registers will not be used.)

In addition, the LIDD_CTRL register (for LIDD Controller) or the RASTER_CTRL register (for Raster Controller) should also be configured appropriately, along with all the timing registers.

To enable DMA transfers, the LIDD_DMA_EN bit (in the LIDD_CTRL register) or the LCDEN bit (in the RASTER_CTRL register) should be written with 1.

NOTE: If the data left in the frame buffer is smaller than the DMA burst size, the DMA by default transfers 1 word (4 bytes) at a time until the entire frame buffer is transferred. This sometimes causes an input FIFO underflow, which can only be recovered through a Power and Sleep Controller (PSC) reset. Thus, it is recommended that the size of the frame buffer be divisible by the chosen burst size.

22.2.3.1 Interrupts

Interrupts in this LCD module are related to DMA engine operation. Three registers are closely related to this subject:

- The LIDD_CTRL and RASTER_CTRL registers enable or disable each individual interrupt sources.
- The LCD_STAT register collects all the interrupt status information.

22.2.3.1.1 LIDD Mode

When operating in LIDD mode, the DMA engine generates one interrupt signal every time the specified frame buffer has been transferred completely.

- The DONE_INT_EN bit in the LIDD_CTRL register specifies if the interrupt signal is delivered to the system interrupt controller, which in turn may or may not generate an interrupt to CPU.
- The EOF1, EOF0, and DONE bits in the LCD_STAT register reflect the interrupt signal, regardless of being delivered to the system interrupt controller or not.

Copyright © 2013–2016, Texas Instruments Incorporated

22.2.3.1.2 Raster Mode

When operating in Raster mode, the DMA engine can generate the interrupts in the following scenarios:

1. **Output FIFO under-run**. This occurs when the DMA engine cannot keep up with the data rate consumed by the LCD (which is determined by the LCD_PCLK.) This is likely due to a system memory throughput issue or an incorrect LCD_PCLK setting. The FUF bit in LCD_STAT is set when this error occurs. This bit is cleared by disabling the Raster Controller (i.e., clearing the LCDEN bit in RASTER_CTRL).
2. **Frame synchronization lost**. This error happens when the DMA engine attempts to read what it believes to be the first word of the video buffer but it cannot be recognized as such. This could be caused by an invalid frame buffer address or an invalid BPP value (for more details, see Section 22.2.5.2). The SYNC bit in the LCD_STAT register is set when such an error is detected. This field is cleared by disabling the Raster Controller (clearing the LCDEN bit in the RASTER_CTRL register).
3. **Palette loaded**. This interrupt can be generated when the palette is loaded into the memory by the DMA engine. At the same time, the PL bit in the LCD_STAT register is set. In data-only (PLM = 2h) and palette-plus-data (PLM = 00) modes, writing 0 to this bit clears the interrupt. In the palette-only (PLM = 1) mode, this bit is cleared by disabling the Raster Controller (clearing the LCDEN bit in the RASTER_CTRL register).
4. **AC bias transition**. If the ACB_I bit in the RASTER_TIMING_2 register is programmed with a non-zero value, an internal counter will be loaded with this value and starts to decrement each time LCD_AC_ENB_CS (AC-bias signal) switches its state. When the counter reaches zero, the ABC bit in the LCD_STAT register is set, which will deliver an interrupt signal to the system interrupt controller (if the interrupt is enabled.) The counter reloads the value in field ACB_I, but does not start to decrement until the ABC bit is cleared by writing 0 to this bit.
5. **Frame transfer completed**. When one frame of data is transferred completely, the DONE bit in the LCD_STAT register is set. This bit is cleared by disabling the Raster Controller (i.e., clearing the LCDEN bit in the RASTER_CTRL register). Note that the EOF0 and EOF1 bits in the LCD_STAT register will be set accordingly.

Note that the interrupt enable bits are in the RASTER_CTRL register. The corresponding enable bit must be set in order to generate an interrupt to the CPU. However, the LCD_STAT register reflects the interrupt signal regardless of the interrupt enable bits settings.

22.2.3.1.3 Interrupt Handling

See your device-specific data manual for information about the LCD interrupt number to the CPU .The interrupt service routine needs to determine the interrupt source by examining the LCD_STAT register and clearing the interrupt properly.

22.2.4 LIDD Controller

The LIDD Controller is designed to support LCD panels with a memory-mapped interface. The types of displays range from low-end character monochrome LCD panels to high-end TFT smart LCD panels.

LIDD mode (and the use of this logic) is enabled by clearing the MODESEL bit in the LCD control register (LCD_CTRL).

LIDD Controller operation is summarized as follows:

- During initialization, the LCD LIDD CS0/CS1 configuration registers (LIDD_CS0_CONF and LIDD_CS1_CONF) are configured to match the requirements of the LCD panel being used.
- During normal operation, the CPU writes display data to the LCD data registers (LIDD_CS0_DATA and LIDD_CS1_DATA). The LIDD interface converts the CPU write into the proper signal transition sequence for the display, as programmed earlier. Note that the first CPU write should send the beginning address of the update to the LCD panel and the subsequent writes update data at display locations starting from the first address and continuing sequentially. Note that DMA may be used instead of CPU.
- The LIDD Controller is also capable of reading back status or data from the LCD panel, if the latter has this capability. This is set up and activated in a similar manner to the write function described above.

Copyright © 2013–2016, Texas Instruments Incorporated

NOTE: If an LCD panel is not used, this interface can be used to control any MCU-like peripheral.

See your device-specific data manual to check the LIDD features supported by the LCD controller.

Table 22-3 describes how the signals are used to interface external LCD modules, which are configured by the LIDD_CTRL register.

Table 22-3. LIDD I/O Name Map

Display Type	Interface Type	Data Bits	LIDD_CTRL [2:0]	I/O Name	Display I/O Name	Comment
Character Display	HD44780 Type	4	100	LCD_D[7:4]	DATA[7:4]	Data Bus (length defined by Instruction)
				LCD_HSYNC	R/$\overline{W}$	Read/$\overline{Write}$
				LCD_VSYNC	RS	Register Select (RS = 0, command; RS = 1, data)
				$\overline{LCD_AC_ENB_CS}$	E (or E0)	Enable Strobe (first display)
				LCD_MCLK	E1	Enable Strobe (second display optional)
Character Display	HD44780 Type	8	100	LCD_D[7:0]	DATA[7:0]	Data Bus (length defined by Instruction)
				LCD_HSYNC	R/$\overline{W}$	Read/$\overline{Write}$
				LCD_VSYNC	RS	Register Select (RS = 0, command; RS = 1, data)
				$\overline{LCD_AC_ENB_CS}$	E (or E0)	Enable Strobe (first display)
				LCD_MCLK	E1	Enable Strobe (second display optional)
Micro Interface Graphic Display	6800 Family	Up to 16	001	LCD_D[15:0]	DATA[15:0]	Data Bus (16 bits always available)
				LCD_PCLK	E	Enable Clock
				LCD_HSYNC	R/$\overline{W}$	Read/$\overline{Write}$
				LCD_VSYNC	RS	Register Select (RS = 0, command; RS = 1, data)
				$\overline{LCD_AC_ENB_CS}$	$\overline{CS}$ (or $\overline{CS0}$)	Chip Select (first display)
				LCD_MCLK	$\overline{CS1}$	Chip Select (second display optional)
			000	LCD_MCLK	None	Synchronous Clock (optional)
Micro Interface Graphic Display	8080 Family	Up to 16	011	LCD_D[15:0]	DATA[15:0]	Data Bus (16 bits always available)
				LCD_PCLK	RD	Read Strobe
				LCD_HSYNC	WR	Write Strobe
				LCD_VSYNC	RS	Register Select (RS = 0, command; RS = 1, data)
				$\overline{LCD_AC_ENB_CS}$	$\overline{CS}$ (or $\overline{CS0}$)	Chip Select (first display)
				LCD_MCLK	$\overline{CS1}$	Chip Select (second display optional)
			010	LCD_MCLK	None	Synchronous Clock (optional)

The timing parameters are defined by the LIDD_CS0_CONF and LIDD_CS1_CONF registers, which are described in Section 22.3.5.

Copyright © 2013–2016, Texas Instruments Incorporated

The timing configuration is based on an internal reference clock, MCLK. The MCLK is generated out of LCD_CLK, which is determined by the CLKDIV bit in the LCD_CTRL register:

$$\text{MCLK} = \text{LCD_CLK when CLKDIV} = 0.$$

$$\text{MCLK} = \frac{\text{LCD_CLK}}{\text{CLKDIV}} \text{ when CLKDIV} \neq 0.$$

See your device-specific data manual for the timing configurations supported by the LCD controller.

22.2.5 Raster Controller

Raster mode (and the use of this logic) is enabled by setting the MODESEL bit in the LCD control register (LCD_CTRL). Table 22-4 shows the active external signals when this mode is active.

Table 22-4. Operation Modes Supported by Raster Controller

Interface	Data Bus Width	Register Bits RASTER_CTRL[9, 7, 1]	Signal Name	Description
Passive (STN) Mono 4-bit	4	001	LCD_D[3:0]	Data bus
			LCD_PCLK	Pixel clock
			LCD_HSYNC	Horizontal clock(Line Clock)
			LCD_VSYNC	Vertical clock (Frame Clock)
			$\overline{\text{LCD_AC_ENB_CS}}$	AC Bias
			LCD_MCLK	Not used
Passive (STN) Mono 8-bit	8	101	LCD_D[7:0]	Data bus
			LCD_PCLK	Pixel clock
			LCD_HSYNC	Horizontal clock(Line Clock)
			LCD_VSYNC	Vertical clock (Frame Clock)
			$\overline{\text{LCD_AC_ENB_CS}}$	AC Bias
			LCD_MCLK	Not used
Passive (STN) Color	8	100	LCD_D[7:0]	Data bus
			LCD_PCLK	Pixel clock
			LCD_HSYNC	Horizontal clock(Line Clock)
			LCD_VSYNC	Vertical clock (Frame Clock)
			$\overline{\text{LCD_AC_ENB_CS}}$	AC Bias
			LCD_MCLK	Not used
Active (TFT) Color	16	x10	LCD_D[15:0]	Data bus
			LCD_PCLK	Pixel clock
			LCD_HSYNC	Horizontal clock(Line Clock)
			LCD_VSYNC	Vertical clock (Frame Clock)
			$\overline{\text{LCD_AC_ENB_CS}}$	Output enable
			LCD_MCLK	Not used

Copyright © 2013–2016, Texas Instruments Incorporated

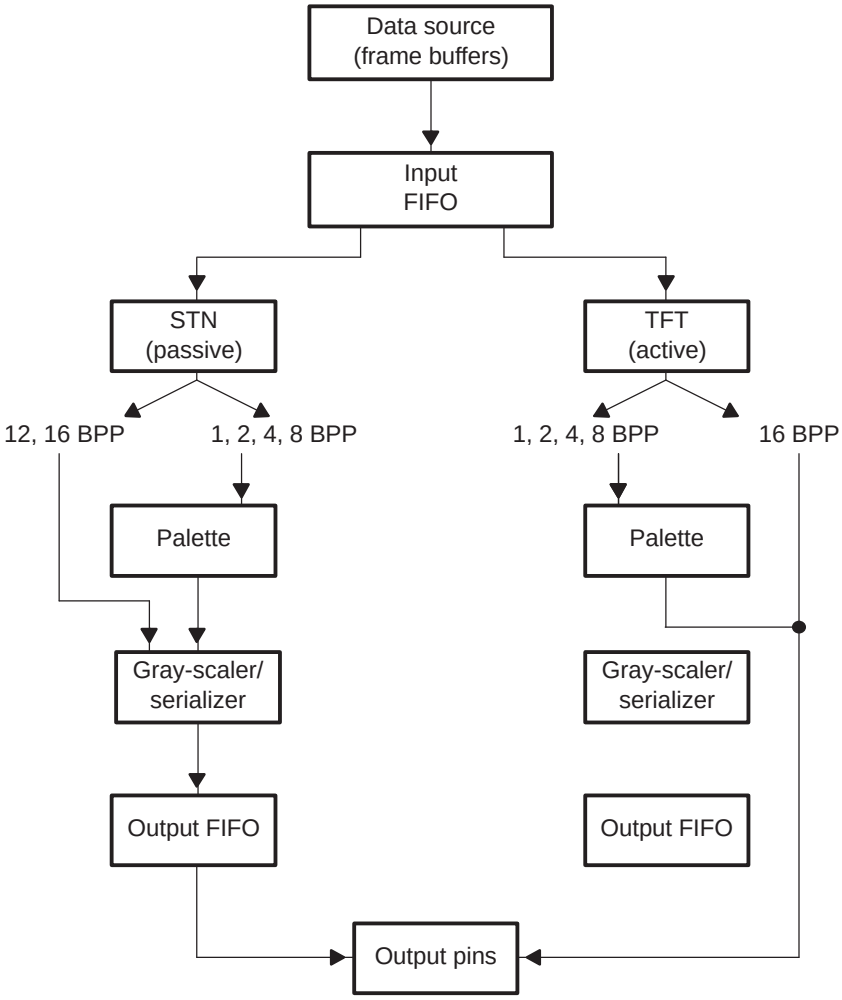
22.2.5.1 Logical Data Path

The block diagram of the Raster Controller is shown in Figure 22-1. Figure 22-3 illustrates its logical data path for various operation modes (passive (STN) versus active (TFT), various BPP size).

Figure 22-3 shows that:

- The gray-scaler/serializer and output FIFO blocks are bypassed in active (TFT) modes.
- The palette is bypassed in both 12- and 16-BPP modes.

Figure 22-3. Logical Data Path for Raster Controller



In summary:

- The display image is stored in frame buffers.
- The built-in DMA engine constantly transfers the data stored in the frame buffers to the Input FIFO.
- The Raster Controller relays data to the external pins according to the specified format.

The remainder of this section describes the functioning blocks in Figure 22-3, including frame buffers, palette, and gray-scaler/serializer. Their operation and programming techniques are covered in detail. The output format is also described in Section 22.2.5.5.

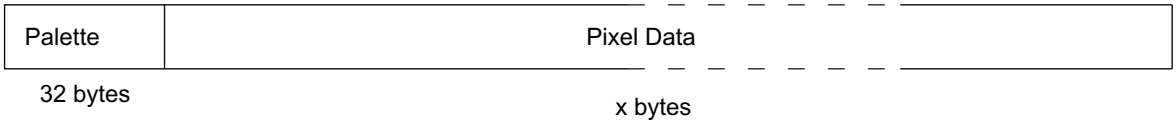
Copyright © 2013–2016, Texas Instruments Incorporated

22.2.5.2 Frame Buffer

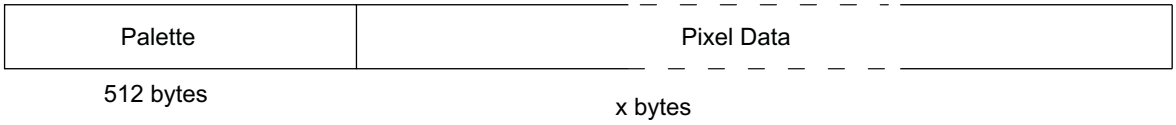
A frame buffer is a contiguous memory block, storing enough data to fill a full LCD screen. For this device, external memory needs to be used for the frame buffer. For specific details on which external memory interface (EMIF) controller can be accessed by the LCD controller, see your device-specific data manual. The data in the frame buffer consists of pixel values as well as a look-up palette. Figure 22-4 shows the frame buffer structure.

Figure 22-4. Frame Buffer Structure

1, 2, 4, 12, 16 BPP Modes



8 BPP Mode



NOTE:

- 8-BPP mode uses the first 512 bytes in the frame buffer as the palette while the other modes use 32 bytes.
- 12- and 16-BPP modes do not need a palette; i.e., the pixel data is the desired RGB value. However, the first 32 bytes are still considered a palette. The first entry should be 4000h (bit 14 is 1) while the remaining entries must be filled with 0. (For details, see Table 22-5.)
- Each entry in a palette occupies 2 bytes. As a result, 8-BPP mode palette has 256 color entries while the other palettes have up to 16 color entries.
- 4-BPP mode uses up the all the 16 entries in a palette.
- 1-BPP mode uses the first 2 entries in a palette while 2-BPP mode uses the first 4 entries. The remaining entries are not used and must be filled with 0.
- In 12- and 16-BPP modes, pixel data is RGB data. For all the other modes, pixel data is actually an index of the palette entry.

Table 22-5. Bits-Per-Pixel Encoding for Palette Entry 0 Buffer

Bit	Name	Value	Description[1] [2]
14-12	BPP		Bits-per-pixel.
		000	1 BPP
		001	2 BPP
		010	4 BPP
		011	8 BPP
		1xx	12 BPP in passive mode (TFT_STN = 0 and STN_565 = 0 in RASTER_CTRL)
			16 BPP in passive mode (TFT_STN = 0 and STN_565 = 1 in RASTER_CTRL)
			16 BPP in active mode (TFT_STN = 1 in RASTER_CTRL)

(1) Eight 1-bit pixels, four 2-bit pixels, and two 4-bit pixels are packed into each byte, and 12-bit pixels are right justified on (16-bit) word boundaries (in the same format as palette entry).

(2) For STN565, see the 16 BPP STN mode bit (Section 22.3.8.8).

Copyright © 2013–2016, Texas Instruments Incorporated

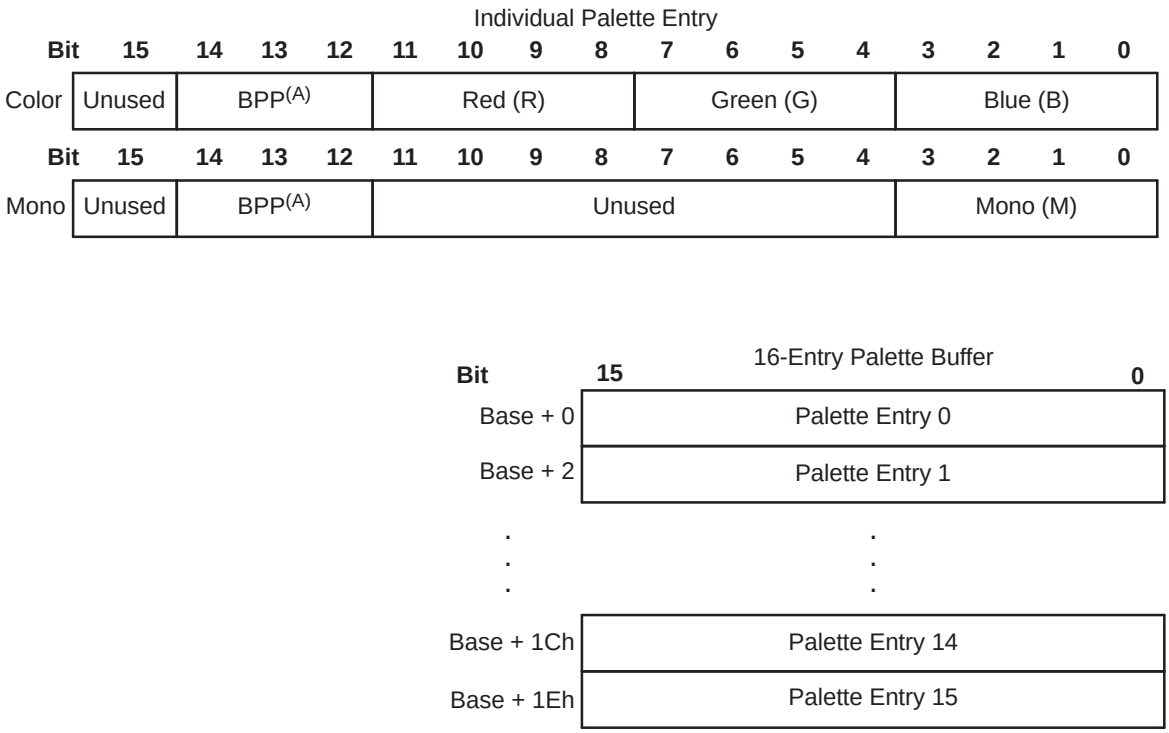
The equations shown in Table 22-6 are used to calculate the total frame buffer size (in bytes) based on varying pixel size encoding and screen sizes.

Figure 22-5 and Figure 22-6 show more detail of the palette entry organization.

Table 22-6. Frame Buffer Size According to BPP

BPP	Frame Buffer Size
1	32 + (Lines × Columns)/8
2	32 + (Lines × Columns)/4
4	32 + (Lines × Columns)/2
8	512 + (Lines × Columns)
12/16	32 + 2 × (Lines × Columns)

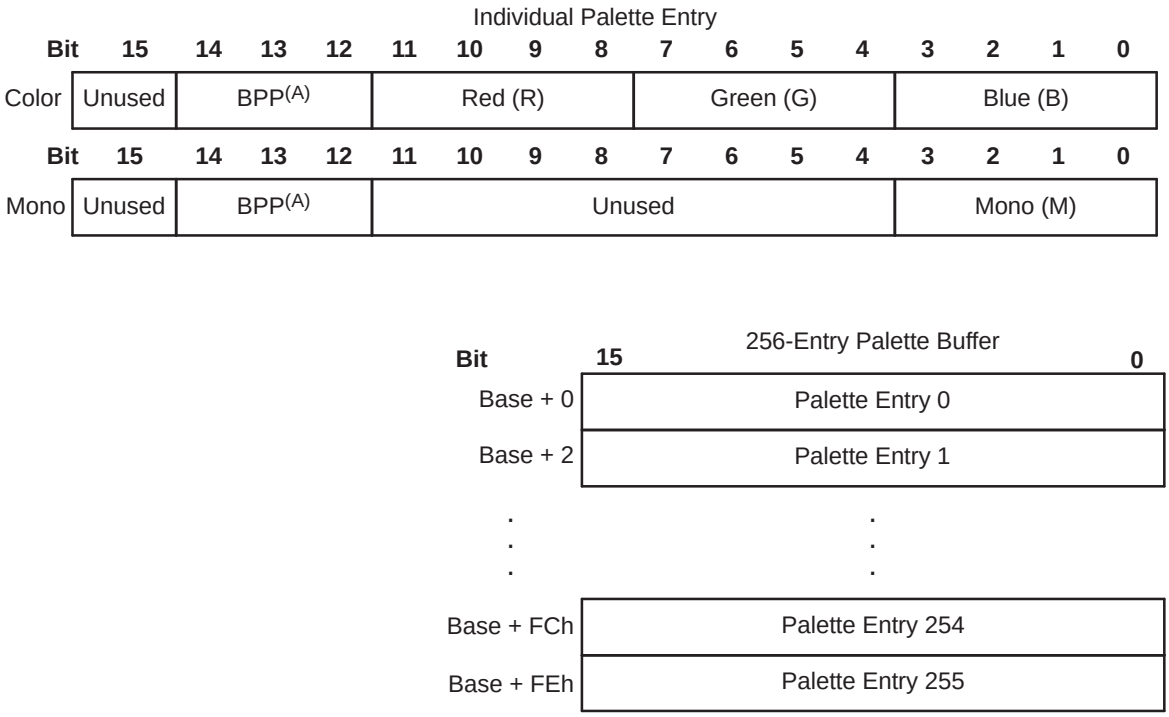
Figure 22-5. 16-Entry Palette/Buffer Format (1, 2, 4, 12, 16 BPP)



A. Bits-per-pixels (BPP) is only contained within the first palette entry (palette entry 0).

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 22-6. 256-Entry Palette/Buffer Format (8 BPP)

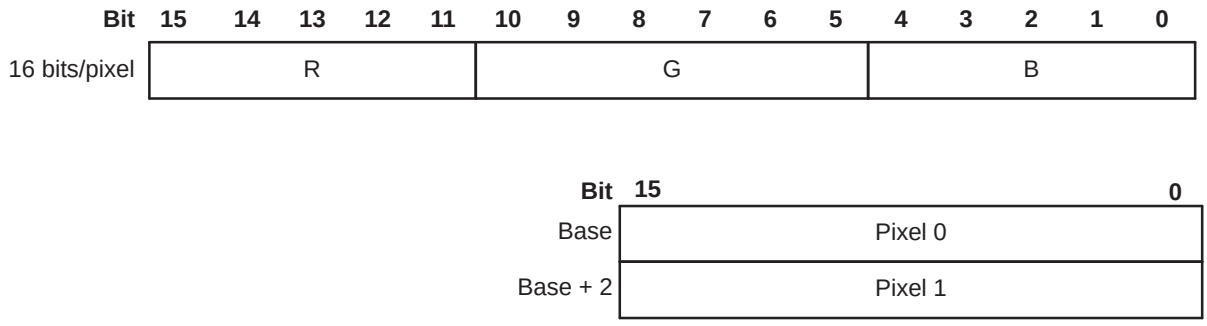


A. Bits-per-pixels (BPP) is only contained within the first palette entry (palette entry 0).

Bits 12, 13, and 14 of the first palette entry select the number of bits-per-pixel to be used in the following frame and thus the number of palette RAM entries. The palette entry is used by the Raster Controller to correctly unpack pixel data.

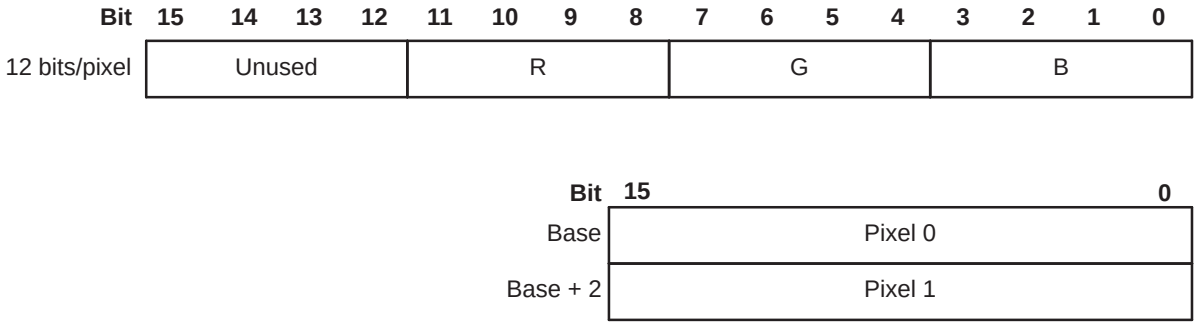
The following figures show the memory organization within the frame buffer for each pixel encoding size.

Figure 22-7. 16-BPP Data Memory Organization (TFT Mode Only)—Little Endian



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 22-8. 12-BPP Data Memory Organization—Little Endian



Unused [15-12] bits are filled with zeroes in TFT mode.

Figure 22-9. 8-BPP Data Memory Organization

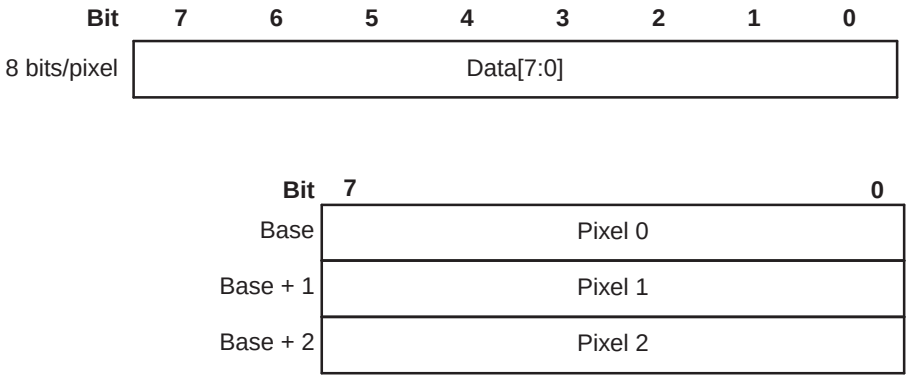
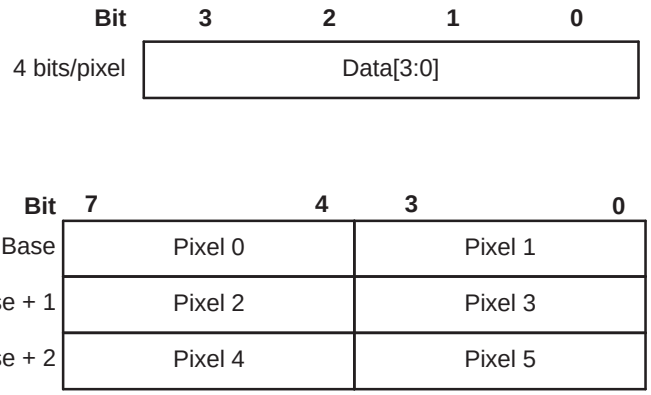


Figure 22-10. 4-BPP Data Memory Organization



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 22-11. 2-BPP Data Memory Organization

Bit	7	6	5	4	3	2	1	0
Base	Pixel 0		Pixel 1		Pixel 2		Pixel 3	
Base + 1	Pixel 4		Pixel 5		Pixel 6		Pixel 7	
Base + 2	Pixel 8		Pixel 9		Pixel 10		Pixel 11	

Figure 22-12. 1-BPP Data Memory Organization

Bit	7	6	5	4	3	2	1	0
Base	P0	P1	P2	P3	P4	P5	P6	P7
Base + 1	P8	P9	P10	P11	P12	P13	P14	P15

22.2.5.3 Palette

As explained in the previous section, the pixel data is an index of palette entry (when palette is used). The number of colors supported is given by $2^{\text{number of BPP}}$. However, due to a limitation of the gray-scaler/serializer block, fewer grayscales or colors may be supported.

The PLM field (in RASTER_CTRL) affects the palette loading:

- If PLM is 00b (palette-plus-data mode) or 01b (palette-only mode), the palette is loaded by the DMA engine at the very beginning, which is followed by the loading of pixel data.
- If PLM is 10b (data-only mode), the palette is not loaded. Instead, the DMA engine loads the pixel data immediately.

22.2.5.4 Gray-Scaler/Serializer

22.2.5.4.1 Passive (STN) Mode

Once a palette entry is selected from the look-up palette by the pixel data, its content is sent to the gray-scaler/serializer. If it is monochrome data, it is encoded as 4 bits. If it is color data, it is encoded as 4 bits (Red), 4 bits (Green), and 4 bits (Blue).

These 4-bit values are used to select one of the 16 intensity levels, as shown in Table 22-7. A patented algorithm is used during this processing to provide an optimized intensity value that matches the eye's visual perception of color/gray gradations.

22.2.5.4.2 Active (TFT) Mode

The gray-scaler/serializer is bypassed.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 22-7. Color/Grayscale Intensities and Modulation Rates

Dither Value (4-Bit Value from Palette)	Intensity (0% is White)	Modulation Rate (Ratio of ON to ON+OFF Pixels)
0000	0.0%	0
0001	11.1%	1/9
0010	20.0%	1/5
0011	26.7%	4/15
0100	33.3%	3/9
0101	40.0%	2/5
0110	44.4%	4/9
0111	50.0%	1/2
1000	55.6%	5/9
1001	60.0%	3/5
1010	66.6%	6/9
1011	73.3%	11/15
1100	80.0%	4/5
1101	88.9%	8/9
1110	100.0%	1
1111	100.0%	1

22.2.5.4.3 Summary of Color Depth

Table 22-8. Number of Colors/Shades of Gray Available on Screen

Number of BPP	Passive Mode (TFT_STN = 0)		Active Mode (TFT_STN = 1)
	Monochrome (MONO_COLOR = 1)	Color (MONO_COLOR = 0)	Color Only (MONO_COLOR = 0)
1	2 palette entries to select within 15 grayscales	2 palette entries to select within 3375 possible colors	2 palette entries to select within 4096 possible colors
2	4 palette entries to select within 15 grayscales	4 palette entries to select within 3375 possible colors	4 palette entries to select within 4096 possible colors
4	16 palette entries to select within 15 grayscales	16 palette entries to select within 3375 possible colors	16 palette entries to select within 4096 possible colors
8	Not relevant since it would consist in 256 palette entries to select within 15 grayscales, but exists anyway	256 palette entries to select 3375 possible colors	256 palette entries to select within 4096 possible colors
12	x	3375 possible colors	4096 possible colors
16	x	3375 possible colors (STN_565 = 1)	Up to 65536 possible colors

Copyright © 2013–2016, Texas Instruments Incorporated

22.2.5.5 Output Format

22.2.5.5.1 Passive (STN) Mode

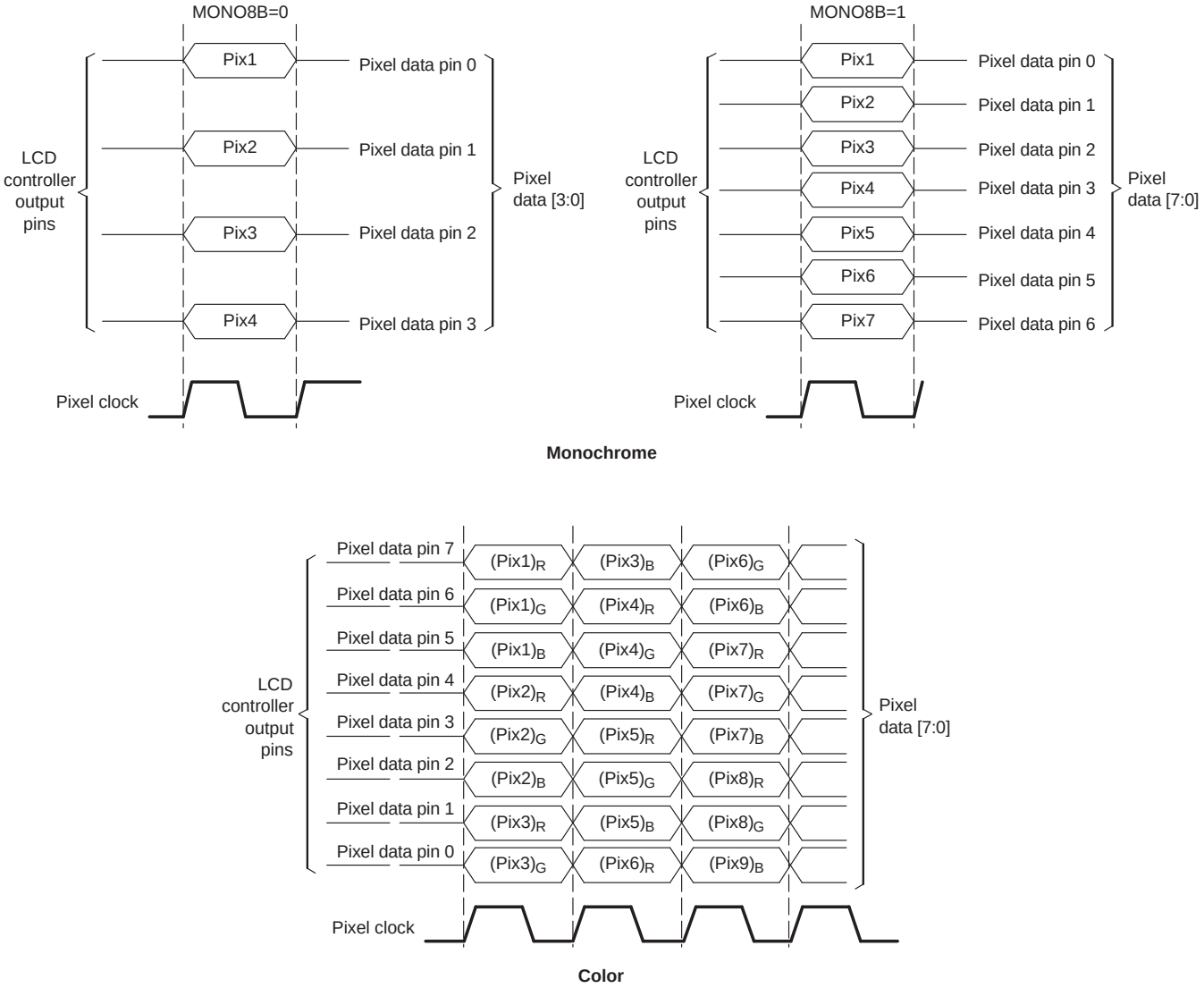
As shown in Figure 22-3, the pixel data stored in frame buffers go through palette (if applicable) and gray-scaler/serializer before reaching the Output FIFO. As a result, it is likely that the data fed to the Output FIFO is numerically different from the data in the frame buffers. (However, they represent the same color or grayscale.)

The output FIFO formats the received data according to display modes (see Table 22-4). Figure 22-13 shows the actual data output on the external pins.

22.2.5.5.2 Active (TFT) Mode

As shown in Figure 22-3, the gray-scaler/serializer and output FIFO are bypassed in active (TFT) mode. Namely, at each pixel clock, one pixel data (16 bits) is output to the external LCD.

Figure 22-13. Monochrome and Color Output



Copyright © 2013–2016, Texas Instruments Incorporated

22.2.5.6 Subpanel Feature

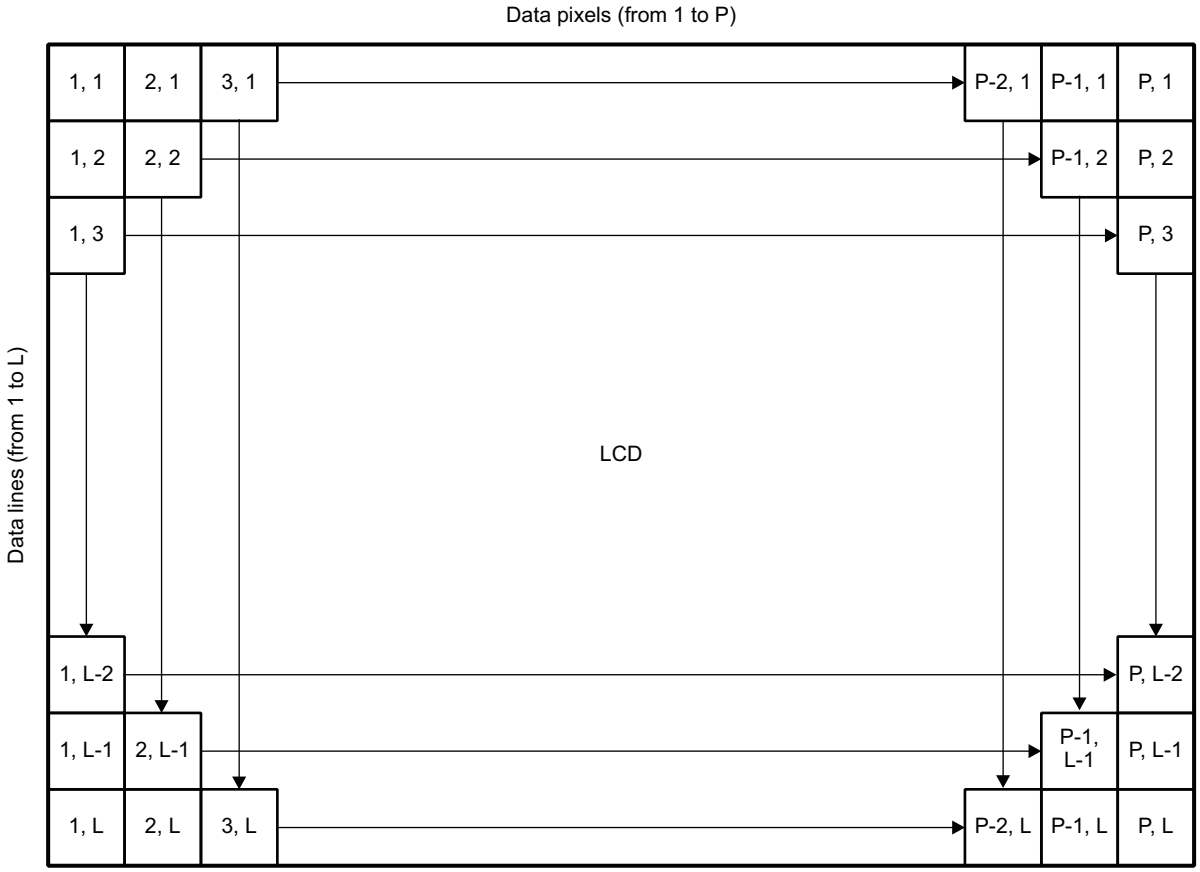
In some applications, it is desired to display only the first or last few lines of the LCD panel (see Figure 22-14). This is mainly used for power saving.

This is supported by the Raster Controller via its subpanel feature. The RASTER_SUBPANEL register fully defines its behavior, that is, the following parameters are defined:

- Whether the first or last few lines will be refreshed.
- A line number, which is the last (or first) line to be refreshed.
- The pixel data to be loaded to the refresh area.

Note that there is only one pixel value for all the pixels in the refresh area. As a result, frame buffers and DMA engine are not used in this case, which leads to power saving.

Figure 22-14. Raster Mode Display Format



Copyright © 2013–2016, Texas Instruments Incorporated

22.3 Registers

Table 22-9 lists the memory-mapped registers for the LCD module.

Table 22-9. LCD Controller (LCDC) Registers

Address Offset	Acronym	Register Description	Section
0h	REVID	LCD Revision Identification Register	Section 22.3.1
4h	LCD_CTRL	LCD Control Register	Section 22.3.2
8h	LCD_STAT	LCD Status Register	Section 22.3.3
Ch	LIDD_CTRL	LCD LIDD Control Register	Section 22.3.4
10h	LIDD_CS0_CONF	LCD LIDD CS0 Configuration Register	Section 22.3.5
14h	LIDD_CS0_ADDR	LCD LIDD CS0 Address Read/Write Register	Section 22.3.6
18h	LIDD_CS0_DATA	LCD LIDD CS0 Data Read/Write Register	Section 22.3.7
1Ch	LIDD_CS1_CONF	LCD LIDD CS1 Configuration Register	Section 22.3.5
20h	LIDD_CS1_ADDR	LCD LIDD CS1 Address Read/Write Register	Section 22.3.6
24h	LIDD_CS1_DATA	LCD LIDD CS1 Data Read/Write Register	Section 22.3.7
28h	RASTER_CTRL	LCD Raster Control Register	Section 22.3.8
2Ch	RASTER_TIMING_0	LCD Raster Timing 0 Register	Section 22.3.9
30h	RASTER_TIMING_1	LCD Raster Timing 1 Register	Section 22.3.10
34h	RASTER_TIMING_2	LCD Raster Timing 2 Register	Section 22.3.11
38h	RASTER_SUBPANEL	LCD Raster Subpanel Display Register	Section 22.3.12
40h	LCDDMA_CTRL	LCD DMA Control Register	Section 22.3.13
44h	LCDDMA_FB0_BASE	LCD DMA Frame Buffer 0 Base Address Register	Section 22.3.14
48h	LCDDMA_FB0_CEILING	LCD DMA Frame Buffer 0 Ceiling Address Register	Section 22.3.15
4Ch	LCDDMA_FB1_BASE	LCD DMA Frame Buffer 1 Base Address Register	Section 22.3.14
50h	LCDDMA_FB1_CEILING	LCD DMA Frame Buffer 1 Ceiling Address Register	Section 22.3.15

22.3.1 LCD Revision Identification Register (REVID)

The LCD revision identification register (REVID) is shown in Figure 22-15 and described in Table 22-10.

Figure 22-15. LCD Revision Identification Register (REVID)

31	0
REV	
R-4C10 0100h	

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

Table 22-10. LCD Revision Identification Register (REVID) Field Descriptions

Bit	Field	Value	Description
31-0	REV	4C10 0100h	Peripheral Identification Number

Copyright © 2013–2016, Texas Instruments Incorporated

22.3.2 LCD Control Register (LCD_CTRL)

The LCD control register (LCD_CTRL) contains the fundamental mode select bit for the LCD controller. The LCD_CTRL is shown in Figure 22-16 and described in Table 22-11.

Figure 22-16. LCD Control Register (LCD_CTRL)

31	16
Reserved	
R-0	

15	8	7	1	0
CLKDIV		Reserved		MODESEL
R/W-0		R-0		R/W-0

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

Table 22-11. LCD Control Register (LCD_CTRL) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-8	CLKDIV	0-FFh	Clock Divisor. Value (from 0 to 255) is used to specify the frequency of the pixel clock (in Raster mode) or MCLK (in LIDD mode) based on the LCD_CLK frequency. Pixel clock frequency can range from LCD_CLK/2 to LCD_CLK/255 (CLKDIV = 0 or CLKDIV = 1 are not valid). MCLK can vary from LCD_CLK to LCD_CLK/255 (CLKDIV = 0 or CLKDIV = 1 sets MCLK = LCD_CLK).
7-1	Reserved	0	Reserved
0	MODESEL		LCD Mode Select
		0	LCD Controller in LIDD mode
		1	LCD Controller in Raster mode

The 8-bit clock divider (CLKDIV) field is used to select the frequency of the pixel clock. CLKDIV can generate a range of pixel clock frequencies from LCD_CLK/2 to LCD_CLK/255. The pixel clock frequency must be adjusted to meet the required screen refresh rate.

The refresh rate depends on:

- The number of pixels for the target display.
- Whether monochrome or color mode is selected.
- The number of pixel clock delays programmed at the beginning and end of each line.
- The number of line clocks inserted at the beginning and end of each frame.
- The width of the frame clock (LCD_VSYNC) signal in active mode or VSW line clocks inserted in passive mode.
- The width of the line clock (LCD_HSYNC) signal.

All of these factors alter the time duration from one frame transmission to the next. Different display manufacturers require different frame refresh rates, depending on the physical characteristics of the display. CLKDIV is used to alter the pixel clock frequency in order to meet these requirements. Pixel clock is used to synchronously signal the device to drive data to the LCD data pins, and to signal the output FIFO to latch the data from the pins. The frequency of the pixel clock for a set CLKDIV value or the required CLKDIV value to yield a target pixel clock frequency can be calculated using the following equation:

$$\text{LCD_PCLK} = \frac{\text{LCD_CLK}}{\text{CLKDIV}}$$

Copyright © 2013–2016, Texas Instruments Incorporated

The pixel clock frequency is programmed taking into account the limitations shown in Table 22-12.

If CLKDIV equals 0 or 1, the effect is undefined. Dividing the pixel clock frequency by an odd number distorts the duty cycle.

Table 22-12. Pixel Clock Frequency Programming Limitations

Type of Screen	Output (In Bits)	Minimum Pixel Clock Divider
TFT 1,2,4,8 BPP	12 (1 pixel)	2
TFT 16 BPP	16 (1 pixel)	2
STN monochrome(4 output lines per panel)	4 (4 pixel)	4
STN monochrome(8 output lines per panel)	8 (8 pixel)	8
STN color	8 (2 2/3 pixel)	3

Copyright © 2013–2016, Texas Instruments Incorporated

22.3.3 LCD Status Register (LCD_STAT)

The LCD status register (LCD_STAT) contains bits that signal status and error conditions to the processor. Each of the LCD status bits signals an interrupt request as long as the bit is set AND the interrupt enable for that bit is also set (see the LCD raster control and LCD DMA control registers for these enables). Writing a 1 to each bit clears it; once the bit is cleared, the interrupt is cleared. The LCD_STAT is shown in Figure 22-17 and described in Table 22-13.

Figure 22-17. LCD Status Register (LCD_STAT)

31											16
Reserved											
R-0											
15	10	9	8	7	6	5	4	3	2	1	0
Reserved		EOF1	EOF0	Rsvd	PL	FUF	Rsvd	ABC	SYNC	Rsvd	DONE
R-0		R/W-0	R/W-0	R-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R-0	R/W-0

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

Table 22-13. LCD Status Register (LCD_STAT) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9	EOF1		End of Frame 1
		0	no end of frame 1 detected
		1	end of frame 1 detected
8	EOF0		End of Frame 0
		0	no end of frame 0 detected
		1	end of frame 0 detected
7	Reserved	0	Reserved
6	PL		Loaded Palette
		0	The palette is not loaded
		1	The palette is loaded
5	FUF		FIFO Underflow Status
		0	FIFO has not underrun
		1	LCD dither logic not supplying data to FIFO at a sufficient rate, FIFO has completely emptied and data pin driver logic has attempted to take added data from FIFO
4	Reserved	0	Reserved
3	ABC		AC Bias Count Status
		0	AC bias transition counter has not decremented to zero
		1	AC bias transition counter has decremented to zero, indicating that the LCD_AC_O line has transitioned the number of times that is specified by the ACB_I control bit field. Counter is reloaded with value in ACB_I but is disabled until the user clears ABC.
2	SYNC		Sync Lost
		0	normal
		1	Frame Synchronization Lost has occurred
1	Reserved	0	Reserved
0	DONE		Raster or LIDD Frame Done (shared; depends on whether Raster or LIDD mode enabled)
		0	Raster or DMA_to_LIDD engine is enabled
		1	Raster or DMA_to_LIDD disabled and the active frame has just completed

Copyright © 2013–2016, Texas Instruments Incorporated

22.3.3.1 Frame Done (DONE)

When the LCD is disabled by clearing the LCD Raster Control enable bit (RASTER_EN = 0) in the LCD Raster Control Register, the LCD allows the current frame to complete before it is disabled. After the last set of pixels is clocked out onto the LCD data pins by the pixel clock, the LCD is disabled and DONE is set.

- DONE = 1 when the frame is complete.
- DONE = 0 as long as the frame is not complete.

The frame done (DONE) bit signals the frame is complete. It is cleared when the RASTER_EN bit is set to 1 (turned ON).

22.3.3.2 Frame Synchronization Lost (SYNC)

The frame synchronization lost (SYNC) bit is set if the LCD controller detects a frame synchronization error. A frame synchronization error can occur for one of two reasons:

- when the LCD controller attempts to read what it believes to be the first word of the video buffer but cannot be recognized as such
- if the LCD controller is starved of data, which can happen due to insufficient bandwidth from the source of LCD data through the system interconnect to the LCD controller

To alleviate data bandwidth bottleneck issues to the LCD controller, the following configuration settings can be experimented with:

- Increase DDR2/mobile DDR (mDDR) memory controller (if used) Command Re-Ordering (PBBPR) setting from default to a value such as 10h to 20h
- Increase LCD FIFO threshold
- Increase priority for LCD controller DMA
- Increase burst size setting for LCD controller DMA
- Run the DDR2/mobile DDR (mDDR) memory controller (if used) at maximum clock speed

This bit is cleared by disabling the LCD controller (RASTER_EN = 0). This also resets the input FIFO in the DMA controller.

- SYNC = 1 when a frame synchronization lost occurred.
- SYNC = 0 as long as no frame synchronization error occurs.

22.3.3.3 AC-Bias Count Status (ABC)

The ac-bias count status (ABC) bit is set each time the ac-bias line transitions a particular number of times as specified by the ac-bias line transitions per interrupt (ACB_I) field in LCD Raster Timing Register 2. If ACB_I is programmed with a non-zero value, a counter is loaded with the value in ACB_I and is decremented each time the ac-bias line reverses state. When the counter reaches zero, the ABC bit is set that signals an interrupt request to the interrupt controller. The counter reloads using the value in ACB_I, but does not start to decrement again until you clear ABC by writing 0 to the LCD status register.

- ABC = 1 when the ac-bias transition counter ACB_I has decremented to 0
- ABC = 0 as long as ACB_I has not decremented to 0

22.3.3.4 FIFO Underflow Status (FUF)

The FIFO underflow status (FUF) bit is set when the input FIFO is completely empty and the LCD data pins driver logic attempts to fetch data from the FIFO. This bit is cleared by disabling the LCD controller (RASTER_EN = 0). To recover from this condition and restart normal function of the LCDC, the peripheral needs to be reset through the Power and Sleep Controller (PSC).

- FUF = 1 when the dithering logic is not supplying data to the FIFO at a sufficient rate.
- FUF = 0 as long as FIFO has not underrun.

Copyright © 2013–2016, Texas Instruments Incorporated

22.3.3.5 Loaded Palette (PL)

The loaded palette (PL) bit is a read-only bit that is set after the LCD finished loading the palette into memory.

- PL = 1 when the palette is loaded.
- PL = 0 as long as the palette is not loaded.

In data-only (PL = 10) and palette-plus-data (PL = 00) modes, write 0 to clear the interrupt. However, in the palette only (PL = 01) mode, LCD must be turned off in order to reset/clear the interrupt. But in this particular mode, make sure not to turn off the LCD before getting the loading interrupt.

Copyright © 2013–2016, Texas Instruments Incorporated

22.3.4 LCD LIDD Control Register (LIDD_CTRL)

The LCD LIDD control register (LIDD_CTRL) contains the polarity controls for LIDD output signals (to account for variety in the external LCD display/peripheral signal requirements), and the LIDD type select bits. These bits are not valid in Raster mode (when LCD control register bit 0 = 1). The LIDD_CTRL is shown in Figure 22-18 and described in Table 22-14.

NOTE: To activate DMA to drive LIDD interface, all other control bit-fields must be programmed before setting LIDD_DMA_EN = 1 and must also disable LIDD_DMA_EN bit when changing the state of any control bit within the LCD controller.

Figure 22-18. LCD LIDD Control Register (LIDD_CTRL)

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					DONE_INT_EN	DMA_CS0_CS1	LIDD_DMA_EN
R-0					R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
CS1_E1_POL	CS0_E0_POL	WS_DIR_POL	RS_EN_POL	RSPOL	LIDD_MODE_SEL		
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 22-14. LCD LIDD Control Register (LIDD_CTRL) Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	0	Reserved
10	DONE_INT_EN		LIDD Frame Done Interrupt Enable
		0	Disable LIDD Frame Done interrupt
		1	Enable LIDD Frame Done interrupt (seen on LCD Status Reg bit 0)
9	DMA_CS0_CS1		CS0/CS1 Select for LIDD DMA writes
		0	DMA writes to LIDD CS0
		1	DMA writes to LIDD CS1
8	LIDD_DMA_EN		LIDD DMA Enable
		0	Deactivate DMA control of LIDD interface; DMA control is released upon completion of transfer of the current frame of data (LIDD Frame Done) after this bit is cleared. The MPU has direct read/write access to the panel in this mode
		1	Activate DMA to drive LIDD interface to support streaming data to "smart" panels. The MPU cannot access the panel directly in this mode
7	CS1_E1_POL		Chip Select 1/Enable 1 (Secondary) Polarity Control
		0	Do Not Invert Chip Select 1/Enable 1
		1	Invert Chip Select 1/Enable 1 Chip Select 1 is active low by default; Enable 1 is active high by default
6	CS0_E0_POL		Chip Select 0/Enable 0 (Primary) Polarity Control
		0	Do Not Invert Chip Select 0/Enable 0
		1	Invert Chip Select 0/Enable 0 Chip Select 0 is active low by default; Enable 0 is active high by default
5	WS_DIR_POL		Write Strobe/Direction Polarity Control
		0	Do Not Invert Write Strobe/Direction
		1	Invert Write Strobe/Direction Write Strobe/Direction is active low/write low by default
4	RS_EN_POL		Read Strobe/Enable Polarity Control
		0	Do Not Invert Read Strobe/Enable
		1	Invert Read Strobe/Enable Read Strobe is active low by default; Enable is active high by default

Copyright © 2013–2016, Texas Instruments Incorporated

Table 22-14. LCD LIDD Control Register (LIDD_CTRL) Field Descriptions (continued)

Bit	Field	Value	Description
3	RSPOL		Register Select (RS) Polarity Control
		0	Do Not Invert RS
		1	Invert RS. RS is active low by default.
2-0	LIDD_MODE_SEL	0-7h	LIDD Mode Select. Selects type of LCD interface for the LIDD to drive. LIDD_MODE_SEL defines the function of LCD external pins as follows:

Pin	001b	011b	100b
LCD_PCLK	E	RD	N/A
LCD_HSYNC	R/$\overline{W}$	WR	R/$\overline{W}$
LCD_VSYNC	RS	RS	RS
LCD_AC_ENB_CS	$\overline{CS0}$	$\overline{CS0}$	E0
LCD_MCLK	$\overline{CS1}$	$\overline{CS1}$	E1

	0	Sync MPU68
	1h	Async MPU68
	2h	Sync MPU80
	3h	Async MPU80
	4h	Hitachi (Async)
	5h-7h	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

22.3.5 LCD LIDD CS*n* Configuration Registers (LIDD_CS0_CONF and LIDD_CS1_CONF)

The LCD LIDD CS*n* configuration registers (LIDD_CS*n*_CONF) provides the capability to configure Write and Read Strobe timing parameters to meet a variety of interface timing requirements for the Chip Select 0 (Primary) device and Chip Select 1(Secondary) device, respectively. These values are in MCLK cycles; MCLK is divided down from LCD_CLK as defined by the CLKDIV field in the LCD control register. The LIDD_CS*n*_CONF is shown in Figure 22-19 and described in Table 22-15.

Figure 22-19. LCD LIDD CS*n* Configuration Register (LIDD_CS*n*_CONF)

31	27	26	21	20	17	16
W_SU		W_STROBE		W_HOLD		R_SU
R/W-0		R/W-1		R/W-1		R/W-0

15	12	11	6	5	2	1	0
R_SU		R_STROBE		R_HOLD		TA	
R/W-0		R/W-1		R/W-1		R/W-0	

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

Table 22-15. LCD LIDD CS*n* Configuration Register (LIDD_CS*n*_CONF) Field Descriptions

Bit	Field	Value	Description
31-27	W_SU	0-1Fh	Write Strobe Set-Up cycles. Field value defines number of MCLK cycles after Data Bus/Pad Output Enable, ALE, Direction bit and Chip Select 0 have been set up before the Write Strobe is asserted when performing a write access.
26-21	W_STROBE	1-3Fh	Write Strobe Duration cycles. Field value defines number of MCLK cycles that the Write Strobe is held active when performing a write access.
20-17	W_HOLD	1-Fh	Write Strobe Hold cycles. Field value defines number of MCLK cycles that the Data Bus/Pad Output Enable, ALE, Direction bit, and Chip Select 0 are held after the Write Strobe is deasserted when performing a write access.
16-12	R_SU	0-1Fh	Read Strobe Set-Up cycles. Field value defines number of MCLK cycles after Data Bus/Pad Output Enable, ALE, Direction bit and Chip Select 0 have been set up before the Read Strobe is asserted when performing a read access.
11-6	R_STROBE	1-3Fh	Read Strobe Duration cycles. Field value defines number of MCLK cycles that the Read Strobe is held active when performing a read access.
5-2	R_HOLD	1-Fh	Read Strobe Hold cycles. Field value defines number of MCLK cycles that the Data Bus/Pad Output Enable, ALE, Direction bit, and Chip Select 0 are held after the Read Strobe is deasserted when performing a read access.
1-0	TA	0-3h	Field value defines number of MCLK cycles between the end of one CS0 device access and the start of another CS0 device access unless the two accesses are both reads, in which case this delay is not incurred. CS_DELAY = ROUNDUP(7/CLKDIV) + TA

Copyright © 2013–2016, Texas Instruments Incorporated

22.3.6 LCD LIDD CS*n* Address Read/Write Registers (LIDD_CS0_ADDR and LIDD_CS1_ADDR)

The LCD LIDD CS0 address read/write registers (LIDD_CS*n*_ADDR) are accessed by the processor to perform the address/index read or write operations on the CS0 and CS1 device respectively. Writing to LIDD_CS0_ADDR asserts CS0 and Address Latch Enable, which loads the ADR_INDX field of this register into the address generator of the peripheral device. Likewise, reading from LIDD_CS0_ADDR asserts CS0 and Address Latch Enable, which loads status information from the peripheral device into the ADR_INDX field of this register. Similarly writing to LIDD_CS1_ADDR asserts CS1 and Address Latch Enable, which loads the ADR_INDX field of this register into the address generator of the peripheral device. Likewise, reading from LIDD_CS1_ADDR asserts CS1 and Address Latch Enable, which loads status information from the peripheral device into the ADR_INDX field of this register. The LIDD_CS*n*_ADDR is shown in Figure 22-20 and described in Table 22-16.

Figure 22-20. LCD LIDD CS*n* Address Read/Write Register (LIDD_CS*n*_ADDR)

31	16
Reserved	
R-0	

15	0
ADR_INDX	
R/W-0	

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

Table 22-16. LCD LIDD CS*n* Address Read/Write Register (LIDD_CS*n*_ADDR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	ADR_INDX	0-FFFFh	Peripheral Device Address/Index value. On writes this field is loaded into the CS0 peripheral device's address generator On reads this field contains the CS0 peripheral device's status.

Copyright © 2013–2016, Texas Instruments Incorporated

22.3.7 LCD LIDD CS*n* Data Read/Write Registers (LIDD_CS0_DATA and LIDD_CS1_DATA)

The LCD LIDD CS0 data read/write registers (LIDD_CS*n*_DATA) are accessed by the processor to perform the data read or write operations on the CS0 and CS1 device respectively . Writing to LIDD_CS0_DATA asserts CS0 and deasserts Address Latch Enable, which loads the DATA field of this register into the peripheral device. Likewise, reading from this register asserts CS0 and deasserts Address Latch Enable, which loads data from the peripheral device into the DATA field of this register. Similarly writing to LIDD_CS1_DATA asserts CS1 and deasserts Address Latch Enable, which loads the DATA field of this register into the peripheral device. Likewise, reading from LIDD_CS1_DATA asserts CS1 and deasserts Address Latch Enable, which loads data from the peripheral device into the DATA field of this register. The LIDD_CS*n*_DATA is shown in Figure 22-21 and described in Table 22-17.

Figure 22-21. LCD LIDD CS*n* Data Read/Write Register (LIDD_CS*n*_DATA)

31	16
Reserved	
R-0	

15	0
DATA	
R/W-0	

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

Table 22-17. LCD LIDD CS*n* Data Read/Write Register (LIDD_CS*n*_DATA) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	DATA	0-FFFFh	Peripheral Device Data value. On writes this field is loaded into the CS0 peripheral device On reads this field contains the CS0 peripheral device's data.

Copyright © 2013–2016, Texas Instruments Incorporated

22.3.8 LCD Raster Control Register (RASTER_CTRL)

The LCD raster control register (RASTER_CTRL) contains bit-fields that are used to control various functions within the Raster controller sub-module. The RASTER_CTRL is shown in Figure 22-22 and described in Table 22-18.

Figure 22-22. LCD Raster Control Register (RASTER_CTRL)

31							25	24
Reserved								STN_565
R-0								R/W-0
23	22	21	20	19				16
TFT_ALT_MAP	NIB_MODE	PLM		FIFO_DMA_DELAY				
R/W-0	R/W-0	R/W-0		R/W-0				
15			12	11	10	9		8
FIFO_DMA_DELAY				Reserved		MONO8B		RD_ORDER
R/W-0				R-0		R/W-0		R/W-0
7	6	5	4	3	2	1		0
TFT_STN	FUF_EN	SL_EN	PL_EN	DONE_EN	AC_EN	MONO_COLOR		RASTER_EN
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 22-18. LCD Raster Control Register (RASTER_CTRL) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reserved
24	STN_565		12-Bit-Per-Pixel (5-6-5) Mode. This is only available in passive-color (STN) mode when 12 BPP is specified in the palette.
		0	Disabled: The lower 12 bits of pixel data is processed and output; i.e., “X X X X R3 R2 R1 R0 G3 G2 G1 G0 B3 B2 B1 B0” (where X is ignored by the Raster Controller).
		1	Enabled: Pixel data in the frame buffer is 16-bit but only 12 of the bits are processed and output; i.e. “R3 R2 R1 R0 X G3 G2 G1 G0 X X B3 B2 B1 B0 X” (where X is ignored by the Raster Controller).
			The data patterns above refer to frame buffer bits not output bits.
23	TFT_ALT_MAP		TFT Alternative Signal Mapping
		0	Output pixel data for 1, 2, 4, and 8 BPP will be right aligned on LCD_D[11:0]. For example, “R3 R2 R1 R0 G3 G2 G1 G0 B3 B2 B1 B0”.
		1	Output pixel data for 1, 2, 4, and 8 BPP will be converted to 5-6-5 format and transferred via LCD_D[15:0]. For example, “R3 R2 R1 R0 R3 G3 G2 G1 G0 G3 G2 B3 B2 B1 B0 B3”.
			The data patterns above refer to output bits not frame buffer bits.
22	NIB_MODE		Nibble Mode
		0	Nibble Mode disabled.
		1	Nibble Mode enabled.
			For 1, 2, and 4-BPP modes, this bit should be enabled. For 8, 12, and 16-BPP modes, this bit should be disabled.
21-20	PLM	0-3h	Palette Loading Mode
		0	Palette and data.
		1h	Palette only.
		2h	Data only.
		3h	Do not use.
19-12	FIFO_DMA_DELAY	0-FFh	FIFO DMA Request Delay. Encoded value used to specify the number of clocks the input FIFO DMA request should be disabled. The delay clock count starts after 16 words are loaded into the input FIFO. Delay Time = [(LCD Pixel Clock) × FIFO_DMA_DELAY]
11-10	Reserved	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

Table 22-18. LCD Raster Control Register (RASTER_CTRL) Field Descriptions (continued)

Bit	Field	Value	Description
9	MONO8B		Mono 8-bit Mode
		0	LCD_D[3:0] is used to output four bits each LCD_PCLK.
		1	LCD_D[7:0] is used to output eight bits each LCD_PCLK.
			This bit is ignored in all other modes.
8	RD_ORDER		Raster Data Order Select
		0	Frame buffer data is ordered from least-to-most significant bit/nibble/byte/word/d-word.
		1	Frame buffer data is ordered from most-to-least significant bit/nibble/byte/word/d-word.
7	TFT_STN		TFT or STN Mode
		0	Passive (STN) display operation enabled.
		1	Active (TFT) display operation enabled.
6	FUF_EN		FIFO Underflow Interrupt Enable
		0	Disable the FIFO Underflow interrupt.
		1	Enable the FIFO Underflow interrupt.
5	SL_EN		Sync Lost Interrupt Enable
		0	Disable the Sync Lost interrupt.
		1	Enable the Sync Lost interrupt.
4	PL_EN		Palette Loaded Interrupt Enable
		0	Disable the Palette Loaded interrupt.
		1	Enable the Palette Loaded interrupt.
3	DONE_EN		Frame Done Interrupt Enable
		0	Disable the Frame Done interrupt.
		1	Enable the Frame Done interrupt.
2	AC_EN		AC Bias Count Interrupt Enable
		0	Disable the AC Bias Count interrupt.
		1	Enable AC Bias Count interrupt.
1	MONO_COLOR		LCD Monochrome or Color
		0	Enable Color display operation.
		1	Enable Monochrome display operation.
0	RASTER_EN		LCD Raster Controller Enable
		0	Disable the LCD Raster controller.
		1	Enable the LCD Raster controller.

Copyright © 2013–2016, Texas Instruments Incorporated

22.3.8.1 LCD Raster Controller Enable (RASTER_EN)

NOTE: All other control bit-fields must be programmed before setting RASTER_EN = 1 and must also disable the LCD controller when changing the state of any control bit within the LCD controller.

The LCD Raster Controller enable (RASTER_EN) bit is used to enable and disable all LCD controller operation.

- When RASTER_EN = 0, the LCD controller is disabled.
- When RASTER_EN = 1, the LCD controller is enabled.

You can program the LCD control register (LcdControl) last, and configure all twenty-five bit fields at the same time via a word32 write to the register. If you clear RASTER_EN bit while the LCD controller is enabled, you can complete transmission of the current frame before being disabled. Completion of the current frame is signaled by the LCD controller to the DMA by setting the frame done (Done) bit within the LCD controller status register (see LCD Controller Status Register), which generates an interrupt request. If the LCD controller is disabled, the signals on pixel data [15:0] pins are set to 0 and the pixel clock, frame clock(vertical clock), line clock (horizontal clock), and ac-bias signals are set to their inactive state. This can be 0 or 1, depending on the inversions programmed in the LCD Raster Timing 2 register.

22.3.8.2 LCD Monochrome (MONO_COLOR)

The color/monochrome select (LcdBW) bit is used to determine whether the LCD controller operates in color or monochrome mode.

- When MONO_COLOR = 0:
  - Color mode is selected.
  - Palette entries are 12 bits wide, providing up to 4096 colors in active (TFT) mode and up to 3375 colors in passive (STN) color mode.
  - All three dither blocks are used (in passive mode only: TFT_STN = 0), one for each color component (R, G, B).
- When MONO_COLOR = 1:
  - Monochrome mode is selected.
  - Palette entries are 4 bits wide effective (15 levels of grayscale).
  - 4 or 8 data lines are enabled, according to the mono 8-bit mode (MONO8B).

Table 22-19. LCD Controller Data Pin Utilization for Mono/Color Passive/Active Panels

Color/Mono BPP	Passive/Active Panel	Pins
Mono 1,2,4	Passive	Pixel data[3:0]
Mono 8	Passive	Pixel data[7:0]
Color 1,2,4,8,12,16 (STN_565 = 1)	Passive	Pixel data[7:0]
Color 1,2,4,8,12	Active	Pixel data[11:0] or Pixel data[15:0] according to TFT_ALT_MAP bit in the LCD Raster Control Register (RASTER_CTRL)
Color 16	Active	Pixel data[15:0]

Copyright © 2013–2016, Texas Instruments Incorporated

22.3.8.3 TFT_STN (TFT_STN)

The TFT_STN (TFT_STN) bit selects whether the LCD controller operates in passive (STN) or active (TFT) display control mode. When TFT_STN = 0, passive or STN mode is selected. LCD data flows from the frame buffer memory, via the LCD dedicated DMA channel, to the palette (the palette is bypassed for the 12 and 16 BPP modes), to the dithering logic and the output FIFO before being output on the LCD data pins. The clock and data pin behaviors is shown for the monochrome passive mode (Figure 22-23) and for the color passive mode (Figure 22-24).

Figure 22-23. Monochrome Passive Mode Pixel Clock and Data Pin Timing

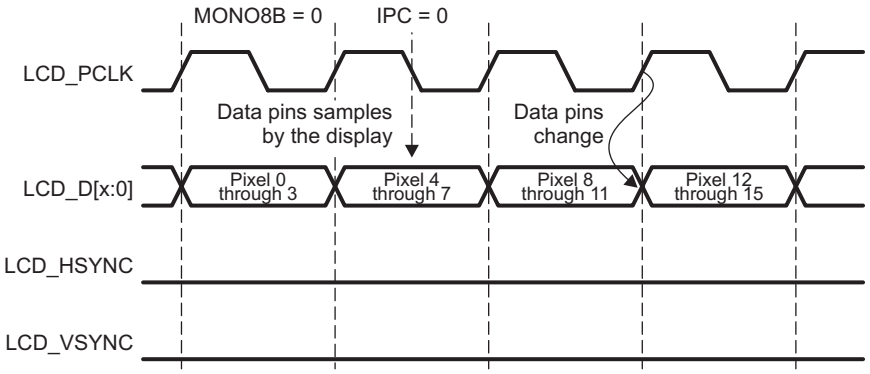
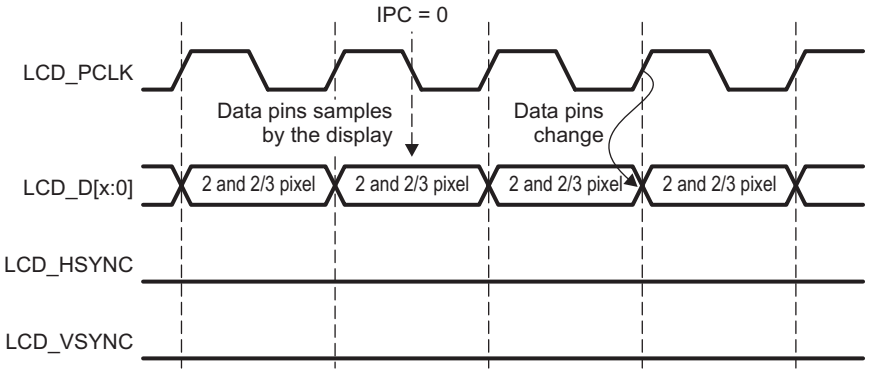


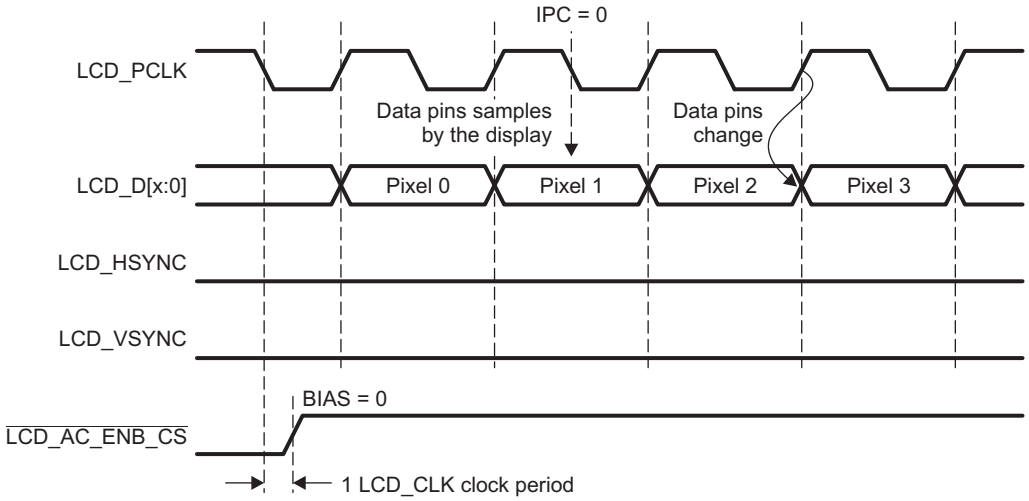
Figure 22-24. Color Passive Mode Pixel Clock and Data Pin Timing



Copyright © 2013–2016, Texas Instruments Incorporated

When TFT_STN = 1, active or TFT mode is selected. Video data is transferred via the DMA from memory to the input FIFO, then is unpacked and used to select an entry from the palette (for 1, 2, 4, and 8 bits per pixel modes), just as in passive mode. The value read from the palette; however, bypasses both the LCD dither logic and the output FIFO to be output on the LCD data pins in TFT mode. The pixel size within the frame buffer is increased to 16 bits when 12- or 16-bit pixel encoding mode is enabled (BPP = 1xx). In TFT mode for 12 and 16 bits per pixel, palette entry is not selected. The clock and data pin behaviors is shown in Figure 22-25.

Figure 22-25. Active Mode Pixel Clock and Data Pin Timing



describes the clocks and data pin behaviors in active mode. The size of the pixel encoding is increased in TFT mode because the LCD dither logic is bypassed (the dither logic only supports 4 bits to encode each color component R, G, B that limits the pixel encoding size in passive mode). Increasing the size of the pixel representation allows a total of 64K colors to be addressed using an off-chip palette in conjunction with the LCD controller.

22.3.8.4 Mono 8 Bit Mode (MONO8B)

NOTE: MONO8B does not affect any of the color modes or TFT.

The mono 8-bit mode (MONO8B) bit selects whether four or eight data lines are used to output pixel data to the LCD screen.

- When MONO8B = 0, pixel data [3:0] is used to output four pixel values to the LCD panel at each pixel clock transition.
- When MONO8B = 1, pixel data [7:0] is used to output eight pixel values to the LCD panel at each pixel clock transition

22.3.8.5 FIFO DMA Request Delay (FIFO_DMA_DELAY)

The 8-bit FIFO DMA request delay (FIFO_DMA_DELAY) field is used to select the minimum number of LCD_CLK cycles to wait between the servicing of each DMA request issued by the LCD controller, sending an address to the input FIFO. The goal is to ensure enough bandwidth to other system accesses. A delay of FIFO_DMA_DELAY cycles is inserted every 16 words read from the input FIFO. This function is a concern only in 8 BPP mode, where the palette is 256 words. The FIFO_DMA_DELAY field needs to be set properly to avoid FIFO underflow during palette loading phase. When FDD = 00h, the FIFO DMA request delay function is disabled. This function is only used for palette loading.

Copyright © 2013–2016, Texas Instruments Incorporated

22.3.8.6 Palette Loading (PLM)

The 2-bit palette loading field describes how the palette loading behaves when each new frame is loaded from memory.

- When PLM = 0, the data in the frame buffer represents the palette data and the picture data. Both palette and picture data are loaded.
- When PLM = 1 (palette-only mode), the data in the frame buffer just represents a new palette to be loaded. This data is loaded and placed into the palette. But be sure to turn off the LCD after getting the loading interrupt, or the LCD behavior would be unpredictable.
- When PLM = 2h (data loading mode), the data in the frame buffer only represents the picture data (data-only). This data is then used as an index (in the palette) or sent directly out. This mode assumes the palette was previously loaded. There is no need to keep loading the palette if it is not changing. As a matter of fact, in data-only mode, the BPP is fixed and can not change on the fly since the palette is not loaded at every frame.
- PLM = 3h is reserved.

22.3.8.7 TFT Alternate Signal Mapping (TFT_ALT_MAP)

This bit is relevant only if TFT_ALT_MAP = 1.

This bit field controls how the TFT pixel data is output. Via this feature, 12-BPP data can be output to all 16-bit LCD pins (this also applies to 1-, 2-, 4-, and 8- BPP). This feature allows you to switch BPP modes on the fly, duplicating the 12-bit output data across the 16 data lines if they are already hardwired to the 16 data lines.

Figure 22-26 shows how the four red, four green, and four blue bits are mapped to all pixel data [15-0] output pins when TFT_ALT_MAP = 1.

Figure 22-26. TFT Alternate Signal Mapping Output

Pins	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data	R3	R2	R1	R0	R3	G3	G2	G1	G0	G3	G2	B3	B2	B1	B0	B3

When TFT_ALT_MAP = 0, the four red, four green, and four blue data are right-aligned on pixel data [11-0]. The upper pixel data [15-12] are cleared to 0. There is no duplication.

Copyright © 2013–2016, Texas Instruments Incorporated

22.3.8.8 16 BPP STN Mode (STN_565)

The STN_565 bit is relevant only if TFT_STN = 0, but has no effect in 1-, 2-, 4- and 8-BPP modes. If STN_565 = 0, the frame buffer organization is in 12 BPP mode. In this mode, each color component is encoded in 4 bits, as shown in Figure 22-27.

Figure 22-27. 12-Bit STN Data in Frame Buffer

Pins	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data	Data Ignored				R3	R2	R1	R0	G3	G2	G1	G0	B3	B2	B1	B0

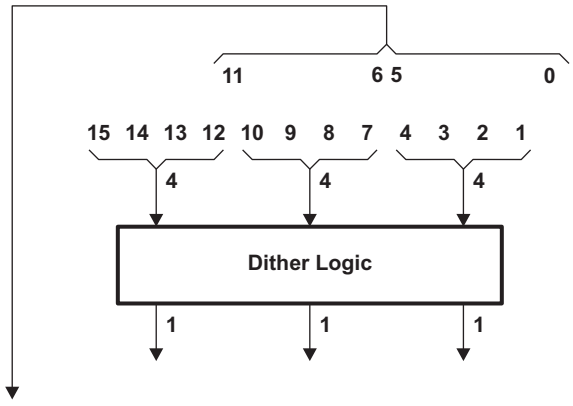
If STN_565 = 1 , the 16-bit STN mode is selected .The only difference from the 12 BPP mode is how the pixel data is organized in the frame buffer and which bits are sent to the dither logic. The 16-bit STN mode appears in frame buffer memory as shown in Figure 22-28.

Figure 22-28. 16-Bit STN Data in Frame Buffer

Pins	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0

The 16-bit STN mode only sends 12 bits to the dithering logic as well as the 12-BPP STN mode. The LSB bit of the red component (bit 11), the two LSBs of green (bits 6 and 5), and the LSB of blue (bit 0) are not sent to the dithering logic, as shown in Figure 22-29.

Figure 22-29. 16-BPP STN Mode



Note:
Red bit 11, green bits 6, 5 and blue bit 0 are not used as image data and are ignored. The 16-bit STN resolution is equal to the 12-bit STN resolution (3375 colors)

Copyright © 2013–2016, Texas Instruments Incorporated

22.3.9 LCD Raster Timing Register 0 (RASTER_TIMING_0)

The LCD raster timing 0 register (RASTER_TIMING_0) contains four bit-fields that are used as modulus values for a collection of down counters, each of which performs a different function to control the timing of several of the LCD's pins. The RASTER_TIMING_0 is shown in Figure 22-30 and described in Table 22-20.

Figure 22-30. LCD Raster Timing Register 0 (RASTER_TIMING_0)

31	24	23	16
HBP		HFP	
R/W-0		R/W-0	

15	10	9	4	3	0
HSW		PPL		Reserved	
R/W-0		R/W-0		R-0	

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

Table 22-20. LCD Raster Timing Register 0 (RASTER_TIMING_0) Field Descriptions

Bit	Field	Value	Description
31-24	HBP	0-FFh	Horizontal Back Porch. Encoded value (HBP + 1) is used to specify number of LCD_PCLKs to add to the beginning of a line transmission before the first set of pixels is output to the display (program to value minus one). Note that pixel clock is held in its inactive state during the beginning of line wait period in STN mode while it is active in TFT mode.
23-16	HFP	0-FFh	Horizontal Front Porch. Encoded value (HFP + 1) is used to specify number of LCD_PCLKs to add to the end of a line transmission before line clock is asserted (program to value minus one). Note that pixel clock is held in its inactive state during the end of line wait period in STN mode while it is active in TFT mode.
15-10	HSW	0-3Fh	Horizontal Sync Pulse Width. Encoded value (HSW + 1) is used to specify number of LCD_PCLKs to pulse the line clock at the end of each line (program to value minus one). Note that pixel clock is held in its inactive state during the generation of the line clock in STN mode while it is active in TFT mode.
9-4	PPL	0-3Fh	Pixels per Line. This value specifies the number of pixel transmissions per line . Number of pixels per Line = (PPL + 1) × 16
3-0	Reserved	0	Reserved

22.3.9.1 Pixels-Per-Line (PPL)

NOTE: PPL must be programmed to the value required minus 1 (for example, for a 640-pixel-per-line LCD panel, PPL = (640/16) - 1 = 40 - 1 = 39 = 27h).

The pixels-per-line (PPL) bit-field is used to specify the number of pixels in each line on the screen. The number of pixels per line = (PPL + 1) × 16, represents the screen width. PPL is a 6-bit value. Taking into account that the bottom 4 bits of this register are reserved, it is possible to support displays where the number of pixels-per-line ranges from 16-1024. PPL is used to count the correct number of pixel clocks that must occur before the line clock can be pulsed.

Copyright © 2013–2016, Texas Instruments Incorporated

22.3.9.2 Horizontal Synchronization Pulse Width (HSW)

NOTE: The pixel clock does not transition during the line clock pulse in passive display mode, but it transitions in active display mode.

The 6-bit horizontal synchronization pulse width (HSW) field is used to specify the pulse width of the line clock in passive mode, or horizontal synchronization pulse in active mode. The line clock (or LCD_HSYNC) is asserted each time a line or row of pixels is output to the display and a programmable number of pixel clock delays have elapsed. When line clock is asserted, the value in HSW is transferred to a 6-bit down counter that uses the programmed pixel clock frequency to decrement. When the counter reaches zero, the line clock is negated. HSW can be programmed to generate a line clock pulse width ranging from 1–64 pixel clock periods (program to value required minus 1).

22.3.9.3 Horizontal Front Porch (HFP)

NOTE: The pixel clock does not transition during these dummy pixel clock cycles in passive display mode, but it transitions continuously in active display mode.

The 8-bit horizontal front porch (HFP) field is used to specify the number of dummy pixel clocks to insert at the end of each line or row of pixels before pulsing the line clock(or LCD_HSYNC) pin. Once a complete line of pixels is transmitted to the LCD driver, the value in HFP is used to count the number of pixel clocks to wait before pulsing the line clock. HFP generates a wait period ranging from 1–256 pixel clock cycles (program to value required minus 1).

22.3.9.4 Horizontal Back Porch (HBP)

NOTE: The pixel clock does not transition during these dummy pixel clock cycles in passive display mode, but it transitions continuously in active display mode.

The 8-bit horizontal back porch (HBP) field is used to specify the number of dummy pixel clocks to insert at the beginning of each line or row of pixels. After the line clock(or LCD_HSYNC) for the previous line has been negated, the value in HBP is used to count the number of pixel clocks to wait before starting to output the first set of pixels in the next line. HBP generates a wait period ranging from 1–256 pixel clock cycles (program to value required minus 1).

Copyright © 2013–2016, Texas Instruments Incorporated

22.3.10 LCD Raster Timing Register 1 (RASTER_TIMING_1)

The LCD raster timing 1 register (RASTER_TIMING_1) contains four bit-fields that are used as modulus values for a collection of down counters, each of which performs a different function to control the timing of several of the LCD's pins. The (RASTER_TIMING_1 is shown in Figure 22-31 and described in Table 22-21.

Figure 22-31. LCD Raster Timing Register 1 (RASTER_TIMING_1)

31	24	23	16
VBP		VFP	
R/W-0		R/W-0	

15	10	9	0
VSW		LPP	
R/W-0		R/W-0	

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

Table 22-21. LCD Raster Timing Register 1 (RASTER_TIMING_1) Field Descriptions

Bit	Field	Value	Description
31-24	VBP	0-FFh	Vertical Back Porch. Encoded value (VBP) used to specify number of line clock (LCD_HSYNC) periods to add to the beginning of a frame before the first set of pixels is output to the display. Note that line clock transitions during the insertion of the extra line clock periods.
23-16	VFP	0-FFh	Vertical Front Porch. Encoded value (VFP) used to specify number of line clock (LCD_HSYNC) periods to add to the end of each frame. Note that the line clock transitions during the insertion of the extra line clock periods.
15-10	VSW	0-3Fh	Vertical Synchronization Pulse Width. In TFT mode, encoded value (VSW + 1) defines the number of line clock (LCD_HSYNC) cycles to hold the frame clock (LCD_VSYNC) active. In STN mode, encoded value (VSW + 1) specifies the number of extra line clock (LCD_HSYNC) cycles to be inserted after the vertical front porch (VFP) period has elapsed.
9-0	LPP	0-3FFh	Lines per Panel. Encoded value (LPP + 1) used to specify number of lines per panel. It represents the total number of lines on the LCD.

22.3.10.1 Lines Per Panel (LPP)

NOTE: LPP must be programmed to the value required minus 1 (C7h for 200 lines per panel).

The lines per panel (LPP) bit-field is used to specify the number of lines or rows per LCD panel being controlled. It represents the total number of lines for the entire LCD display (the screen height). LPP is a 10-bit value, which represents between 1–1024 lines per panel. LPP is used to count the correct number of line clocks that must occur before the frame clock can be pulsed.

Copyright © 2013–2016, Texas Instruments Incorporated

22.3.10.2 Vertical Synchronization Pulse Width (VSW)

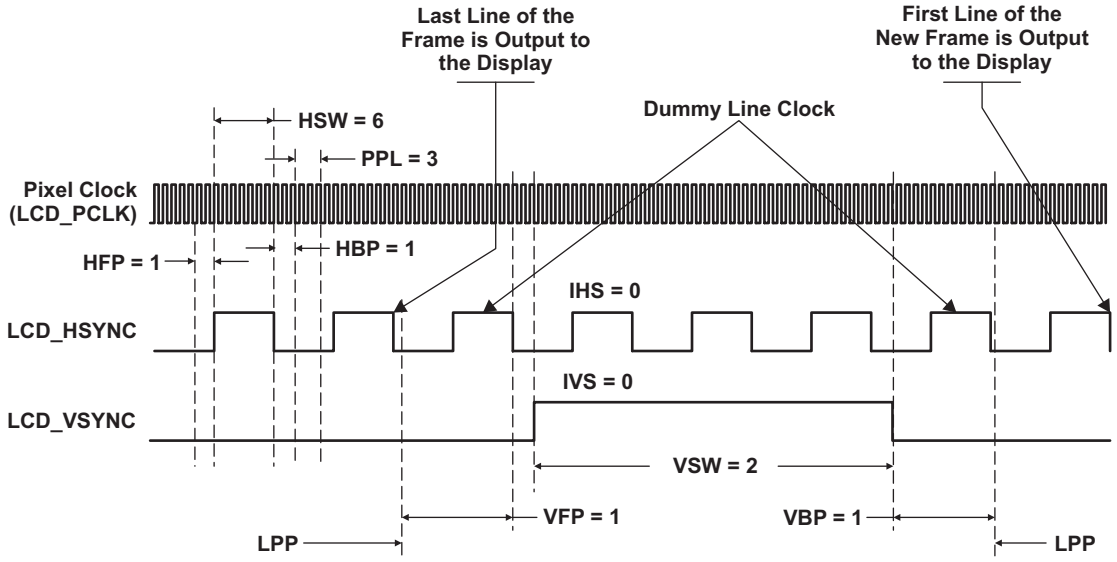
The 6-bit vertical synchronization pulse width (VSW) field is used to specify the pulse width of the vertical synchronization pulse in active mode or is used to add extra dummy line clock cycles between the vertical front porch and vertical back porch in passive mode.

22.3.10.2.1 Active Mode

NOTE: Remember that most of the parameters (HSW, HFP, PPL, HBP) must be programmed to value required minus 1.

In active mode (TFT_STN = 1), VSYNC is asserted each time the last line or row of pixels from the previous frame is output to the display and a programmable number of line clock delays (VFP) has elapsed. When the frame clock (LCD_VSYNC) is asserted, the value in VSW is transferred to a 6-bit down counter that uses the line clock frequency to decrement. When the counter reaches zero, the frame clock (LCD_VSYNC) is negated. VSW can be programmed to generate a vertical synchronization pulse width ranging from 1–64 line clock periods (program to value required minus 1—see Figure 22-32). The following frame starts after LCD_VSYNC is deasserted and a programmable number of line clock delays (VBP) has elapsed.

Figure 22-32. Vertical Synchronization Pulse Width (VSW) - Active Mode



22.3.10.2.2 Passive Mode

NOTE: The pixel clock does not transition during the whole dummy line clock periods that are inserted in passive mode before the frame pulse. The line clock does transition during the insertion of the dummy line clock cycles. VSW must be long enough to load the palette.

In passive mode (TFT_STN = 0), VSW does not affect the timing of the frame clock, but instead can be used to add extra line clock cycles between the end and beginning of frame line clock cycle counts. The total number of line clock cycles that are inserted between each frame is equal to the sum of the values in VFP, VSW and VBP. A counter is used to insert dummy line clock cycles between frames by first using the value in VFP, then VSW, then VBP. You must ensure that the sum of the values in the three fields is equal to the total number of line clock cycles that are needed between frames. The LCD controller frame clock pin is asserted on the rising-edge of the first pixel clock for each frame. The frame clock remains asserted for the remainder of the first line as pixels are output to the display, also during the assertion of the first line clock for the frame, and then negated on the rising-edge of the first pixel clock of the second line of each frame.

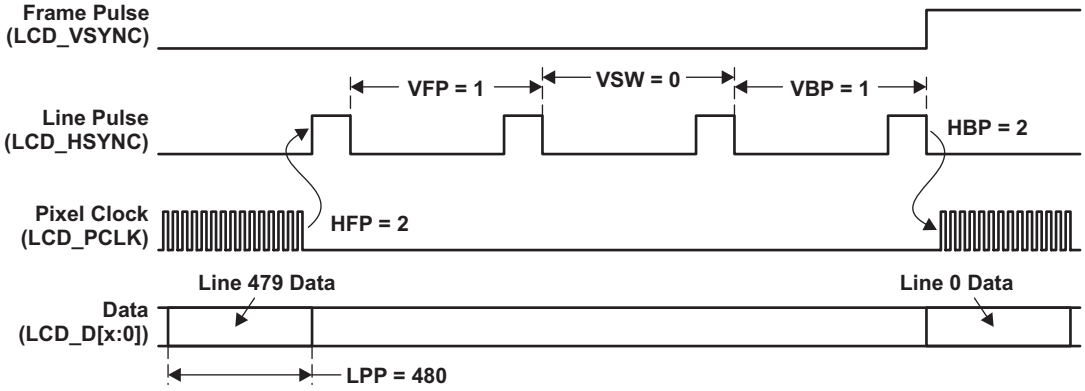
Copyright © 2013–2016, Texas Instruments Incorporated

22.3.10.3 Vertical Front Porch (VFP)

NOTE: Remember that VSW must be programmed to value required minus 1.

The 8-bit vertical front porch (VFP) field is used to specify the number of line clocks to insert at the end of each frame. Once a complete frame of pixels is transmitted to the LCD display, the value in VFP is used to count the number of line clock periods to wait. After the count has elapsed the LCD_VSYNC signal is pulsed in active mode or extra line clocks are inserted as specified by the VSW bit-field in passive mode. VFP generates from 0–255 line clock cycles (see Figure 22-33).

Figure 22-33. Vertical Front Porch (VFP)



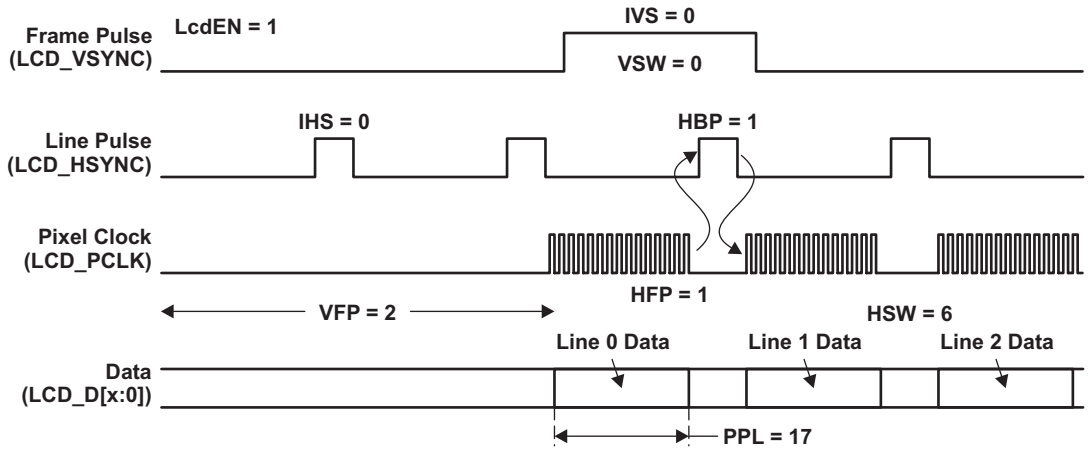
Copyright © 2013–2016, Texas Instruments Incorporated

22.3.10.4 Vertical Back Porch (VBP)

NOTE: The line clock transitions during the generation of the VBP line clock wait periods. Note also that you must adjust the value of VBP appropriately such that enough line clock cycles are permitted to elapse; this allows the palette to be completely filled via the DMA, and allows a sufficient number of encoded pixel values to be input from the frame buffer, processed by the dither logic, then placed in the output FIFO, ready to be output to the LCD data lines.

The 8-bit vertical back porch (VBP) field is used to specify the number of line clocks (or LCD_HSYNC) to insert at the beginning of each frame. The VBP count starts just after the LCD_VSYNC signal for the previous frame has been negated for active mode, or the extra line clocks have been inserted as specified by the VSW bit-field in passive mode. After this has occurred, the value in VBP is used to count the number of line clock periods to insert before starting to output pixels in the next frame. VBP generates from 0–255 extra line clock cycles (see Figure 22-34).

Figure 22-34. Vertical Back Porch (VBP)



Copyright © 2013–2016, Texas Instruments Incorporated

22.3.11 LCD Raster Timing Register 2 (RASTER_TIMING_2)

LCD raster timing 2 register (RASTER_TIMING_2) contains bit-fields that are used to control various functions associated with the timing of the LCD controller. The RASTER_TIMING_2 is shown in Figure 22-35 and described in Table 22-22.

Figure 22-35. LCD Raster Timing Register 2 (RASTER_TIMING_2)

31	26	25	24
Reserved		SYNC_CTRL	SYNC_EDGE
R-0		R/W-0	R/W-0

23	22	21	20	19	16
BIAS	IPC	IHS	IVS	ACB_I	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

15	8	7	0
ACB		Reserved	
R/W-0		R-0	

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

Table 22-22. LCD Raster Timing Register 2 (RASTER_TIMING_2) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	Reserved
25	SYNC_CTRL		Horizontal and Vertical Sync Control
		0	Inactive. SYNC_EDGE is ignored and the activation and deactivation of LCD_HSYNC and LCD_VSYNC will be defined by bit 22 below.
		1	Active. Allow SYNC_EDGE to define the LCD_PCLK edge (rising or falling) used to activate and deactivate LCD_HSYNC and LCD_VSYNC.
24	SYNC_EDGE		Horizontal and Vertical Sync Edge. The activation and deactivation of LCD_HSYNC and LCD_VSYNC will occur on the defined LCD_PCLK edge.
		0	Rising edge.
		1	Falling edge.
			SYNC_CTRL must be active in order to use this bit.
23	BIAS		Invert AC Bias
		0	$\overline{\text{LCD_AC_ENB_CS}}$ is an active-high pulse.
		1	$\overline{\text{LCD_AC_ENB_CS}}$ is an active-low pulse.
			In STN mode, the activation of this bit is ignored.
22	IPC		Invert Pixel Clock
		0	LCD Data (LCD_D[15:0]) is driven on the rising edge of LCD_PCLK, while LCD_VSYNC, LCD_HSYNC, and $\overline{\text{LCD_AC_ENB_CS}}$ are driven on the falling edge.
		1	LCD Data (LCD_D[15:0]) is driven on the falling edge of LCD_PCLK, while LCD_VSYNC, LCD_HSYNC, and $\overline{\text{LCD_AC_ENB_CS}}$ are driven on the rising edge.
			LCD_VSYNC and LCD_HSYNC may be altered as defined by bits 24 and 25 above.
21	IHS		Invert Line Clock
		0	LCD_HSYNC is an active high pulse.
		1	LCD_HSYNC is an active low pulse.
20	IVS		Invert Frame Clock
		0	LCD_VSYNC is an active high pulse.
		1	LCD_VSYNC is an active low pulse.
19-16	ACB_I	0-Fh	This value is used to specify the number of AC Bias ($\overline{\text{LCD_AC_ENB_CS}}$) output transition counts before setting the AC bias interrupt bit in register LCD_STAT. This counter is stopped when the interrupt is set and remains stopped until the AC bias interrupt status is cleared. A value of zero will not produce an interrupt.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 22-22. LCD Raster Timing Register 2 (RASTER_TIMING_2) Field Descriptions (continued)

Bit	Field	Value	Description
15-8	ACB	0-FFh	AC Bias Pin Frequency. This value defines the number of Line Clock (LCD_HSYNC) cycles to count before transitioning signal LCD_AC_ENB_CS. This output may be used to periodically invert the polarity of the power supply in order to prevent a display DC charge build-up on the LCD panel. AC Bias Time Period = [2 × ((Line Clock) × (ABC))]
7-0	Reserved	0	Reserved

22.3.11.1 AC-Bias Pin Frequency (ACB)

NOTE: The 8-bit ac-bias frequency (ACB) field has no effect in active mode. This is due to the fact that the pixel clock transitions continuously in active mode ; the ac-bias line is used as an output enable signal. The ac-bias is asserted by the LCD controller in active mode; this occurs whenever pixel data is driven out to the data pins to signal to the display when it can latch pixels using the pixel clock.

The 8-bit ac-bias frequency (ACB) field is used to specify the number of line clock periods to count between each toggle of the ac-bias pin. After the LCD controller is enabled, the value in ACB is loaded to an 8-bit down counter, and the counter begins to decrement using the line clock. When the counter reaches zero it stops, the state of ac-bias pin is reversed, and the whole procedure starts again. The number of line clocks between each ac-bias pin transition ranges from 1–256 (program to value required minus 1). This line is used by the LCD display to periodically reverse the polarity of the power supplied to the screen to eliminate DC offset.

22.3.11.2 AC-Bias Line Transitions Per Interrupt (ACB_I)

The 4-bit ac-bias line transitions per interrupt (ACB_I) field is used to specify the number of line transitions to count before setting the ac-bias count status (ABC) bit in the LCD controller status register that signals an interrupt request. After the LCD controller is enabled, the value in ACB_I is loaded to a 4-bit down counter, and the counter decrements each time the ac-bias line state is inverted. When the counter reaches zero it stops, the ac-bias count (ABC) bit is set in the status register. Once ABC is set, the 4-bit down counter is reloaded with the value in ACB_I and is disabled until ABC is cleared. Once ABC is cleared by the CPU, the down counter is enabled, and again decrements each time the ac-bias line is flipped. The number of ac-bias line transitions between each interrupt request ranges from 0 to 15. Programming ACB_I = 0000 disables the ac-bias line transitions per interrupt function.

22.3.11.3 Invert VSYNC (IVS)

The invert VSYNC(IVS) bit is used to invert the polarity of the frame clock (VSYNC).

- When IVS = 1, the frame clock (VSYNC) is active low.
- When IVS = 0, it is active high.

22.3.11.4 Invert HSYNC (IHS)

The invert HSYNC(IHS) bit is used to invert the polarity of the line clock (HSYNC).

- When IHS = 1, the line clock (HSYNC) is active low.
- When IHS = 0, it is active high.

22.3.11.5 Invert Pixel Clock (IPC)

The invert pixel clock (IPC) bit is used to select the edge of the pixel clock that drives pixel data out onto the LCD data lines.

- When IPC = 1, data is driven onto the LCD data lines on the falling edge of the pixel clock.
- When IPC = 0, data is driven onto the LCD data lines on the rising edge of the pixel clock.

Copyright © 2013–2016, Texas Instruments Incorporated

22.3.11.6 Invert Output Enable (BIAS)

NOTE: BIAS does not affect the ac-bias pin in passive display mode.

The invert output enable (BIAS) bit is used to select the active or inactive state of the output enable signal in active display mode. In this mode, the ac-bias pin is used as an enable that signals the device when data is actively being driven out using the pixel clock.

- When BIAS = 1, the ac-bias pin is active low. In active display mode, data is driven onto the LCD data lines on the programmed edge of the pixel clock when ac-bias pin is in its active state.
- When BIAS = 0, the ac-bias pin is active high.

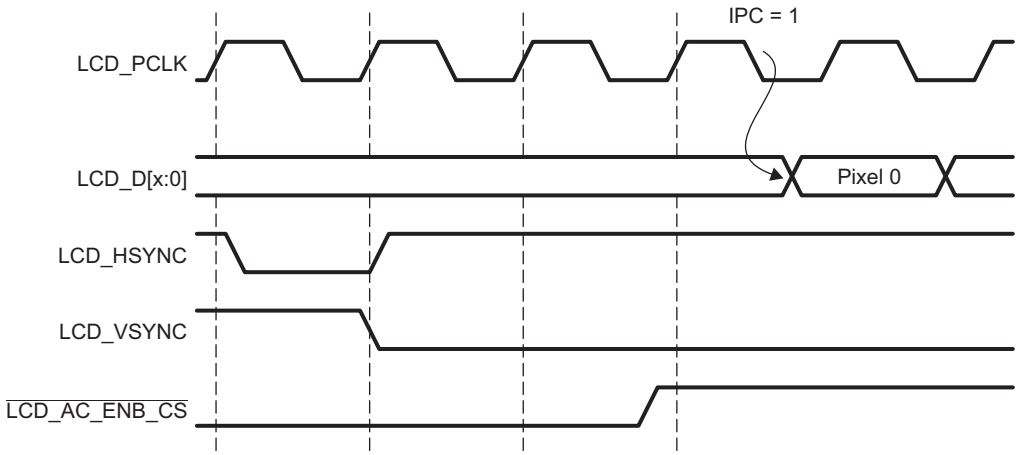
22.3.11.7 Horizontal and Vertical Sync Edge (SYNC_EDGE)

This bit determines whether the HSYNC/VSYNC is driven on the rising or falling edge of the pixel clock (see the SYNC_CTRL bit; SYNC_CTRL must be turned on first). By default, the LCD_HSYNC and LCD_VSYNC signals are driven on the falling edge of the pixel clock, and the pixel data is driven on the rising edge of pixel clock. However, if the invert pixel clock (IPC) bit is set to 1, then the LCD_HSYNC and LCD_VSYNC signals are driven on rising edge of pixel clock and pixel data is driven on falling edge. By setting the SYNC_EDGE bit and enabling it (SYNC_CTRL = 1), you can control on which edge the signals are driven.

In Figure 22-36:

- IPC = 1, pixel data is driven onto the LCD data lines on the falling edge of the pixel clock.
- SYNC_CTRL = 0, LCD_HSYNC and LCD_VSYNC signals are driven on opposite edges of the pixel clock from pixel data (=> rising edge). The rising edge or falling edge is determined by the IPC bit.

Figure 22-36. SYNC_CTRL = 0, IPC = 1 in TFT Mode

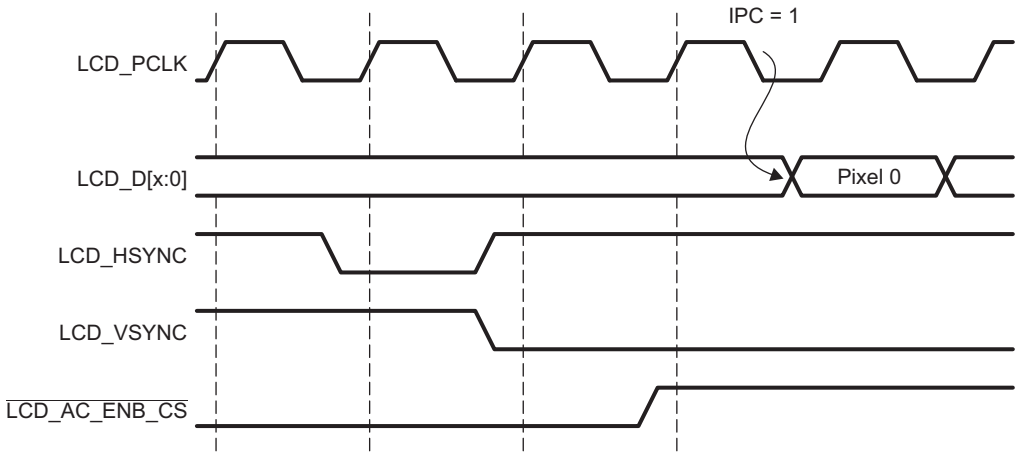


Copyright © 2013–2016, Texas Instruments Incorporated

In Figure 22-37:

- IPC = 1, pixel data is driven on the falling edge of the pixel clock.
- SYNC_CTRL = 1, LCD_HSYNC and LCD_VSYNC signals are driven according to the SYNC_EDGE bit.
- SYNC_EDGE = 0, LCD_HSYNC and LCD_VSYNC signals are driven on the falling edge of the pixel clock.

Figure 22-37. SYNC_CTRL = 1, SYNC_EDGE = 0, and IPC = 1



22.3.11.8 Horizontal and Vertical Sync Control (SYNC_CTRL)

This bit enables/disables the possibility to make HSYNC and VSYNC programmable.

- When SYNC_CTRL = 1, HSYNC and VSYNC are driven according to the SYNC_EDGE bit.
- When SYNC_CTRL = 0, HSYNC and VSYNC are driven on opposite edges of the pixel clock from pixel data.

Copyright © 2013–2016, Texas Instruments Incorporated

22.3.12 LCD Raster Subpanel Display Register (RASTER_SUBPANEL)

LCD raster subpanel display register (RASTER_SUBPANEL) displays only the first or last *n* lines of the panel and sends a fixed content for the others is supported with the LCD raster subpanel display register. For these others, there is no access to the frame buffer because the value stored in Default Pixel Data will be used. The RASTER_SUBPANEL is shown in Figure 22-38 and described in Table 22-23.

If LPPT is greater than the number of lines per panel then:

- If HOLS = 1: normal panel
- If HOLS = 0: panel with default data

LPPT has a minimum value = 2. DPD, LPPT, HOLS, and SPEN bit fields and bits are not considered if SPEN = 0.

Figure 22-38. LCD Raster Subpanel Display Register (RASTER_SUBPANEL)

31	30	29	28 26	25 16
SPEN	Rsvd	HOLS	Reserved	LPPT
R/W-0	R-0	R/W-0	R-0	R/W-0

15 4	3 0
DPD	Reserved
R/W-0	R-0

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

Table 22-23. LCD Raster Subpanel Display Register (RASTER_SUBPANEL) Field Descriptions

Bit	Field	Value	Description
31	SPEN		Subpanel Enable
		0	Disabled
		1	Enabled
30	Reserved	0	Reserved
29	HOLS		High or Low Signal. The field indicates the position of subpanel compared to the LPPT value.
		0	Low, below
		1	High, above
28-26	Reserved	0	Reserved
25-16	LPPT	0-3FFh	Line Per Panel Threshold. This field defines the number of lines to be refreshed (1 to 1024).
15-4	DPD	0-FFFh	Default Pixel Data. DPD defines the default value of the pixel data sent to the panel for the lines until LPPT is reached or after passing the LPPT.
3-0	Reserved	0	Reserved

22.3.12.1 Default Pixel Data (DPD)

The default pixel data (DPD) defines a default value, which is sent to the display in either the top or bottom region of the screen delimited by the LPPT threshold.

22.3.12.2 Line-per-Panel Threshold (LPPT)

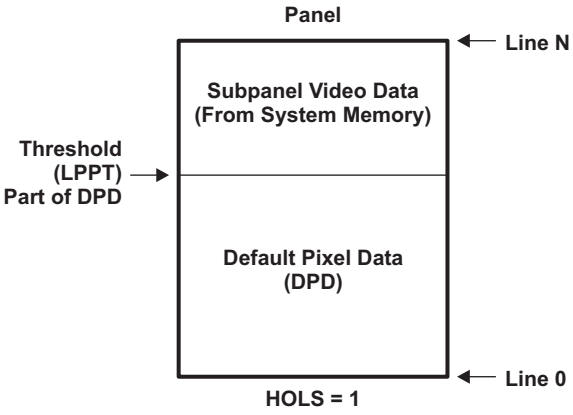
The line-per-panel threshold bit-field delimits the screen portion filled with data fetched from the frame buffer (the subpanel) and the rest of the screen filled with default pixel data (DPD). Note that the LPPT line number points on a line filled with a DPD value when HOLS = 1, but on one filled with video data when HOLS = 0 (see Figure 22-39 and Figure 22-40).

Copyright © 2013–2016, Texas Instruments Incorporated

22.3.12.3 High Or Low Signal (HOLS)

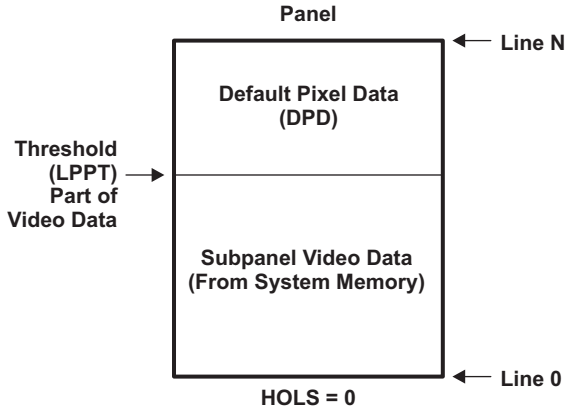
The HOLS bit indicates the position of the subpanel compared to the LPPT value. When HOLS = 1, the image from system memory is displayed above the threshold value. The threshold value is the line number where the DPD value begins to be displayed. The rest of the screen is filled with DPD value.

Figure 22-39. Subpanel Display: SPEN = 1, HOLS = 1



When HOLS = 0, the beginning of the screen is filled with DPD value until the LPPT excluded. From the LPPT line number, the rest of the screen (below LPPT) displays the image from system memory.

Figure 22-40. Subpanel Display: SPEN = 1, HOLS = 0



The bottom of the panel is line 0, and top line of the panel is Line N (where N is the number of lines-per-panel). For example, if you want to display four lines of video data at the bottom of the panel, the correct settings are HOLS = 0 and LPPT = 3. Here, the amount of video data to be transferred from the DMA_LCD channel is only four lines.

If the LPPT is above the number of LPP, then:

- When HOLS = 1: panel with default data (whole panel is filled with DPD value).
- When HOLS = 0: normal panel (whole panel is filled with video data from the frame buffer).

Copyright © 2013–2016, Texas Instruments Incorporated

22.3.13 LCD DMA Control Register (LCDDMA_CTRL)

The LCD DMA control register (LCDDMA_CTRL) contains bits that control the LCD channel operation. The LCDDMA_CTRL is shown in Figure 22-41 and described in Table 22-24.

Figure 22-41. LCD DMA Control Register (LCDDMA_CTRL)

31	16
Reserved	
R-0	

15	11	10	8
Reserved		TH_FIFO_READY	
R-0		R/W-0	

7	6	4	3	2	1	0
Reserved	BURST_SIZE		Reserved	EOF_INTEN	BIGENDIAN	FRAME_MODE
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 22-24. LCD DMA Control Register (LCDDMA_CTRL) Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	0	Reserved
10-8	TH_FIFO_READY	0-7h	DMA FIFO threshold. The input FIFO becomes ready so that the Raster controller can start reading its content only when the number of dwords (1 dword is 4 bytes) specified by TH_FIFO_READY have been loaded by the DMA from the frame buffer to the input FIFO.
		0	8 dwords
		1h	16 dwords
		2h	32 dwords
		3h	64 dwords
		4h	128 dwords
		5h	256 dwords
		6h	512 dwords
		7h	Reserved
7	Reserved	0	Reserved
6-4	BURST_SIZE	0-7h	Burst Size setting for DMA transfers (all DMA transfers are 32 bits wide)
		0	Burst size of 1
		1h	Burst size of 2
		2h	Burst size of 4
		3h	Burst size of 8
		4h	Burst size of 16
		5h-7h	Reserved
3	Reserved	0	Reserved
2	EOF_INTEN		End-of-Frame Interrupt Enable. Setting this bit allows the end-of-frame 0 or 1 status bits in the LCD status register to trigger an interrupt.
		0	End-of-frame 0/1 interrupt disabled
		1	End-of-frame 0/1 interrupt enabled
1	BIGENDIAN		Big Endian Enable. Use this bit when the processor is operating in Big Endian mode and writes to the frame buffer(s) are less than 32 bits wide; in this scenario only, change the byte alignment for data coming into the FIFO from the frame buffer(s).
		0	Big Endian data reordering disabled
		1	Big Endian data reordering enabled
0	FRAME_MODE		Frame Mode
		0	One frame buffer (FB0 only) used.
		1	Two frame buffers used; DMA ping-pongs between FB0 and FB1 in this mode.

Copyright © 2013–2016, Texas Instruments Incorporated

22.3.14 LCD DMA Frame Buffer *n* Base Address Registers (LCDDMA_FB0_BASE and LCDDMA_FB1_BASE)

The LCD DMA frame buffer *n* base address register (LCDDMA_FB*n*_BASE) contains the start address for frame buffer *n*, specified in 32-bit words. The LCDDMA_FB*n*_BASE is shown in Figure 22-42 and described in Table 22-25.

Figure 22-42. LCD DMA Frame Buffer *n* Base Address Register (LCDDMA_FB*n*_BASE)

31	0
FB*n*_BASE	
R/W-0	

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

Table 22-25. LCD DMA Frame Buffer *n* Base Address Register (LCDDMA_FB*n*_BASE) Field Descriptions

Bit	Field	Value	Description
31-0	FB*n*_BASE	0-FFFF FFFFh	Frame Buffer *n* Base Address pointer. Note: The 2 LSBs are hardwired to 00b.

22.3.15 LCD DMA Frame Buffer *n* Ceiling Address Registers (LCDDMA_FB0_CEILING and LCDDMA_FB1_CEILING)

The LCD DMA frame buffer *n* ceiling address register (LCDDMA_FB*n*_CEILING) contains the ending address for frame buffer *n*, specified in 32-bit words. The LCDDMA_FB*n*_CEILING is shown in Figure 22-43 and described in Table 22-26.

Figure 22-43. LCD DMA Frame Buffer *n* Ceiling Address Register (LCDDMA_FB*n*_CEILING)

31	0
FB*n*_CEIL	
R/W-0	

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

Table 22-26. LCD DMA Frame Buffer *n* Ceiling Address Register (LCDDMA_FB*n*_CEILING) Field Descriptions

Bit	Field	Value	Description
31-0	FB*n*_CEIL	0-FFFF FFFFh	Frame Buffer *n* Ceiling Address pointer. Note: The 2 LSBs are hardwired to 00b.

Copyright © 2013–2016, Texas Instruments Incorporated

Multichannel Audio Serial Port (McASP)

This chapter describes the multichannel audio serial port (McASP). See your device-specific data manual to determine how many McASPs are available on your device.

Topic **Page**

23.1 Registers **996**

Copyright © 2013–2016, Texas Instruments Incorporated

Introduction

Purpose of the Peripheral

The multichannel audio serial port (McASP) functions as a general-purpose audio serial port optimized for the needs of multichannel audio applications. The McASP is useful for time-division multiplexed (TDM) stream, Inter-IC Sound (I2S) protocols, and intercomponent digital audio interface transmission (DIT).

The McASP consists of transmit and receive sections that may operate synchronized, or completely independently with separate master clocks, bit clocks, and frame syncs, and using different transmit modes with different bit-stream formats. The McASP module also includes up to 16 serializers that can be individually enabled to either transmit or receive. In addition, all of the McASP pins can be configured as general-purpose input/output (GPIO) pins.

23.0.16 Features

Features of the McASP include:

- Two independent clock generator modules for transmit and receive
  - Clocking flexibility allows the McASP to receive and transmit at different rates. For example, the McASP can receive data at 48 kHz but output up-sampled data at 96 kHz or 192 kHz.
- Independent transmit and receive modules, each includes:
  - Programmable clock and frame sync generator
  - TDM streams from 2 to 32, and 384 time slots
  - Support for time slot sizes of 8, 12, 16, 20, 24, 28, and 32 bits
  - Data formatter for bit manipulation
- Up to 16 individually assignable serial data pins:
  - McASP0 can have up to 16 serial data pins
- Glueless connection to audio analog-to-digital converters (ADC), digital-to-analog converters (DAC), codec, digital audio interface receiver (DIR), and S/PDIF transmit physical layer components
- Wide variety of Inter-IC Sound (I2S) and similar bit-stream formats
- 384-slot TDM with external digital audio interface receiver (DIR) device
  - For DIR reception, an external DIR receiver integrated circuit should be used with I2S output format and connected to the McASP receive section.
- Extensive error checking and recovery:
  - Transmit underruns and receiver overruns due to the system not meeting real-time requirements
  - Early or late frame sync in TDM mode
  - Out-of-range high-frequency master clock for both transmit and receive
  - External error signal coming into the AMUTEIN input
  - DMA error due to incorrect programming
- McASP Audio FIFO (AFIFO):
  - Provides additional data buffering
  - Provides added tolerance to variations in host/DMA controller response times
  - May be used as a DMA event pacer
  - Independent Read FIFO and Write FIFO
  - 256 bytes of RAM for each FIFO (read and write)
    - 256 bytes = four 32-bit words per serializer in the case of 16 data pins
    - 256 bytes = 64 32-bit words in the case of one data pin
  - Option to bypass Write FIFO and/or Read FIFO independently

Copyright © 2013–2016, Texas Instruments Incorporated

23.0.17 Protocols Supported

The McASP supports a wide variety of protocols.

- Transmit section supports
  - Wide variety of I2S and similar bit-stream formats
  - TDM streams from 2 to 32 time slots
  - S/PDIF, IEC60958-1, AES-3 formats
- Receive section supports
  - Wide variety of I2S and similar bit-stream formats
  - TDM streams from 2 to 32 time slots
  - TDM stream of 384 time slots specifically designed for easy interface to external digital interface receiver (DIR) device transmitting DIR frames to McASP using the I2S protocol (one time slot for each DIR subframe)

The transmit and receive sections may each be individually programmed to support the following options on the basic serial protocol:

- Programmable clock and frame sync polarity (rising or falling edge): ACLKR/X, AHCLKR/X, and AFSR/X
- Slot length (number of bits per time slot): 8, 12, 16, 20, 24, 28, 32 bits supported
- Word length (bits per word): 8, 12, 16, 20, 24, 28, 32 bits; always less than or equal to the time slot length
- First-bit data delay: 0, 1, 2 bit clocks
- Left/right alignment of word inside slot
- Bit order: MSB first or LSB first
- Bit mask/pad/rotate function
  - Automatically aligns data for CPU internally in either Q31 or integer formats
  - Automatically masks nonsignificant bits (sets to 0, 1, or extends value of another bit)

In I2S mode, the transmit and receive sections can support simultaneous transfers on up to all serial data pins operating as 192 kHz stereo channels.

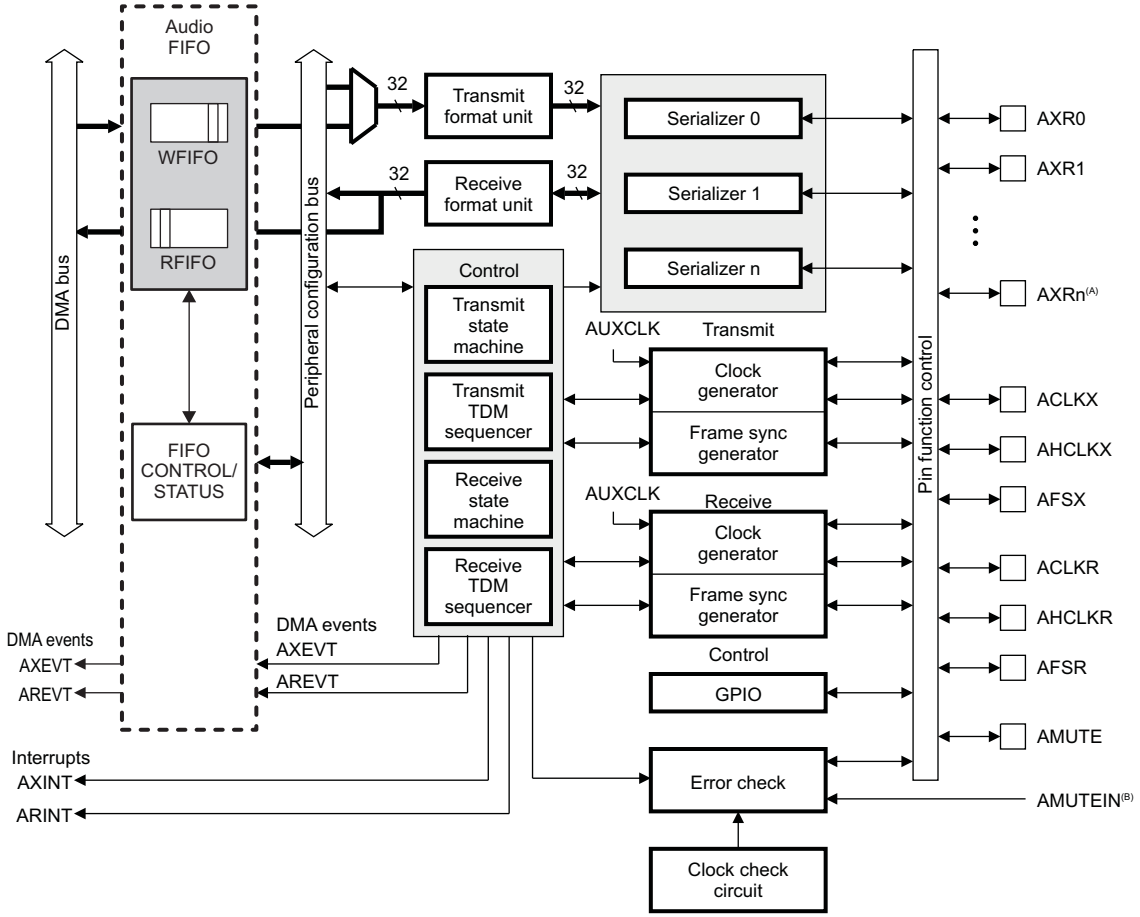
In DIT mode, the transmitter can support a 192 kHz frame rate (stereo) on up to 2 serial data pins simultaneously (note that the internal bit clock for DIT runs two times faster than the equivalent bit clock for I2S mode, due to the need to generate Biphase Mark Encoded Data).

Copyright © 2013–2016, Texas Instruments Incorporated

23.0.18 Functional Block Diagram

A block diagram of the McASP is shown in Figure 23-1. The McASP has independent receive/transmit clock generators and frame sync generators.

Figure 23-1. McASP Block Diagram



A McASP0 has up to 16 serial data pins, *n* = 15

B One of the external pins, see your device-specific data manual.

Copyright © 2013–2016, Texas Instruments Incorporated

23.0.18.1 System Level Connections

Figure 23-2 through Figure 23-5 show examples of McASP usage in digital audio encoder/decoder systems.

Figure 23-2. McASP to Parallel 2-Channel DACs

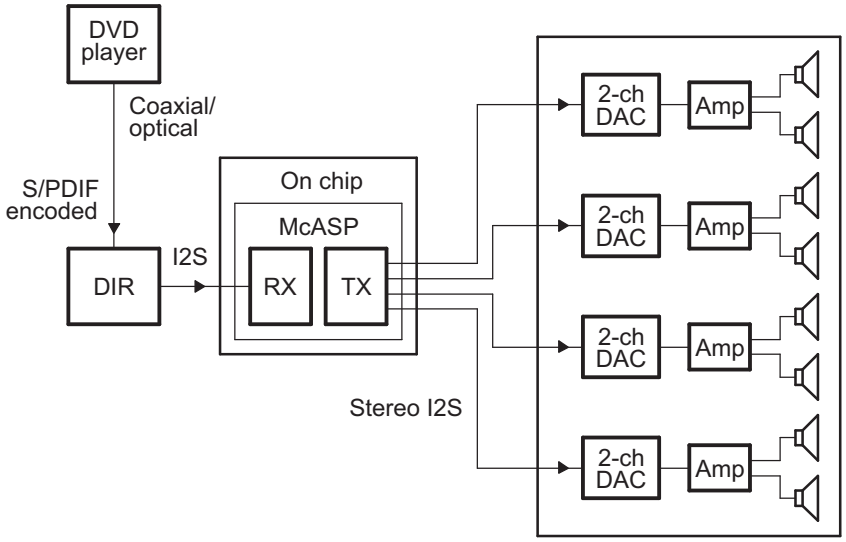
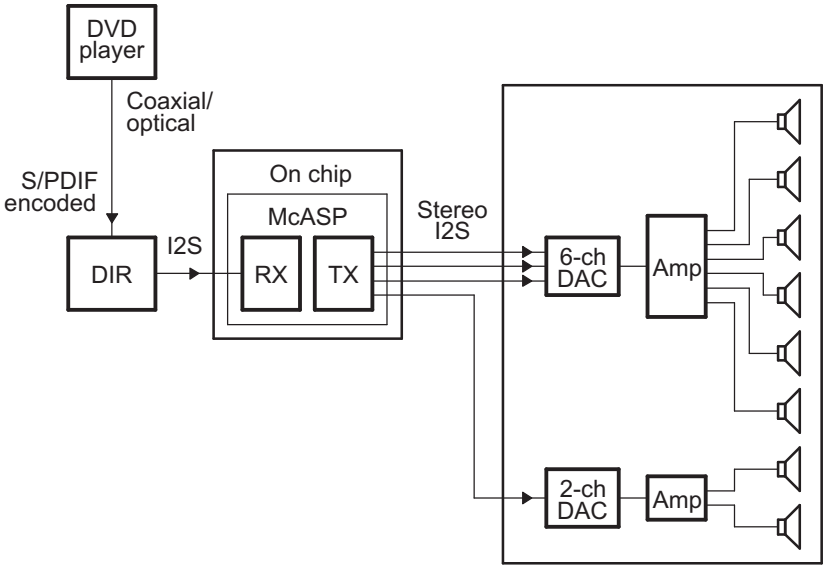


Figure 23-3. McASP to 6-Channel DAC and 2-Channel DAC



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 23-4. McASP to Digital Amplifier

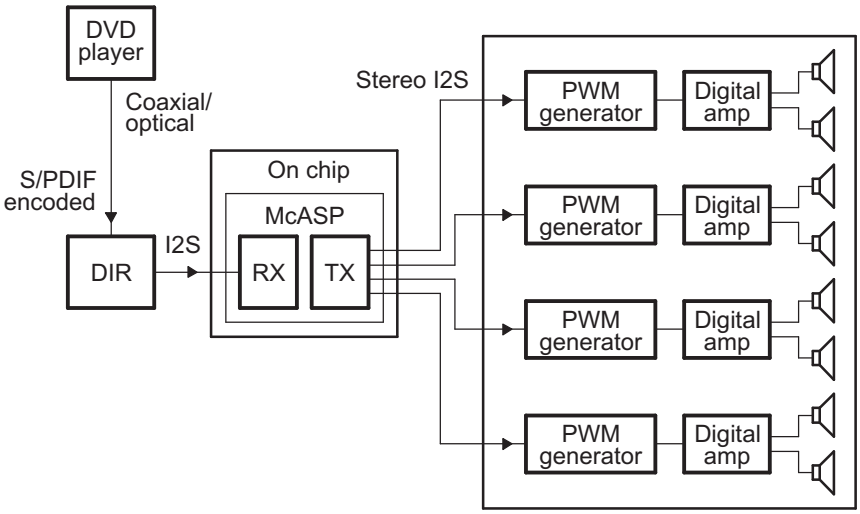
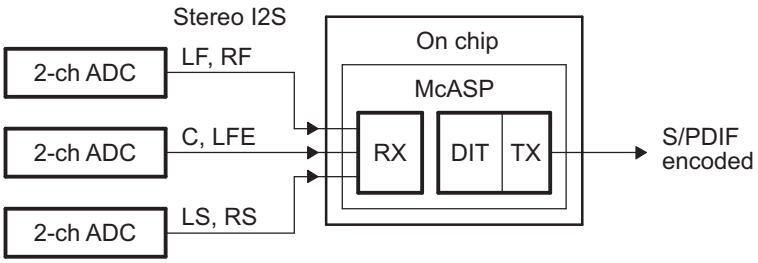


Figure 23-5. McASP as Digital Audio Encoder



Copyright © 2013–2016, Texas Instruments Incorporated

Industry Standard Compliance Statement

The McASP supports the following industry standard interfaces.

23.0.18.1 TDM Format

The McASP transmitter and receiver support the multichannel, synchronous time-division-multiplexed (TDM) format via the TDM transfer mode. Within this transfer mode, a wide variety of serial data formats are supported, including formats compatible with devices using the Inter-IC Sound (I2S) protocol. This section briefly discusses the TDM format and the I2S protocol.

23.0.18.1.1 TDM Format

The TDM format is typically used when communicating between integrated circuit devices on the same printed circuit board or on another printed circuit board within the same piece of equipment. For example, the TDM format is used to transfer data between the CPU and one or more analog-to-digital converter (ADC), digital-to-analog converter (DAC), or S/PDIF receiver (DIR) devices.

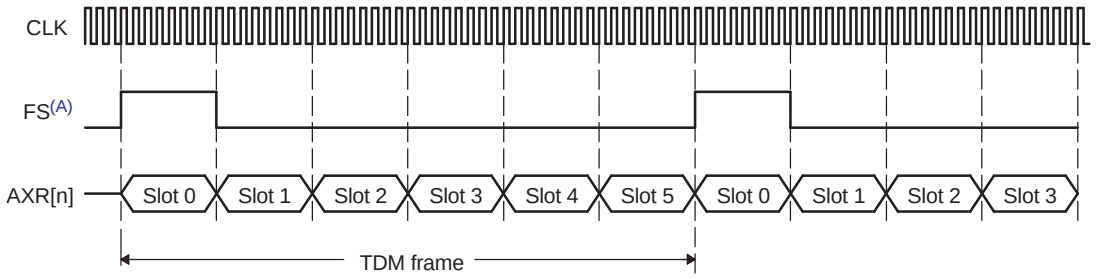
The TDM format consists of three components in a basic synchronous serial transfer: the clock, the data, and the frame sync. In a TDM transfer, all data bits (AXR[n]) are synchronous to the serial clock (ACLKX or ACLKR). The data bits are grouped into words and slots (as defined in Section 23.0.19). The "slots" are also commonly referred to as "time slots" or "channels" in TDM terminology. A frame consists of multiple slots (or channels). Each TDM frame is defined by the frame sync signal (AFSX or AFSR). Data transfer is continuous and periodic, since the TDM format is most commonly used to communicate with data converters that operate at a fixed sample rate.

There are no delays between slots. The last bit of slot N is followed immediately on the next serial clock cycle with the first bit of slot N + 1, and the last bit of the last slot is followed immediately on the next serial clock with the first bit of the first slot. However, the frame sync may be offset from the first bit of the first slot with a 0, 1, or 2-cycle delay.

It is required that the transmitter and receiver in the system agree on the number of bits per slot, since the determination of a slot boundary is not made by the frame sync signal (although the frame sync marks the beginning of slot 0 and the beginning of a new frame).

Figure 23-6 shows the TDM format. Figure 23-7 shows the different bit delays from the frame sync.

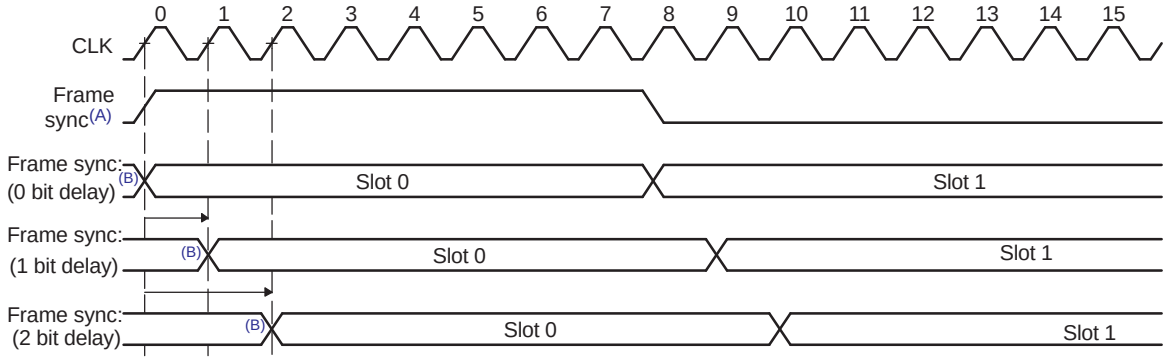
Figure 23-6. TDM Format–6 Channel TDM Example



A FS duration of slot is shown. FS duration of single bit is also supported.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 23-7. TDM Format Bit Delays from Frame Sync



A FS duration of slot is shown. FS duration of single bit is also supported.

B Last bit of last slot of previous frame. No gap between this bit and the first bit of slot 0 is allowed.

In a typical audio system, one frame of data is transferred during each data converter sample period f_s. To support multiple channels, the choices are to either include more time slots per frame (thus operating with a higher bit clock rate), or to use additional data pins to transfer the same number of channels (thus operating with a slower bit clock rate).

For example, a particular six channel DAC may be designed to transfer over a single serial data pin AXR[n] as shown in Figure 23-6. In this case the serial clock must run fast enough to transfer a total of 6 channels within each frame period. Alternatively, a similar six channel DAC may be designed to use three serial data pins AXR[0,1,2], transferring two channels of data on each pin during each sample period (Figure 23-8). In the latter case, if the sample period remains the same, the serial clock can run three times slower than the former case. The McASP is flexible enough to support either type of DAC.

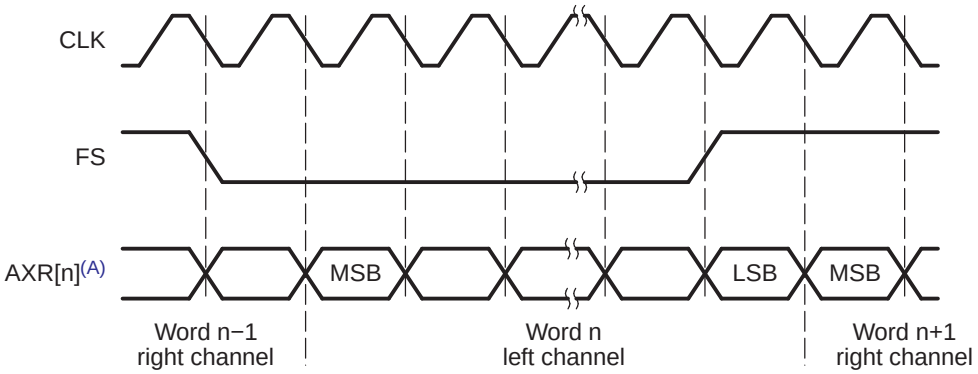
23.0.18.1.2 Inter-IC Sound (I2S) Format

The Inter-IC Sound (I2S) format is used extensively in audio interfaces. The TDM transfer mode of the McASP supports the I2S format when configured to 2 slots per frame.

I2S format is specifically designed to transfer a stereo channel (left and right) over a single data pin AXR[n]. "Slots" are also commonly referred to as "channels". The frame width duration in the I2S format is the same as the slot size. The frame signal is also referred to as "word select" in the I2S format. Figure 23-8 shows the I2S protocol.

The McASP supports transfer of multiple stereo channels over multiple AXR[n] pins.

Figure 23-8. Inter-IC Sound (I2S) Format



A 1 to 16 data pins may be supported.

Copyright © 2013–2016, Texas Instruments Incorporated

23.0.18.2 S/PDIF Coding Format

The McASP transmitter supports the S/PDIF format with 3.3V biphase-mark encoded output. The S/PDIF format is supported by the digital audio interface transmit (DIT) transfer mode of the McASP. This section briefly discusses the S/PDIF coding format.

23.0.18.2.1 Biphase-Mark Code (BMC)

In S/PDIF format, the digital signal is coded using the biphase-mark code (BMC). The clock, frame, and data are embedded in only one signal—the data pin AXR[n]. In the BMC system, each data bit is encoded into two logical states (00, 01, 10, or 11) at the pin. These two logical states form a cell. The duration of the cell, which equals to the duration of the data bit, is called a time interval. A logical 1 is represented by two transitions of the signal within a time interval, which corresponds to a cell with logical states 01 or 10. A logical 0 is represented by one transition within a time interval, which corresponds to a cell with logical states 00 or 11. In addition, the logical level at the start of a cell is inverted from the level at the end of the previous cell. Figure 23-9 and Table 23-1 show how data is encoded to the BMC format.

As shown in Figure 23-9, the frequency of the clock is twice the unencoded data bit rate. In addition, the clock is always programmed to 128 × f_s, where f_s is the sample rate (see Section 23.0.18.2.3 for details on how this clock rate is derived based on the S/PDIF format). The device receiving in S/PDIF format can recover the clock and frame information from the BMC signal.

Figure 23-9. Biphase-Mark Code (BMC)

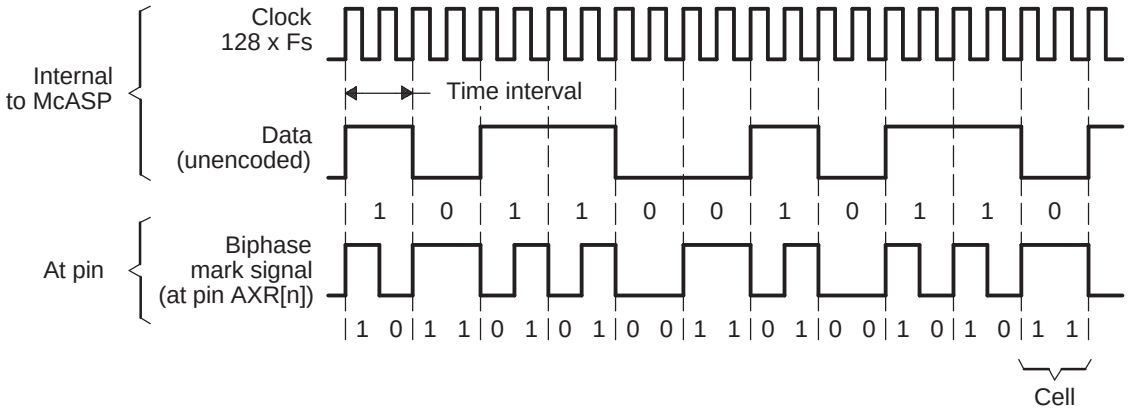


Table 23-1. Biphase-Mark Encoder

Data (Unencoded)	Previous State at Pin AXR[n]	BMC-Encoded Cell Output at AXR[n]
0	0	11
0	1	00
1	0	10
1	1	01

Copyright © 2013–2016, Texas Instruments Incorporated

23.0.18.2.2 Subframe Format

Every audio sample transmitted in a subframe consists of 32 S/PDIF time intervals (or cells), numbered from 0 to 31. Figure 23-10 shows a subframe.

- **Time intervals 0-3** carry one of the three permitted preambles to signify the type of audio sample in the current subframe. The preamble is *not* encoded in BMC format, and therefore the preamble code can contain more than two consecutive 0 or 1 logical states in a row. See Table 23-2.
- **Time intervals 4-27** carry the audio sample word in linear 2s-complement representation. The most-significant bit (MSB) is carried by time interval 27. When a 24-bit coding range is used, the least-significant bit (LSB) is in time interval 4. When a 20-bit coding range is used, time intervals 8-27 carry the audio sample word with the LSB in time interval 8. Time intervals 4-7 may be used for other applications and are designated auxiliary sample bits.
- If the source provides fewer bits than the interface allows (either 20 or 24), the unused LSBs are set to logical 0. For a nonlinear PCM audio application or a data application, the main data field may carry any other information.
- **Time interval 28** carries the validity bit (V) associated with the main data field in the subframe.
- **Time interval 29** carries the user data channel (U) associated with the main data field in the subframe.
- **Time interval 30** carries the channel status information (C) associated with the main data field in the subframe. The channel status indicates if the data in the subframe is digital audio or some other type of data.
- **Time interval 31** carries a parity bit (P) such that time intervals 4-31 carry an even number of 1s and an even number of 0s (even parity). As shown in Table 23-2, the preambles (time intervals 0-3) are also defined with even parity.

Figure 23-10. S/PDIF Subframe Format

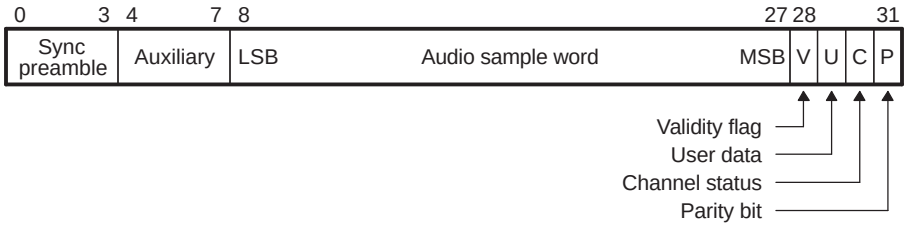


Table 23-2. Preamble Codes

Preamble Code[1]	Previous Logical State	Logical States on pin AXR[n][2]	Description
B (or Z)	0	1110 1000	Start of a block and subframe 1
M (or X)	0	1110 0010	Subframe 1
W (or Y)	0	1110 0100	Subframe 2

(1) Historically, preamble codes are referred to as B, M, W. For use in professional applications, preambles are referred to as Z, X, Y, respectively.

(2) The preamble is not BMC encoded. Each logical state is synchronized to the serial clock. These 8 logical states make up time slots (cells) 0 to 3 in the S/PDIF stream.

As shown in Table 23-2, the McASP DIT only generates one polarity of preambles and it assumes the previous logical state to be 0. This is because the McASP assures an even-polarity encoding scheme when transmitting in DIT mode. If an underrun condition occurs, the DIT resynchronizes to the correct logic level on the AXR[n] pin before continuing with the next transmission.

Copyright © 2013–2016, Texas Instruments Incorporated

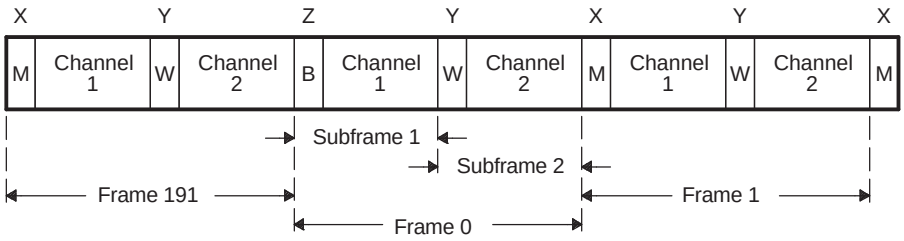
23.0.18.2.3 Frame Format

An S/PDIF frame is composed of two subframes (Figure 23-11). For linear coded audio applications, the rate of frame transmission normally corresponds exactly to the source sampling frequency f_s. The S/PDIF format clock rate is therefore 128 × f_s (128 = 32 cells/subframe × 2 clocks/cell × 2 subframes/sample). For example, for an S/PDIF stream at a 192 kHz sampling frequency, the serial clock is 128 × 192 kHz = 24.58 MHz.

In 2-channel operation mode, the samples taken from both channels are transmitted by time multiplexing in consecutive subframes. Both subframes contain valid data. The first subframe (**left** or **A** channel in stereophonic operation and **primary** channel in monophonic operation) normally starts with preamble M. However, the preamble of the first subframe changes to preamble B once every 192 frames to identify the start of the block structure used to organize the channel status information. The second subframe (**right** or **B** channel in stereophonic operation and **secondary** channel in monophonic operation) always starts with preamble W.

In single-channel operation mode in a professional application, the frame format is the same as in the 2-channel mode. Data is carried in the first subframe and may be duplicated in the second subframe. If the second subframe is not carrying duplicate data, time slot 28 (validity bit) is set to logical 1.

Figure 23-11. S/PDIF Frame Format



Copyright © 2013–2016, Texas Instruments Incorporated

23.0.19 Definition of Terms

The serial bit stream transmitted or received by the McASP is a long sequence of 1s and 0s, either output or input on one of the audio transmit/receive pins (AXR[n]). However, the sequence has a hierarchical organization that can be described in terms of frames of data, slots, words, and bits.

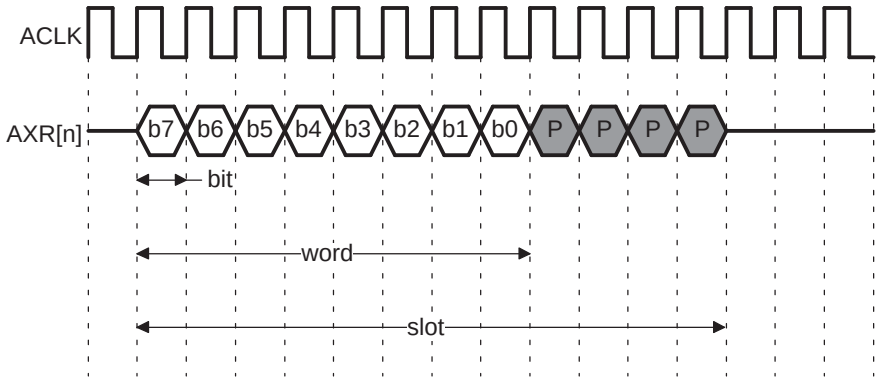
A basic synchronous serial interface consists of three important components: clock, frame sync, and data. Figure 23-12 shows two of the three basic components—the clock (ACLK) and the data (AXR[n]). Figure 23-12 does not specify whether the clock is for transmit (ACLKX) or receive (ACLKR) because the definitions of terms apply to both receive and transmit interfaces. In operation, the transmitter uses ACLKX as the serial clock, and the receiver uses ACLKR as the serial clock. Optionally, the receiver can use ACLKX as the serial clock when the transmitter and receiver of the McASP are configured to operate synchronously.

Bit A bit is the smallest entity in the serial data stream. The beginning and end of each bit is marked by an edge of the serial clock. The duration of a bit is a serial clock period. A 1 is represented by a logic high on the AXR[n] pin for the entire duration of the bit. A 0 is represented by a logic low on the AXR[n] pin for the entire duration of the bit.

Word A word is a group of bits that make up the data being transferred between the CPU and the external device. Figure 23-12 shows an 8-bit word.

Slot A slot consists of the bits that make up the word, and may consist of additional bits used to pad the word to a convenient number of bits for the interface between the CPU and the external device. In Figure 23-12, the audio data consists of only 8 bits of useful data (8-bit word), but it is padded with 4 zeros (12-bit slot) to satisfy the desired protocol in interfacing to an external device. Within a slot, the bits may be shifted in/out of the McASP on the AXR[n] pin either MSB or LSB first. When the word size is smaller than the slot size, the word may be aligned to the left (beginning) of the slot or to the right (end) of the slot. The additional bits in the slot not belonging to the word may be padded with 0, 1, or with one of the bits (the MSB or the LSB typically) from the data word. These options are shown in Figure 23-13.

Figure 23-12. Definition of Bit, Word, and Slot



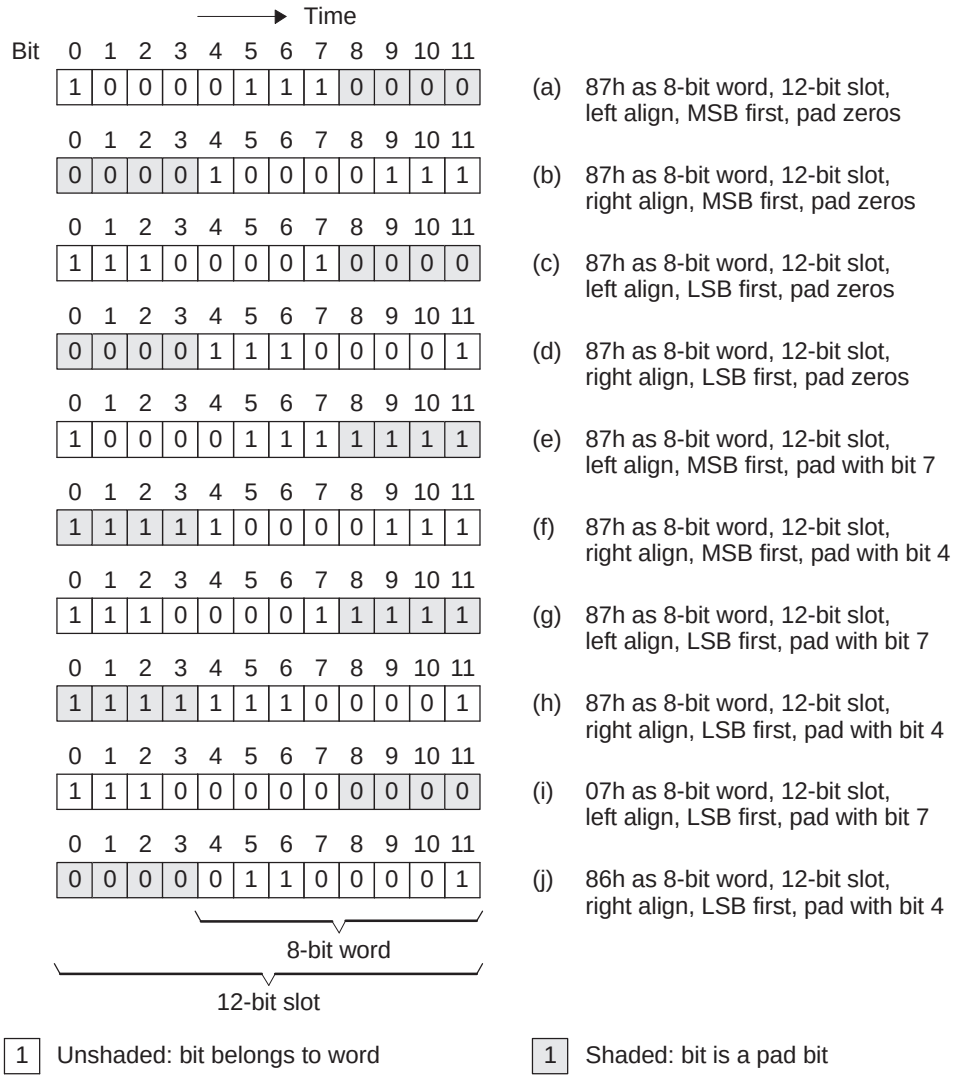
(1) b7:b0 - bits. Bits b7 to b0 form a word.

(2) P - pad bits. Bits b7 to b0, together with the four pad bits, form a slot.

(3) In this example, the data is transmitted MSB first, left aligned.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 23-13. Bit Order and Word Alignment Within a Slot Examples



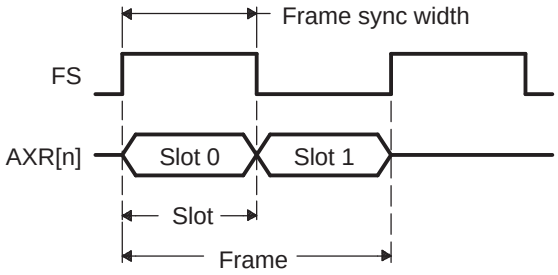
Copyright © 2013–2016, Texas Instruments Incorporated

The third basic element of a synchronous serial interface is the frame synchronization signal, also referred to as frame sync in this chapter.

Frame A frame contains one or multiple slots, as determined by the desired protocol. Figure 23-14 shows an example frame of data and the frame definitions. Figure 23-14 does not specify whether the frame sync (FS) is for transmit (AFSX) or receive (AFSR) because the definitions of terms apply to both receive and transmit interfaces. In operation, the transmitter uses AFSX and the receiver uses AFSR. Optionally, the receiver can use AFSX as the frame sync when the transmitter and receiver of the McASP are configured to operate synchronously.

This section only shows the generic definition of the frame sync. See Section 23.0.18.1, Section 23.0.18.2, and Section 23.0.21.2.1 for details on the frame sync formats required for the different transfer modes and protocols (burst mode, TDM mode and I2S format, DIT mode and S/PDIF format).

Figure 23-14. Definition of Frame and Frame Sync Width



(1) In this example, there are two slots in a frame, and FS duration of slot length is shown.

Other terms used throughout this chapter:

TDM	Time-division multiplexed. See Section 23.0.18.1 for details on the TDM protocol.
DIR	Digital audio interface receive. The McASP does not natively support receiving in the S/PDIF format. The McASP supports I2S format output by an external DIR device.
DIT	Digital audio interface transmit. The McASP supports transmitting in S/PDIF format on up to all data pins configured as outputs.
I2S	Inter-IC Sound protocol, commonly used on audio interfaces. The McASP supports the I2S protocol as part of the TDM mode (when configured as a 2-slot frame).
Slot or Time Slot	For TDM format, the term time slot is interchangeable with the term slot defined in this section. For DIT format, a McASP time slot corresponds to a DIT subframe.

Copyright © 2013–2016, Texas Instruments Incorporated

Architecture

23.0.20 Overview

Figure 23-1 shows the major blocks of the McASP. The McASP has independent receive/transmit clock generators and frame sync generators, error-checking logic, and up to 16 serial data pins. See your device-specific data manual for the number of data pins available on your device.

All the McASP pins on the device may be individually programmed as general-purpose I/O (GPIO) if they are not used for serial port functions.

The McASP includes the following pins:

- Serializers
  - Data pins AXR[n]: Up to sixteen per McASP
- Transmit clock generator:
  - AHCLKX: McASP transmit high-frequency master clock
  - ACLKX: McASP transmit bit clock
- Transmit Frame Sync Generator
  - AFSX: McASP transmit frame sync or left/right clock (LRCLK)
- Receive clock generator:
  - AHCLKR: McASP receive high-frequency master clock
  - ACLKR: McASP receive bit clock
- Receive Frame Sync Generator
  - AFSR: McASP receive frame sync or left/right clock (LRCLK)
- Mute in/out:
  - AMUTEIN: McASP mute input (from external device)
  - AMUTE: McASP mute output
  - Data pins AXR[n]

23.0.21 Clock and Frame Sync Generators

The McASP clock generators are able to produce two independent clock zones: transmit and receive clock zones. The serial clock generators may be programmed independently for the transmit section and the receive section, and may be completely asynchronous to each other. The serial clock (clock at the bit rate) may be sourced:

- **Internally** - by passing through two clock dividers off the internal clock source (AUXCLK)
- **Externally** - directly from ACLKR/X pin
- **Mixed** - an external high-frequency clock is input to the McASP on either the AHCLKX or AHCLKR pins, and divided down to produce the bit rate clock

In the internal/mixed cases, the bit rate clock is generated internally and should be driven out on the ACLKX (for transmit) or ACLKR (for receive) pins. In the internal case, an internally-generated high-frequency clock may be driven out onto the AHCLKX or AHCLKR pins to serve as a reference clock for other components in the system.

The McASP requires a minimum of a bit clock and a frame sync to operate, and provides the capability to reference these clocks from an external high-frequency master clock. In DIT mode, it is possible to use only internally-generated clocks and frame syncs.

Copyright © 2013–2016, Texas Instruments Incorporated

23.0.21.1 Transmit Clock

The transmit bit clock, ACLKX, (Figure 23-15) may be either externally sourced from the ACLKX pin or internally generated, as selected by the CLKXM bit. If internally generated (CLKXM = 1), the clock is divided down by a programmable bit clock divider (CLKXDIV) from the transmit high-frequency master clock (AHCLKX).

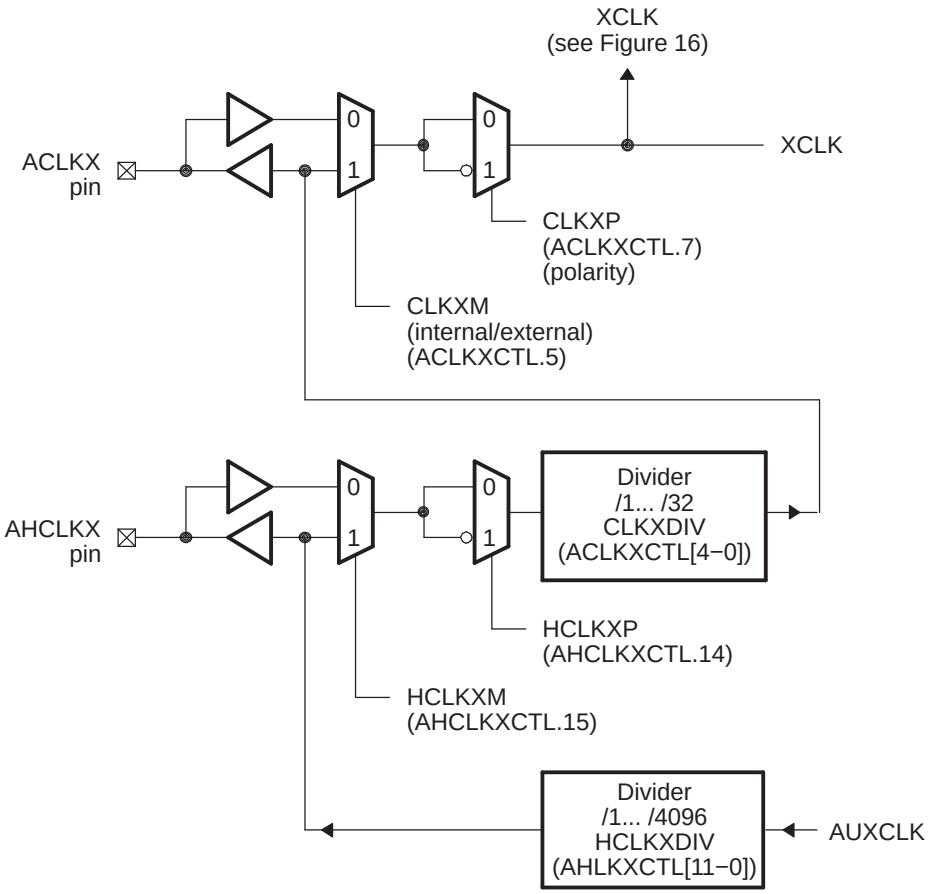
Internally, the McASP always shifts transmit data at the rising edge of the internal transmit clock, XCLK, (Figure 23-15). The CLKXP mux determines if ACLKX needs to be inverted to become XCLK. If CLKXP = 0, the CLKXP mux directly passes ACLKX to XCLK. As a result, the McASP shifts transmit data at the rising edge of ACLKX. If CLKXP = 1, the CLKX mux passes the inverted version of ACLKX to XCLK. As a result, the McASP shifts transmit data at the falling edge of ACLKX.

The transmit high-frequency master clock, AHCLKX, may be either externally sourced from the AHCLKX pin or internally generated, as selected by the HCLKXM bit. If internally generated (HCLKXM = 1), the clock is divided down by a programmable high clock divider (HCLKXDIV) from McASP internal clock source AUXCLK. The transmit high-frequency master clock may be (but is not required to be) output on the AHCLKX pin where it is available to other devices in the system.

The transmit clock configuration is controlled by the following registers:

- ACLKXCTL
- AHCLKXCTL

Figure 23-15. Transmit Clock Generator Block Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

23.0.21.2 Receive Clock

The receiver has a clock generation circuit identical to (but independent of) that of the transmitter. The receive bit clock, ACLKR, (Figure 23-16) may be either externally sourced from the ACLKR pin or internally generated, as selected by the CLKRM bit. If internally generated (CLKRM = 1), the clock is divided down by a programmable divider (CLKRDIV) from the receive high-frequency master clock (AHCLKR). Regardless if ACLKR is either internally generated or externally sourced, polarity of the clock may be programmed (CLKRP) to be either rising or falling edge.

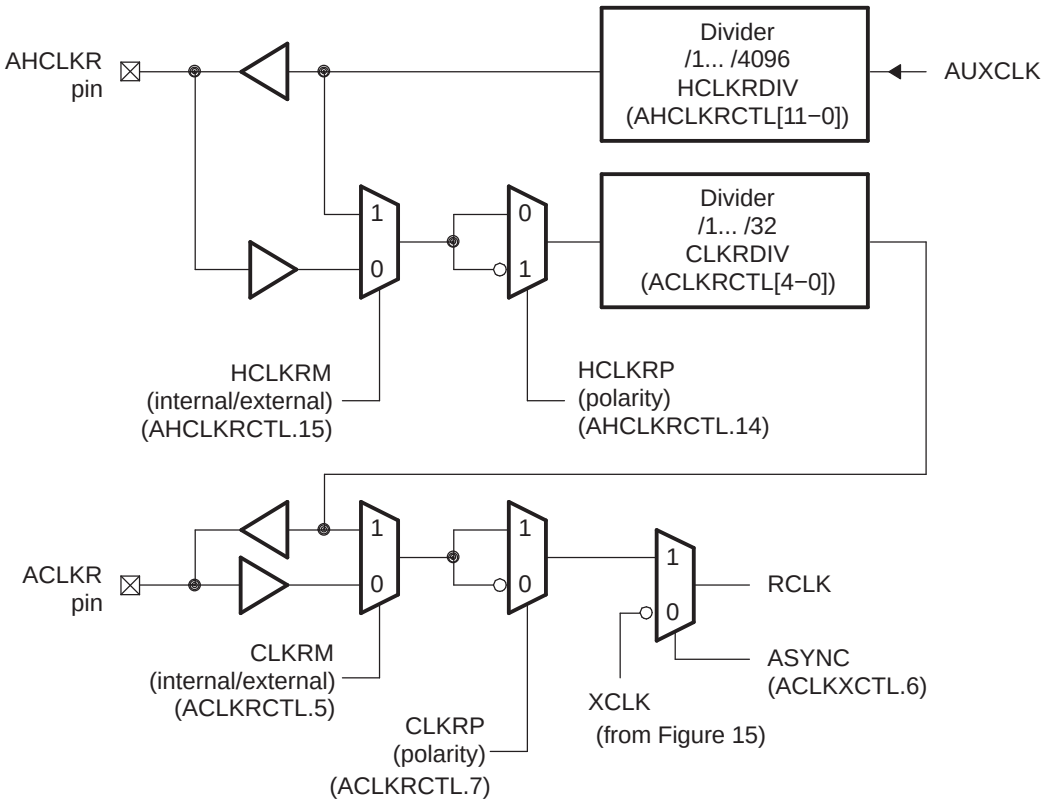
The receive high-frequency master clock, AHCLKR, may be either externally sourced from the AHCLKR pin or internally generated, as selected by the HCLKRM bit. If internally generated (HCLKRM = 1), the clock is divided down by a programmable divider (HCLKRDIV) from AUXCLK. The receive high-frequency master clock may be (but is not required to be) output on the AHCLKR pin where it is available to other devices in the system. Regardless if AHCLKR is either internally generated or externally sourced, polarity of the high-frequency clock may be programmed (HCLKRP) to be either rising or falling edge.

The receiver also has the option to operate synchronously from the ACLKX and AFSX signals. This is achieved when the ASYNC bit in the transmit clock control register (ACLKXCTL) is cleared to 0. See Section 23.0.21.1.5 for details on McASP operation when ACLKXCTL.ASYNC = 0.

The receive clock configuration is controlled by the following registers:

- ACLKRCTL
- AHCLKRCTL

Figure 23-16. Receive Clock Generator Block Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

23.0.21.3 Frame Sync Generator

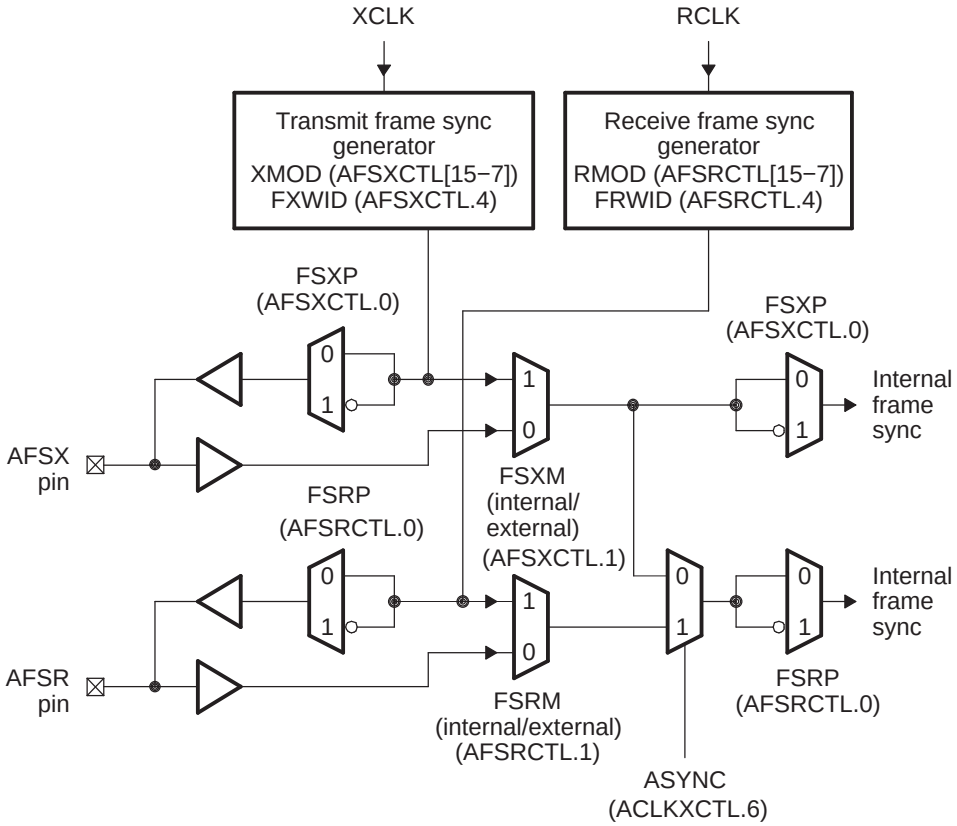
There are two different modes for frame sync: burst and TDM. A block diagram of the frame sync generator is shown in Figure 23-17. The frame sync options are programmed by the receive and transmit frame sync control registers (AFSRCTL and AFSXCTL). The options are:

- Internally-generated or externally-generated
- Frame sync polarity: rising edge or falling edge
- Frame sync width: single bit or single word
- Bit delay: 0, 1, or 2 cycles before the first data bit

The transmit frame sync pin is AFSX and the receive frame sync pin is AFSR. A typical usage for these pins is to carry the left/right clock (LRCLK) signal when transmitting and receiving stereo data.

Regardless if the AFSX/AFSR is internally generated or externally sourced, the polarity of AFSX/AFSR is determined by FSXP/FSRP, respectively, to be either rising or falling edge. If FSXP/FSRP = 0, the frame sync polarity is rising edge. If FSXP/FSRP = 1, the frame sync polarity is falling edge.

Figure 23-17. Frame Sync Generator Block Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

23.0.21.4 Clocking Examples

Some examples of processes using the McASP clocking and frame flexibility are:

- Receive data from a DVD at 48 kHz, but output up-sampled or decoded audio at 96 kHz or 192 kHz. This could be accomplished by inputting a high-frequency master clock (for example, 512 × receive FS), receiving with an internally-generated bit clock ratio of divide-by-8, and transmitting with an internally-generated bit clock ratio of divide-by-4 or divide-by-2.
- Transmit/receive data based on one sample rate (for example, 44.1 kHz), and transmit/receive data at a different sample rate (for example, 48 kHz).

General Architecture

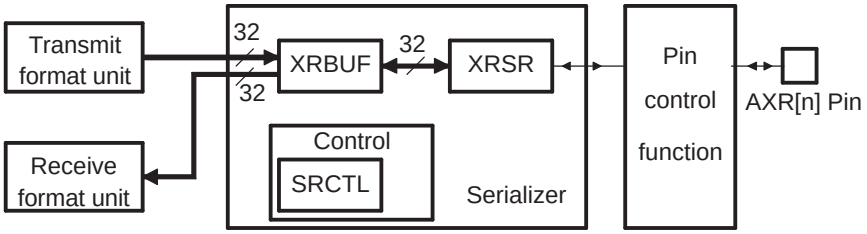
23.0.21.1 Serializers

The serializers take care of shifting serial data in and out of the McASP. Each serializer consists of a shift register (XRSR), a data buffer (XRBUF), a control register (SRCTL), and logic to support the data alignment options of the McASP. For each serializer, there is a dedicated serial data pin (AXR[n]) and a dedicated control register (SRCTL[n]). The control register allows the serializer to be configured as a transmitter, receiver, or as inactive. When configured as a transmitter the serializer shifts out data to the serial data pin AXR[n]. When configured as a receiver, the serializer shifts in data from the AXR[n] pin. The serializer is clocked from the transmit/receive section clock (ACLKX/ACLKR) if configured to transmit/receive respectively.

All serializers that are configured to transmit operate in lock-step. Similarly, all serializers that are configured to receive also operate in lock-step. This means that at most there are two zones per McASP, one for transmit and one for receive.

Figure 23-18 shows the block diagram of the serializer and its interface to other units within the McASP.

Figure 23-18. Individual Serializer and Connections Within McASP



For receive, data is shifted in through the AXR[n] pin to the shift register XRSR. Once the entire slot of data is collected in the XRSR, the data is copied to the data buffer XRBUF. The data is now ready to be read by the CPU through the RBUF register, which is an alias of the XRBUF for receive. When the CPU reads from the RBUF, the McASP passes the data from RBUF through the receive format unit and returns the formatted data to the CPU.

For transmit, the CPU services the McASP by writing data into the XBUF register, which is an alias of the XRBUF for transmit. The data automatically passes through the transmit format unit before actually reaching the XRBUF in the serializer. The data is then copied from XRBUF to XRSR, and shifted out from the AXR[n] synchronously to the serial clock.

In DIT mode, in addition to the data, the serializer shifts out other DIT-specific information accordingly (preamble, user data, etc.).

The serializer configuration is controlled by SRCTL[n].

Copyright © 2013–2016, Texas Instruments Incorporated

23.0.21.2 Format Unit

The McASP has two data formatting units, one for transmit and one for receive. These units automatically remap the data bits within the transmitted and received words between a natural format for the CPU (such as a Q31 representation) and the required format for the external serial device (such as "I2S format"). During the remapping process, the format unit also can mask off certain bits or perform sign extension.

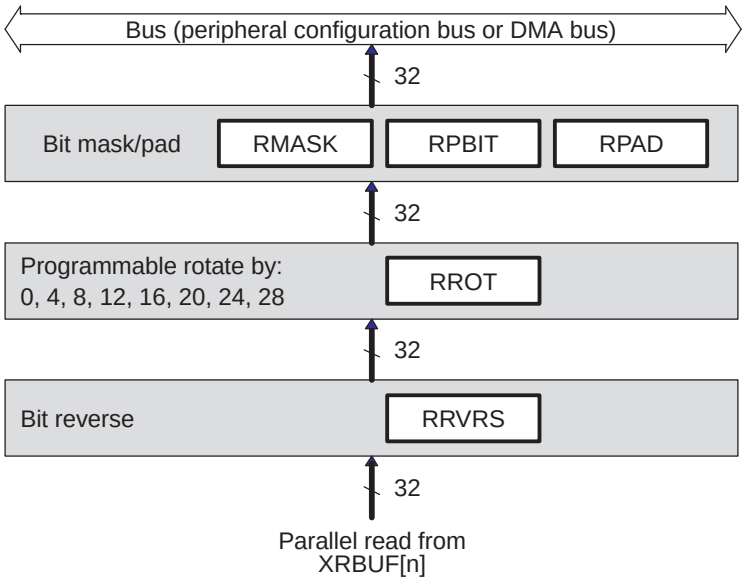
Since all transmitters share the same data formatting unit, the McASP only supports one transmit format at a time. For example, the McASP will not transmit in "I2S format" on serializer 0, while transmitting "Left Justified" on serializer 1. Likewise, the receiver section of the McASP only supports one data format at a time, and this format applies to all receiving serializers. However, the McASP can transmit in one format while receiving in a completely different format.

This formatting unit consists of three stages:

- Bit mask and pad (masks off bits, performs sign extension)
- Rotate right (aligns data within word)
- Bit reversal (selects between MSB first or LSB first)

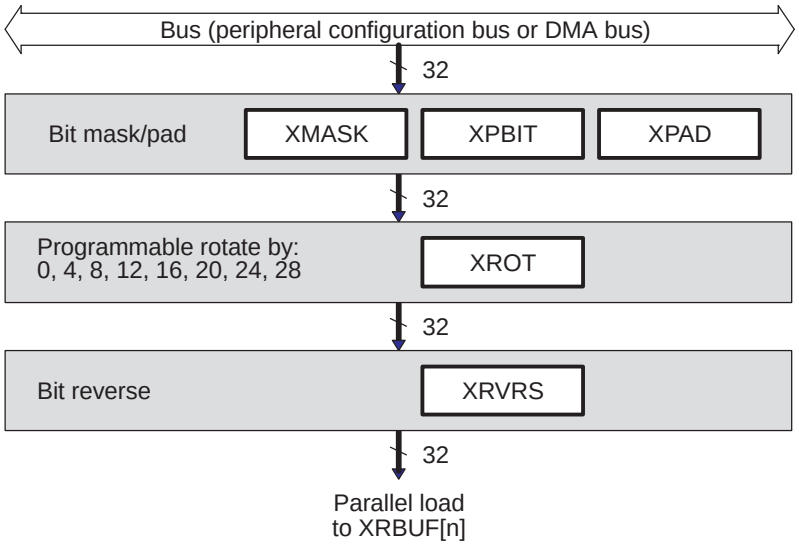
Figure 23-19 shows a block diagram of the receive formatting unit, and Figure 23-20 shows the transmit formatting unit. Note that the order in which data flows through the three stages is different between the transmit and receive formatting units.

Figure 23-19. Receive Format Unit



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 23-20. Transmit Format Unit



The bit mask and pad stage includes a full 32-bit mask register, allowing selected individual bits to either pass through the stage unchanged, or be masked off. The bit mask and pad then pad the value of the masked off bits by inserting either a 0, a 1, or one of the original 32 bits as the pad value. The last option allows for sign-extension when the sign bit is selected to pad the remaining bits.

The rotate right stage performs bitwise rotation by a multiple of 4 bits (between 0 and 28 bits), programmable by the (R/X)FMT register. Note that this is a rotation process, not a shifting process, so bit 0 gets shifted back into bit 31 during the rotation.

The bit reversal stage either passes all 32 bits directly through, or swaps them. This allows for either MSB or LSB first data formats. If bit reversal is not enabled, then the McASP will naturally transmit and receive in an LSB first order.

Finally, note that the (R/X)DATDLY bits in (R/X)FMT also determine the data format. For example, the difference between I2S format and left-justified is determined by the delay between the frame sync edge and the first data bit of a given time slot. For I2S format, (R/X)DATDLY should be set to a 1-bit delay, whereas for left-justified format, it should be set to a 0-bit delay.

The combination of all the options in (R/X)FMT means that the McASP supports a wide variety of data formats, both on the serial data lines, and in the internal CPU representation.

Section 23.0.21.4 provides more detail and specific examples. The examples use internal representation in integer and Q31 notation, but other fractional notations are also possible.

23.0.21.3 State Machine

The receive and transmit sections have independent state machines. Each state machine controls the interactions between the various units in the respective section. In addition, the state machine keeps track of error conditions and serial port status.

No serial transfers can occur until the respective state machine is released from reset. See initialization sequence for details (Section 23.0.21.1).

The receive state machine is controlled by the RFMT register, and it reports the McASP status and error conditions in the RSTAT register. Similarly, the transmit state machine is controlled by the XFMT register, and it reports the McASP status and error conditions in the XSTAT register.

Copyright © 2013–2016, Texas Instruments Incorporated

23.0.21.4 TDM Sequencer

There are separate TDM sequencers for the transmit section and the receive section. Each TDM sequencer keeps track of the slot count. In addition, the TDM sequencer checks the bits of (R/X)TDM and determines if the McASP should receive/transmit in that time slot.

If the McASP should participate (transmit/receive bit is active) in the time slot, the McASP functions normally. If the McASP should not participate (transmit/receive bit is inactive) in the time slot, no transfers between the XRBUF and XRSR registers in the serializer would occur during that time slot. In addition, the serializers programmed as transmitters place their data output pins in a predetermined state (logic low, high, or high impedance) as programmed by each serializer control register (SRCTL). Refer also to Section 23.0.21.2.2 for details on how DMA event or interrupt generations are handled during inactive time slots in TDM mode.

The receive TDM sequencer is controlled by register RTDM and reports current receive slot to RSLOT. The transmit TDM sequencer is controlled by register XTDM and reports current transmit slot to XSLOT.

23.0.21.5 Clock Check Circuit

A common source of error in audio systems is a serial clock failure due to instabilities in the off-chip DIR circuit. To detect a clock error quickly, a clock-check circuit is included in the McASP for both transmit and receive clocks, since both may be sourced from off chip.

The clock check circuit can detect and recover from transmit and receive clock failures. See Section 23.0.21.6.6 for implementation and programming details.

23.0.21.6 Pin Function Control

All McASP pins except AMUTEIN are bidirectional input/output pins. In addition, these bidirectional pins function either as McASP or general-purpose I/O (GPIO) pins. The following registers control the pin functions:

- Pin function register (PFUNC): selects pin to function as McASP or GPIO
- Pin direction register (PDIR): selects pin to be input or output
- Pin data input register (PDIN): shows data input at the pin
- Pin data output register (PDOUT): data to be output at the pin if the pin is configured as GPIO output (PFUNC[n] = 1 and PDIR[n] = 1). Not applicable when the pin is configured as McASP pin (PFUNC[n] = 0).
- Pin data set register (PDSET): alias of PDOUT. Writing a 1 to PDSET[n] sets the respective PDOUT[n] to 1. Writing a 0 has no effect. Applicable only when the pin is configured as GPIO output (PFUNC[n] = 1 and PDIR[n] = 1).
- Pin data clear register (PDCLR): alias of PDOUT. Writing a 1 to PDCLR[n] clears the respective PDOUT[n] to 0. Writing a 0 has no effect. Applicable only when the pin is configured as GPIO output (PFUNC[n] = 1 and PDIR[n] = 1).

See the register descriptions in Section 23.1 for details on the mapping of each McASP pin to the register bits. Figure 23-21 shows the pin control block diagram.

23.0.21.6.1 McASP Pin Control-Transmit and Receive

You must correctly set the McASP GPIO registers PFUNC and PDIR, even when McASP pins are used for their serial port (non-GPIO) function.

Serial port functions include:

- Clock pins (ACLKX, ACLKR, AHCLKX, AHCLKR, AFSX, AFSR) used as clock inputs and outputs
- Serializer data pins (AXR[n]) used to transmit or receive
- AMUTE used as a mute output signal

When using these pins in their serial port function, you must clear PFUNC[n] to 0 for each pin, as opposed to PFUNC[n] = 1, which makes the pin a GPIO.

Copyright © 2013–2016, Texas Instruments Incorporated

Also, certain outputs require PDIR[n] = 1, such as clock pins used as clock outputs, serializer data pins used to transmit, and AMUTE used as mute output.

Clock inputs and serializers configured to receive must have PDIR[n] = 0.

PFUNC and PDIR do not control the AMUTEIN device pin, it is usually tied to a device pin (see your device-specific data manual). If used as a mute input, this pin needs to be configured as an input in the appropriate peripheral.

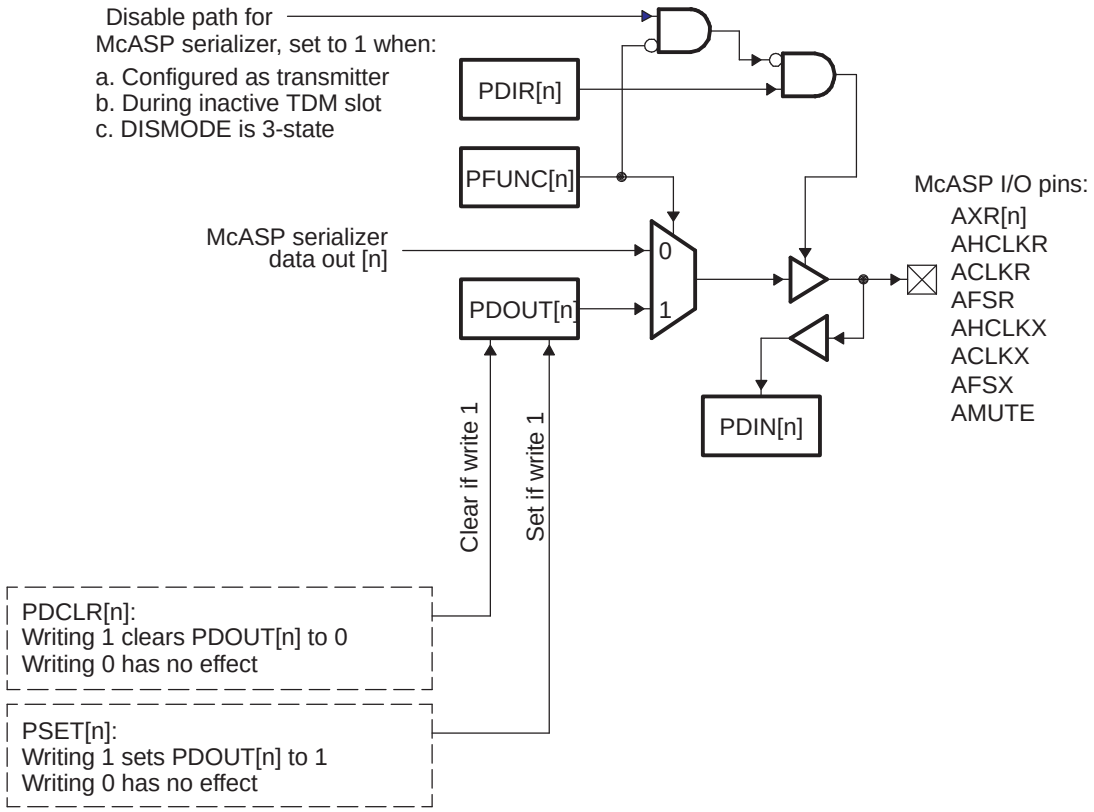
Finally, there is an important advantage to having separate control of pin direction (by PDIR), and the choice of internal versus external clocking (by CLKRM/CLKXM). Depending on the specific device and usage, you might select an external clock (CLKRM = 0), while enabling the internal clock divider, and the clock pin as an output in the PDIR register (PDIR[ACLKR] = 1). In this case, the bit clock is an output (PDIR[ACLKR] = 1) and, therefore, routed to the ACLKR pin. However, because CLKRM = 0, the bit clock is then routed back to the McASP module as an "external" clock source. This may result in less skew between the clock inside the McASP and the clock in the external device, thus producing more balanced setup and hold times for a particular system. As a result, this may allow a higher serial clock rate interface.

23.0.21.6.2 GPIO Pin Control

For GPIO operation, you must set the desired PFUNC[n] to 1 to indicate GPIO function. PDIR[n] must be configured to the desired direction. PDOUT, PDSET, PDCLR control the output value on the pin. PDIN always reflects the state at the pin, regardless of the PDIR and PFUNC setting.

Figure 23-21 and Figure 23-22 display the pin descriptions. The examples that follow (Example 23-1 through Example 23-4) show how the pins can be used as general-purpose input or output pins.

Figure 23-21. McASP I/O Pin Control Block Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 23-22. McASP I/O Pin to Control Register Mapping

31	30	29	28	27	26	25	24
AFSR	AHCLKR	ACLKR	AFSX	AHCLKX	ACLKX	AMUTE	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

23	16
Reserved	
R-0	

15	14	13	12	11	10	9	8
AXR15	AXR14	AXR13	AXR12	AXR11	AXR10	AXR9	AXR8
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
AXR7	AXR6	AXR5	AXR4	AXR3	AXR2	AXR1	AXR0
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

Example 23-1. General-Purpose Input Pin

Because the PDIN register always reflects the state at the pin, you can read the PDIN register to obtain the pin input state. To explicitly set the pin as a general-purpose input pin, you can set the registers as follows:

- PDIR[n] = 0 (input)
- PFUNC[n] = 1 (GPIO function)

Example 23-2. General-Purpose Output Pin—Initialization Using PDOUT

All pins default as inputs. To initialize a pin as output, you should follow this sequence:

1. PDIR[n] = 0 (default as input)
2. PFUNC[n] = 1 (GPIO function)
3. PDOUT[n] = desired output value
4. PDIR[n] = 1 (change to output after desired value is configured in PDOUT[n])

Example 23-3. General-Purpose Output Pin—Change Data from 0 to 1 Using PDSET

If the pin is already configured as a general-purpose output pin driving a 0, and you want to change the output from 0 to 1, the recommended method is to use the PDSET register instead of the PDOUT register. This is because writing to the PDSET register only affects pin(s) in concern. To change a pin from 0 to 1:

- Set PDSET[n]. This sets the respective PDOUT[n].

Example 23-4. General-Purpose Output Pin—Change Data from 1 to 0 Using PDCLR

If the pin is already configured as a general-purpose output pin driving a 1, and you want to change the output from 1 to 0, the recommended method is to use the PDCLR register instead of the PDOUT register. This is because writing to the PDCLR register only affects pin(s) in concern. To change a pin from 1 to 0:

- Set PDCLR[n]. This clears the respective PDOUT[n].

Copyright © 2013–2016, Texas Instruments Incorporated

McASP Audio FIFO (AFIFO)

The McASP Audio FIFO (AFIFO) provides additional data buffering for the McASP. The time it takes the host CPU or DMA controller to respond to DMA requests from the McASP may vary; the additional buffering provided by the AFIFO allows greater tolerance to such variations.

For convenience, the AFIFO is treated here as a block between McASP and the host/DMA controller (see Figure 23-1). Details on configuration of the AFIFO are provided in McASP Audio FIFO (AFIFO).

Operation

This section discusses the operation of the McASP.

23.0.21.1 Setup and Initialization

This section discusses steps necessary to use the McASP module.

23.0.21.1.1 Considerations When Using a McASP

The following is a list of things to be considered for systems using a McASP:

23.0.21.1.1.1 Clocks

For each receive and transmit section:

- External or internal generated bit clock and high frequency clock?
- If internally generated, what is the bit clock speed and the high frequency clock speed?
- Clock polarity?
- External or internal generated frame sync?
- If internally generated, what is frame sync speed?
- Frame sync polarity?
- Frame sync width?
- Transmit and receive sync or asynchronous?

23.0.21.1.1.2 Data Pins

For each pin of each McASP:

- McASP or GPIO?
- Input or output?

23.0.21.1.1.3 Data Format

For each transmit and receive data:

- Internal numeric representation (integer, Q31 fraction)?
- I2S or DIT (transmit only)?
- Time slot delay (0, 1, or 2 bit)?
- Alignment (left or right)?
- Order (MSB first, LSB first)?
- Pad (if yes, pad with what value)?
- Slot size?
- Rotate?
- Mask?

Copyright © 2013–2016, Texas Instruments Incorporated

23.0.21.1.1.4 Data Transfers

- Internal: DMA or CPU?
- External: TDM or burst?
- Bus: peripheral configuration bus or DMA bus?

23.0.21.1.2 Transmit/Receive Section Initialization

You must follow the following steps to properly configure the McASP. If external clocks are used, they should be present prior to the following initialization steps.

1. Reset McASP to default values by setting GBLCTL = 0.
2. Configure McASP Audio FIFO. Recall that the Write FIFO and Read FIFO are enabled/disabled independently.
   (a) Write FIFO:
   - If the Write FIFO will not be enabled, verify that WFIFOCTL.WENA is cleared to 0 (the default value).
   - If the Write FIFO will be enabled, configure WFIFOCTL. Note that WFIFOCTL.WENA should not be set to 1 (enabled) until the other bitfields in this register are configured.

   (b) Read FIFO:
   - If the Read FIFO will not be enabled, verify that RFIFOCTL.RENA is cleared to 0 (the default value).
   - If the Read FIFO will be enabled, configure RFIFOCTL. Note that RFIFOCTL.RENA should not be set to 1 (enabled) until the other bitfields in this register are configured.
3. Configure all McASP registers except GBLCTL in the following order:
   (a) Receive registers: RMASK, RFMT, AFSRCTL, ACLKRCTL, AHCLKRCTL, RTDM, RINTCTL, RCLKCHK. If external clocks AHCLKR and/or ACLKR are used, they must be running already for proper synchronization of the GBLCTL register.
   (b) Transmit registers: XMASK, XFMT, AFSXCTL, ACLKXCTL, AHCLKXCTL, XTDM, XINTCTL, XCLKCHK. If external clocks AHCLKX and/or ACLKX are used, they must be running already for proper synchronization of the GBLCTL register.
   (c) Serializer registers: SRCTL[n].
   (d) Global registers: Registers PFUNC, PDIR, DITCTL, DLBCTL, AMUTE. Note that PDIR should only be programmed after the clocks and frames are set up in the steps above. This is because the moment a clock pin is configured as an output in PDIR, the clock pin starts toggling at the rate defined in the corresponding clock control register. Therefore you must ensure that the clock control register is configured appropriately before you set the pin to be an output. A similar argument applies to the frame sync pins.
   (e) DIT registers: For DIT mode operation, set up registers DITCSRA[n], DITCSRB[n], DITUDRA[n], and DITUDRB[n].
4. Start the respective high-frequency serial clocks AHCLKX and/or AHCLKR. This step is necessary even if external high-frequency serial clocks are used:
   (a) Take the respective internal high-frequency serial clock divider(s) out of reset by setting the RHCLKRST bit for the receiver and/or the XHCLKRST bit for the transmitter in GBLCTL. All other bits in GBLCTL should be held at 0.
   (b) Read back from GBLCTL to ensure the bit(s) to which you wrote are successfully latched in GBLCTL before you proceed.

Copyright © 2013–2016, Texas Instruments Incorporated

5. Start the respective serial clocks ACLKX and/or ACLKR. This step can be skipped if external serial clocks are used and they are running:
   (a) Take the respective internal serial clock divider(s) out of reset by setting the RCLKRST bit for the receiver and/or the XCLKRST bit for the transmitter in GBLCTL. All other bits in GBLCTL should be left at the previous state.
   (b) Read back from GBLCTL to ensure the bit(s) to which you wrote are successfully latched in GBLCTL before you proceed.
6. Setup data acquisition as required:
   (a) If DMA is used to service the McASP, set up data acquisition as desired and start the DMA in this step, before the McASP is taken out of reset.
   (b) If CPU interrupt is used to service the McASP, enable the transmit and/ or receive interrupt as required.
   (c) If CPU polling is used to service the McASP, no action is required in this step.
7. Activate serializers.
   (a) Before starting, clear the respective transmitter and receiver status registers by writing XSTAT = FFFFh and RSTAT = FFFFh.
   (b) Take the respective serializers out of reset by setting the RSRCLR bit for the receiver and/or the XSRCLR bit for the transmitter in GBLCTL. All other bits in GBLCTL should be left at the previous state.
   (c) Read back from GBLCTL to ensure the bit(s) to which you wrote are successfully latched in GBLCTL before you proceed.
8. Verify that all transmit buffers are serviced. Skip this step if the transmitter is not used. Also, skip this step if time slot 0 is selected as inactive (special cases, see Figure 23-24, second waveform). As soon as the transmit serializer is taken out of reset, XDATA in the XSTAT register is set, indicating that XBUF is empty and ready to be serviced. The XDATA status causes a DMA event AXEVT to be generated, and can cause an interrupt AXINT to be generated if it is enabled in the XINTCTL register.
   (a) If DMA is used to service the McASP, the DMA automatically services the McASP upon receiving AXEVT. Before proceeding in this step, you should verify that the XDATA bit in the XSTAT is cleared to 0, indicating that all transmit buffers are already serviced by the DMA.
   (b) If CPU interrupt is used to service the McASP, interrupt service routine is entered upon the AXINT interrupt. The interrupt service routine should service the XBUF registers. Before proceeding in this step, you should verify that the XDATA bit in XSTAT is cleared to 0, indicating that all transmit buffers are already serviced by the CPU.
   (c) If CPU polling is used to service the McASP, the XBUF registers should be written to in this step.

CAUTION

The CPU does not support the emulation suspend signal. Therefore, if a data window is open in the Code Composer Studio™ integrated development environment to observe the XRBUF locations, the emulation read from the XRBUF locations causes an undesirable side effect of clearing the RDATA bit in RSTAT. Furthermore, if you write to the XRBUF through the Code Composer Studio™ integrated development environment, the emulation write to the XRBUF locations causes the XDATA bit in XSTAT to be cleared.

9. Release state machines from reset.
   (a) Take the respective state machine(s) out of reset by setting the RSMRST bit for the receiver and/or the XSMRST bit for the transmitter in GBLCTL. All other bits in GBLCTL should be left at the previous state.
   (b) Read back from GBLCTL to ensure the bit(s) to which you wrote are successfully latched in GBLCTL before you proceed.

Copyright © 2013–2016, Texas Instruments Incorporated

10. Release frame sync generators from reset. Note that it is necessary to release the internal frame sync generators from reset, even if an external frame sync is being used, because the frame sync error detection logic is built into the frame sync generator.
    (a) Take the respective frame sync generator(s) out of reset by setting the RFRST bit for the receiver, and/or the XFRST bit for the transmitter in GBLCTL. All other bits in GBLCTL should be left at the previous state.
    (b) Read back from GBLCTL to ensure the bit(s) to which you wrote are successfully latched in GBLCTL before you proceed.
11. Upon the first frame sync signal, McASP transfers begin. The McASP synchronizes to an edge on the frame sync pin, not the level on the frame sync pin. This makes it easy to release the state machine and frame sync generators from reset.
    (a) For example, if you configure the McASP for a rising edge transmit frame sync, then you do not need to wait for a low level on the frame sync pin before releasing the McASP transmitter state machine and frame sync generators from reset.

23.0.21.1.3 Separate Transmit and Receive Initialization

In many cases, it is desirable to separately initialize the McASP transmitter and receiver. For example, you may delay the initialization of the transmitter until the type of data coming in on the receiver is recognized. Or a change in the incoming data stream on the receiver may necessitate a reinitialization of the transmitter.

In this case, you may still follow the sequence outlined in Section 23.0.21.1.2, but use it for each section (transmit, receive) individually. The GBLCTL register is aliased to RGBLCTL and XGBLCTL to facilitate separate initialization of transmit and receive sections.

Also, make sure that the initialization or reinitialization sequence follows the guidelines in Bits With Restrictions on When They May be Changed.

23.0.21.1.4 Importance of Reading Back GBLCTL

In Section 23.0.21.1.2, steps 4b, 5b, 7c, 9b, and 10b state that GBLCTL should be read back until the bits that were written are successfully latched. This is important, because the transmitter and receiver state machines run off of the respective bit clocks, which are typically about tens to hundreds of times slower than the CPU's internal bus clock. Therefore, it takes many cycles between when the CPU writes to GBLCTL (or RGBLCTL and XGBLCTL), and when the McASP actually recognizes the write operation. If you skip this step, then the McASP may never see the reset bits in the global control registers get asserted and deasserted; resulting in an uninitialized McASP.

Therefore, the logic in McASP has been implemented such that once the CPU writes GBLCTL, RGBLCTL, or XGBLCTL, the resulting write is not visible by reading back GBLCTL until the McASP has recognized the change. This typically requires two bit clocks plus two CPU bus clocks to occur.

Also, if the bit clocks can be completely stopped, any software that polls GBLCTL should be implemented with a time-out. If GBLCTL does not have a time-out, and the bit clock stops, the changes written to GBLCTL will not be reflected until the bit clock restarts.

Finally, please note that while RGBLCTL and XGBLCTL allow separate changing of the receive and transmit halves of GBLCTL, they also immediately reflect the updated value (useful for debug purposes). Only GBLCTL can be used for the read back step.

23.0.21.1.5 Synchronous Transmit and Receive Operation (ASYNC = 0)

When ASYNC = 0 in ACLKXCTL, the transmit and receive sections operate synchronously from the transmit section clock and transmit frame sync signals (Figure 23-15). The receive section may have a different (but compatible in terms of slot size) data format. Note that when ASYNC = 0, XCLK is automatically inverted to produce RCLK (note the inversion on the ASYNC multiplexer as shown in Figure 23-16).

Copyright © 2013–2016, Texas Instruments Incorporated

When ASYNC = 0, the transmit and receive sections must share some common settings, since they both use the same clock and frame sync signals:

- DITEN = 0 in DITCTL (TDM mode is enabled)
- The total number of bits per frame must be the same (that is, RSSZ × RMOD must equal XSSZ × XMOD)
- Both transmit and receive should either be specified as burst or TDM mode, but not mixed
- The settings in ACLKRCTL are irrelevant
- RCLK is an inverted version of XCLK (note the inversion on the multiplexer labeled “ASYNC” shown in Figure 23-16)
- FSXM must match FSRM
- FXWID must match FRWID

For all other settings, the transmit and receive sections may be programmed independently.

23.0.21.1.6 Asynchronous Transmit and Receive Operation (ASYNC = 1)

When ASYNC = 1 in ACLKXCTL, the transmit and receive sections operate completely independently and have separate clock and frame sync signals (Figure 23-15, Figure 23-16, and Figure 23-17). The events generated by each section come asynchronously.

23.0.21.2 Transfer Modes

23.0.21.2.1 Burst Transfer Mode

The McASP supports a burst transfer mode, which is useful for nonaudio data such as passing control information between two CPUs. Burst transfer mode uses a synchronous serial format similar to the TDM mode. The frame sync generation is not periodic or time-driven as in TDM mode, but data driven, and the frame sync is generated for each data word transferred.

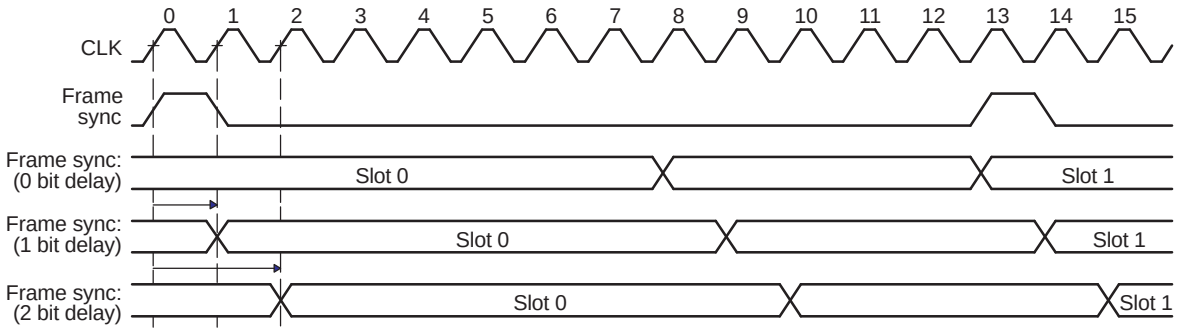
When operating in burst frame sync mode (Figure 23-23), as specified for transmit (XMOD = 0 in AFSXCTL) and receive (RMOD = 0 in AFSRCTL), one slot is shifted for each active edge of the frame sync signal that is recognized. Additional clocks after the slot and before the next frame sync edge are ignored.

In burst frame sync mode, the frame sync delay may be specified as 0, 1, or 2 serial clock cycles. This is the delay between the frame sync active edge and the start of the slot. The frame sync signal lasts for a single bit clock duration (FRWID = 0 in AFSRCTL, FXWID = 0 in AFSXCTL).

For transmit, when generating the transmit frame sync internally, the frame sync begins when the previous transmission has completed and when all the XBUF*n* (for every serializer set to operate as a transmitter) has been updated with new data.

For receive, when generating the receive frame sync internally, frame sync begins when the previous transmission has completed and when all the RBUF*n* (for every serializer set to operate as a receiver) has been read.

Figure 23-23. Burst Frame Sync Mode



Copyright © 2013–2016, Texas Instruments Incorporated

The control registers must be configured as follows for the burst transfer mode. The burst mode specific bit fields are in bold face:

- PFUNC: The clock, frame, data pins must be configured for McASP function.
- PDIR: The clock, frame, data pins must be configured to the direction desired.
- PDOUT, PDIN, PDSET, PDCLR: Not applicable. Leave at default.
- GBLCTL: Follow the initialization sequence in Section 23.0.21.1.2 to configure this register.
- AMUTE: Not applicable. Leave at default.
- DLBCTL: If loopback mode is desired, configure this register according to Section 23.0.21.7, otherwise leave this register at default.
- DITCTL: DITEN must be left at default 0 to select non-DIT mode. Leave the register at default.
- RMASK/XMASK: Mask desired bits according to Section 23.0.21.2 and Section 23.0.21.4.
- RFMT/XFMT: Program all fields according to data format desired. See Section 23.0.21.4.
- AFSRCTL/AFSXCTL: Clear **RMOD/XMOD** bits to 0 to indicate burst mode. Clear **FRWID/FXWID** bits to 0 for single bit frame sync duration. Configure other fields as desired.
- ACLKRCTL/ACLKXCTL: Program all fields according to bit clock desired. See Section 23.0.21.
- AHCLKRCTL/AHCLKXCTL: Program all fields according to high-frequency clock desired. See Section 23.0.21.
- RTDM/XTDM: Program RTDMS0/XTDMS0 to 1 to indicate one active slot only. Leave other fields at default.
- RINTCTL/XINTCTL: Program all fields according to interrupts desired.
- RCLKCHK/XCLKCHK: Not applicable. Leave at default.
- SRCTLn: Program SRMOD to inactive/transmitter/receiver as desired. DISMOD is not applicable and should be left at default.
- DITCSRA[n], DITCSRB[n], DITUDRA[n], DITUDRB[n]: Not applicable. Leave at default.

Copyright © 2013–2016, Texas Instruments Incorporated

23.0.21.2.2 Time-Division Multiplexed (TDM) Transfer Mode

The McASP time-division multiplexed (TDM) transfer mode supports the TDM format discussed in Section 23.0.18.1.

Transmitting data in the TDM transfer mode requires a minimum set of pins:

- ACLKX - transmit bit clock
- AFSX - transmit frame sync (or commonly called left/right clock)
- One or more serial data pins, AXR[n], whose serializers have been configured to transmit

The transmitter has the option to receive the ACLKX bit clock as an input, or to generate the ACLKX bit clock by dividing down the AHCLKX high-frequency master clock. The transmitter can either generate AHCLKX internally or receive AHCLKX as an input. See Section 23.0.21.1.

Similarly, to receive data in the TDM transfer mode requires a minimum set of pins:

- ACLKR - receive bit clock
- AFSR - receive frame sync (or commonly called left/right clock)
- One or more serial data pins, AXR[n], whose serializers have been configured to receive

The receiver has the option to receive the ACLKR bit clock as an input or to generate the ACLKR bit clock by dividing down the AHCLKR high-frequency master clock. The receiver can either generate AHCLKR internally or receive AHCLKR as an input. See Section 23.0.21.2 and Section 23.0.21.3.

The control registers must be configured as follows for the TDM mode. The TDM mode specific bit fields are in bold face:

- PFUNC: The clock, frame, data pins must be configured for McASP function.
- PDIR: The clock, frame, data pins must be configured to the direction desired.
- PDOUT, PDIN, PDSET, PDCLR: Not applicable. Leave at default.
- GBLCTL: Follow the initialization sequence in Section 23.0.21.1.2 to configure this register.
- AMUTE: Program all fields according to mute control desired.
- DLBCTL: If loopback mode is desired, configure this register according to Section 23.0.21.7, otherwise leave this register at default.
- DITCTL: DITEN must be left at default 0 to select TDM mode. Leave the register at default.
- RMASK/XMASK: Mask desired bits according to Section 23.0.21.2 and Section 23.0.21.4.
- RFMT/XFMT: Program all fields according to data format desired. See Section 23.0.21.4.
- AFSRCTL/AFSXCTL: Set **RMOD/XMOD** bits to 2-32 for TDM mode. Configure other fields as desired.
- ACLKRCTL/ACLKXCTL: Program all fields according to bit clock desired. See Section 23.0.21.
- AHCLKRCTL/AHCLKXCTL: Program all fields according to high-frequency clock desired. See Section 23.0.21.
- RTDM/XTDM: Program all fields according to the time slot characteristics desired.
- RINTCTL/XINTCTL: Program all fields according to interrupts desired.
- RCLKCHK/XCLKCHK: Program all fields according to clock checking desired.
- SRCTLn: Program all fields according to serializer operation desired.
- DITCSRA[n], DITCSRB[n], DITUDRA[n], DITUDRB[n]: Not applicable. Leave at default.

23.0.21.2.2.1 TDM Time Slots

TDM mode on the McASP can extend to support multiprocessor applications, with up to 32 time slots per frame. For each of the time slots, the McASP may be configured to participate or to be inactive by configuring XTDM and/or RTDM (this allows multiple CPUs to communicate on the same TDM serial bus).

The TDM sequencer (separate ones for transmit and receive) functions in this mode. The TDM sequencer counts the slots beginning with the frame sync. For each slot, the TDM sequencer checks the respective bit in either XTDM or RTDM to determine if the McASP should transmit/receive in that time slot.

Copyright © 2013–2016, Texas Instruments Incorporated

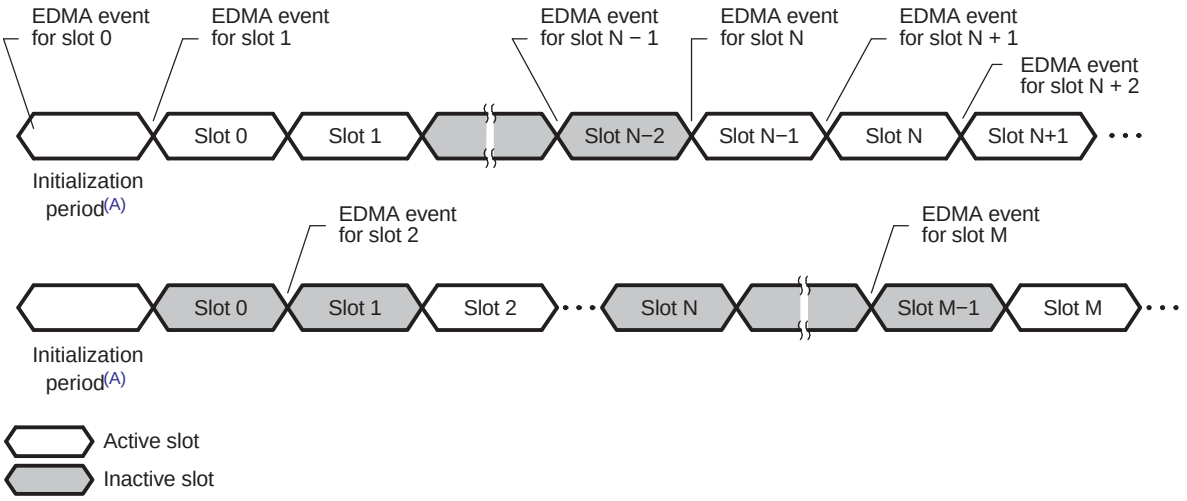
If the transmit/receive bit is active, the McASP functions normally during that time slot; otherwise, the McASP is inactive during that time slot; no update to the buffer occurs, and no event is generated. Transmit pins are automatically set to a high-impedance state, 0, or 1 during that slot, as determined by bit DISMOD in SRCTL[n].

Figure 23-24 shows when the transmit DMA event AXEVT is generated. See Section 23.0.21.3.1 for details on data ready and the initialization period indication. The transmit DMA event for an active time slot (slot N) is generated during the previous time slot (slot N - 1), regardless if the previous time slot (slot N - 1) is active or inactive.

During an active transmit time slot (slot N), if the next time slot (slot N + 1) is configured to be active, the copy from XRBUF[n] to XRSR[n] generates the DMA event for time slot N + 1. If the next time slot (slot N + 1) is configured to be inactive, then the DMA event will be delayed to time slot M - 1. In this case, slot M is the next active time slot. The DMA event for time slot M is generated during the first bit time of slot M - 1.

The receive DMA request generation does not need this capability, since the receive DMA event is generated after data is received in the buffer (looks back in time). If a time slot is disabled, then no data is copied to the buffer for that time slot and no DMA event is generated.

Figure 23-24. Transmit DMA Event (AXEVT) Generation in TDM Time Slots



A See Section 23.0.21.1.2, step 7a.

23.0.21.2.2.2 Special 384 Slot TDM Mode for Connection to External DIR

The McASP receiver also supports a 384 time slot TDM mode (DIR mode), to support S/PDIF, AES-3, IEC-60958 receiver ICs whose natural block (block corresponds to McASP frame) size is 384 samples. The advantage to using the 384 time slot TDM mode is that interrupts may be generated synchronous to the S/PDIF, AES-3, IEC-60958, such as the last slot interrupt.

The receive TDM time slot register (RTDM) should be programmed to all 1s during reception of a DIR block. Other TDM functionalities (for example, inactive slots) are not supported (only the slot counter counts the 384 subframes in a block).

To receive data in the DIR mode, the following pins are typically needed:

- ACLKR - receive bit clock.
- AFSR - receive frame sync (or commonly called left/right clock). In this mode, AFSR should be connected to a DIR which outputs a start of block signal, instead of LRCLK.
- One or more serial data pins, AXR[n], whose serializers have been configured to receive.

For this special DIR mode, the control registers can be configured just as for TDM mode, except set RMOD in AFSRCTL to 384 to receive 384 time slots.

Copyright © 2013–2016, Texas Instruments Incorporated

23.0.21.2.3 Digital Audio Interface Transmit (DIT) Transfer Mode

In addition to the TDM and burst transfer modes, which are suitable for transmitting audio data between ICs inside the same system, the digital audio interface transmit (DIT) transfer mode of the McASP also supports transmission of audio data in the S/PDIF, AES-3, or IEC-60958 format. These formats are designed to carry audio data between different systems through an optical or coaxial cable. The DIT mode only applies to serializers configured as transmitters, not receivers. Refer to Section 23.0.18.2 for a description of the S/PDIF format.

23.0.21.2.3.1 Transmit DIT Encoding

The McASP operation in DIT mode is basically identical to the 2 time slot TDM mode, but the data transmitted is output as a biphase mark encoded bit stream, with preamble, channel status, user data, validity, and parity automatically stuffed into the bit stream by the McASP. The McASP includes separate validity bits for even/odd subframes and two 384-bit RAM modules to hold channel status and user data bits.

The transmit TDM time slot register (XTDM) should be programmed to all 1s during DIT mode. TDM functionality is not supported in DIT mode, except that the TDM slot counter counts the DIT subframes.

To transmit data in the DIT mode, the following pins are typically needed:

- AHCLKX - transmit high-frequency master clock
- One or more serial data pins, AXR[n], whose serializers have been configured to transmit

AHCLKX is optional (the internal clock source may be used instead), but if used as a reference, the CPU provides a clock check circuit that continually monitors the AHCLKX input for stability.

If the McASP is configured to transmit in the DIT mode on more than one serial data pin, the bit streams on all pins will be synchronized. In addition, although they will carry unique audio data, they will carry the same channel status, user data, and validity information.

The actual 24-bit audio data must always be in bit positions 23-0 after passing through the first three stages of the transmit format unit.

For left-aligned Q31 data, the following transmit format unit settings process the data into right aligned 24-bit audio data ready for transmission:

- XROT = 010 (rotate right by 8 bits)
- XRVRS = 0 (no bit reversal, LSB first)
- XMASK = FFFF FF00h-FFFF 0000h (depending upon whether 24, 23, 22, 21, 20, 19, 18, 17, or 16 valid audio data bits are present)
- XPAD = 00 (pad extra bits with 0)

For right-aligned data, the following transmit format unit settings process the data into right aligned 24-bit audio data ready for transmission:

- XROT = 000 (rotate right by 0 bits)
- XRVRS = 0 (no bit reversal, LSB first)
- XMASK = 00FF FFFFh to 0000 FFFFh (depending upon whether 24, 23, 22, 21, 20, 19, 18, 17, or 16 valid audio data bits are present)
- XPAD = 00 (pad extra bits with 0)

Copyright © 2013–2016, Texas Instruments Incorporated

23.0.21.2.3.2 Transmit DIT Clock and Frame Sync Generation

The DIT transmitter only works in the following configuration:

- In transmit frame control register (AFSXCTL):
  - Internally-generated transmit frame sync, FSXM = 1
  - Rising-edge frame sync, FSXP = 0
  - Bit-width frame sync, FXWID = 0
  - 384-slot TDM, XMOD = 1 1000 0000b
- In transmit clock control register (ACLKXCTL), ASYNC = 1
- In transmit bitstream format register (XFMT), XSSZ = 1111 (32-bit slot size)

All combinations of AHCLKX and ACLKX are supported.

This is a summary of the register configurations required for DIT mode. The DIT mode specific bit fields are in bold face:

- PFUNC: The data pins must be configured for McASP function. If AHCLKX is used, it must also be configured for McASP function. Other pins can be configured to function as GPIO if desired.
- PDIR: The data pins must be configured as outputs. If AHCLKX is used as an input reference, it should be configured as input. If internal clock source AUXCLK is used as the reference clock, it may be output on the AHCLKX pin by configuring AHCLKX as an output.
- PDOUT, PDIN, PDSET, PDCLR: Not applicable for DIT operation. Leave at default.
- GBLCTL: Follow the initialization sequence in Section 23.0.21.1.2 to configure this register.
- AMUTE: Program all fields according to mute control desired.
- DLBCTL: Not applicable. Loopback is not supported for DIT mode. Leave at default.
- DITCTL: **DITEN** bit must be set to 1 to enable DIT mode. Configure other bits as desired.
- RMASK: Not applicable. Leave at default.
- RFMT: Not applicable. Leave at default.
- AFSRCTL: Not applicable. Leave at default.
- ACLKRCTL: Not applicable. Leave at default.
- AHCLKRCTL: Not applicable. Leave at default.
- RTDM: Not applicable. Leave at default.
- RINTCTL: Not applicable. Leave at default.
- RCLKCHK: Not applicable. Leave at default.
- **XMASK**: Mask desired bits according to the discussion in this section, depending upon left-aligned or right-aligned internal data.
- **XFMT: XDATDLY** = 0. **XRVRS** = 0. **XPAD** = 0. **XPBIT** = default (not applicable). **XSSZ** = Fh (32-bit slot). XBUSEL = configured as desired. **XROT** bit is configured according to the discussion in this section, either 0 or 8-bit rotate.
- **AFSXCTL**: Configure the bits according to the discussion in this section.
- **ACLKXCTL**: **ASYNC** = 1. Program CLKXDIV bits to obtain the bit clock rate desired. Configure CLKXP and CLKXM bits as desired, because CLKX is not actually used in the DIT protocol.
- **AHCLKXCTL**: Program all fields according to high-frequency clock desired.
- **XTDM**: Set to FFFF FFFFh for all active slots for DIT transfers.
- XINTCTL: Program all fields according to interrupts desired.
- XCLKCHK: Program all fields according to clock checking desired.
- SRCTLn: Set **SRMOD** = 1 (transmitter) for the DIT pins. DISMOD field is don't care for DIT mode.
- **DITCSRA[n], DITCSRB[n]**: Program the channel status bits as desired.
- **DITUDRA[n], DITUDRB[n]**: Program the user data bits as desired.

Copyright © 2013–2016, Texas Instruments Incorporated

23.0.21.2.3.3 DIT Channel Status and User Data Register Files

The channel status registers (DITCSRA*n* and DITCSRB*n*) and user data registers (DITUDRA*n* and DITUDRB*n*) are not double buffered. Typically the programmer uses one of the synchronizing interrupts, such as last slot, to create an event at a safe time so the register may be updated. In addition, the CPU reads the transmit TDM slot counter to determine which word of the register is being used.

It is a requirement that the software avoid writing to the word of user data and channel status that are being used to encode the current time slot; otherwise, it will be indeterminate whether the old or new data is used to encode the bitstream.

The DIT subframe format is defined in Section 23.0.18.2.2. The channel status information (C) and user data (U) are defined in these DIT control registers:

- DITCSRA0 to DITCSRA5: The 192 bits in these six registers contain the channel status information for the LEFT channel within each frame.
- DITCSRB0 to DITCSRB5: The 192 bits in these six registers contain the channel status information for the RIGHT channel within each frame.
- DITUDRA0 to DITUDRA5: The 192 bits in these six registers contain the user data information for the LEFT channel within each frame.
- DITUDRB0 to DITUDRB5: The 192 bits in these six registers contain the user data information for the RIGHT channel within each frame.

The S/PDIF block format is shown in Figure 23-11. There are 192 frames within a block (frame 0 to frame 191). Within each frame there are two subframes (subframe 1 and 2 for left and right channels, respectively). The channel status and user data information sent on each subframe is summarized in Table 23-3.

23.0.21.3 Data Transmission and Reception

The CPU services the McASP by writing data to the XBUF register(s) for transmit operations, and by reading data from the RBUF register(s) for receive operations. The McASP sets status flag and notifies the CPU whenever data is ready to be serviced. Section 23.0.21.3.1 discusses data ready status in detail.

The XBUF and RBUF registers can be accessed through one of the two peripheral ports of the device:

- The DMA port: This port is dedicated for data transfers on the device.
- The peripheral configuration port: This port is used for both data transfers and peripheral configuration control on the device.

Section 23.0.21.3.2 and Section 23.0.21.3.3 discuss how to perform transfers through the DMA bus and the peripheral configuration bus.

Either the CPU or the DMA can be used to service the McASP through any of these two peripheral ports. The CPU and DMA usages are discussed in Section 23.0.21.3.4 and Section 23.0.21.3.5.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 23-3. Channel Status and User Data for Each DIT Block

Frame	Subframe	Preamble	Channel Status defined in:	User Data defined in:
Defined by DITCSRA0, DITCSRB0, DITUDRA0, DITUDRB0				
0	1 (L)	B	DITCSRA0[0]	DITUDRA0[0]
0	2 (R)	W	DITCSRB0[0]	DITUDRB0[0]
1	1 (L)	M	DITCSRA0[1]	DITUDRA0[1]
1	2 (R)	W	DITCSRB0[1]	DITUDRB0[1]
2	1 (L)	M	DITCSRA0[2]	DITUDRA0[2]
2	2 (R)	W	DITCSRB0[2]	DITUDRB0[2]
…	…	…	…	…
31	1 (L)	M	DITCSRA0[31]	DITUDRA0[31]
31	2 (R)	W	DITCSRB0[31]	DITUDRB0[31]
Defined by DITCSRA1, DITCSRB1, DITUDRA1, DITUDRB1				
32	1 (L)	M	DITCSRA1[0]	DITUDRA1[0]
32	2 (R)	W	DITCSRB1[0]	DITUDRB1[0]
…	…	…	…	…
63	1 (L)	M	DITCSRA1[31]	DITUDRA1[31]
63	2 (R)	W	DITCSRB1[31]	DITUDRB1[31]
Defined by DITCSRA2, DITCSRB2, DITUDRA2, DITUDRB2				
64	1 (L)	M	DITCSRA2[0]	DITUDRA2[0]
64	2 (R)	W	DITCSRB2[0]	DITUDRB2[0]
…	…	…	…	…
95	1 (L)	M	DITCSRA2[31]	DITUDRA2[31]
95	2 (R)	W	DITCSRB2[31]	DITUDRB2[31]
Defined by DITCSRA3, DITCSRB3, DITUDRA3, DITUDRB3				
96	1 (L)	M	DITCSRA3[0]	DITUDRA3[0]
96	2 (R)	W	DITCSRB3[0]	DITUDRB3[0]
…	…	…	…	…
127	1 (L)	M	DITCSRA3[31]	DITUDRA3[31]
127	2 (R)	W	DITCSRB3[31]	DITUDRB3[31]
Defined by DITCSRA4, DITCSRB4, DITUDRA4, DITUDRB4				
128	1 (L)	M	DITCSRA4[0]	DITUDRA4[0]
128	2 (R)	W	DITCSRB4[0]	DITUDRB4[0]
…	…	…	…	…
159	1 (L)	M	DITCSRA4[31]	DITUDRA4[31]
159	2 (R)	W	DITCSRB4[31]	DITUDRB4[31]
Defined by DITCSRA5, DITCSRB5, DITDRA5, DITUDRB5				
160	1 (L)	M	DITCSRA5[0]	DITUDRA5[0]
160	2 (R)	W	DITCSRB5[0]	DITUDRB5[0]
…	…	…	…	…
191	1 (L)	M	DITCSRA5[31]	DITUDRA5[31]
191	2 (R)	W	DITCSRB5[31]	DITUDRB5[31]

Copyright © 2013–2016, Texas Instruments Incorporated

23.0.21.3.1 Data Ready Status and Event/Interrupt Generation

23.0.21.3.1.1 Transmit Data Ready

The transmit data ready flag XDATA bit in the XSTAT register reflects the status of the XBUF register. The XDATA flag is set when data is transferred from the XRBUF[n] buffers to the XRSR[n] shift registers, indicating that the XBUF is empty and ready to accept new data from the CPU. This flag is cleared when the XDATA bit is written with a 1, or when all the serializers configured as transmitters are written by the CPU.

Whenever XDATA is set, an DMA event AXEVT is automatically generated to notify the DMA of the XBUF empty status. An interrupt AXINT is also generated if XDATA interrupt is enabled in the XINTCTL register (See Section 23.0.21.5.1 for details).

For DMA requests, the McASP does not require XSTAT to be read between DMA events. This means that even if XSTAT already has the XDATA flag set to 1 from a previous request, the next transfer triggers another DMA request.

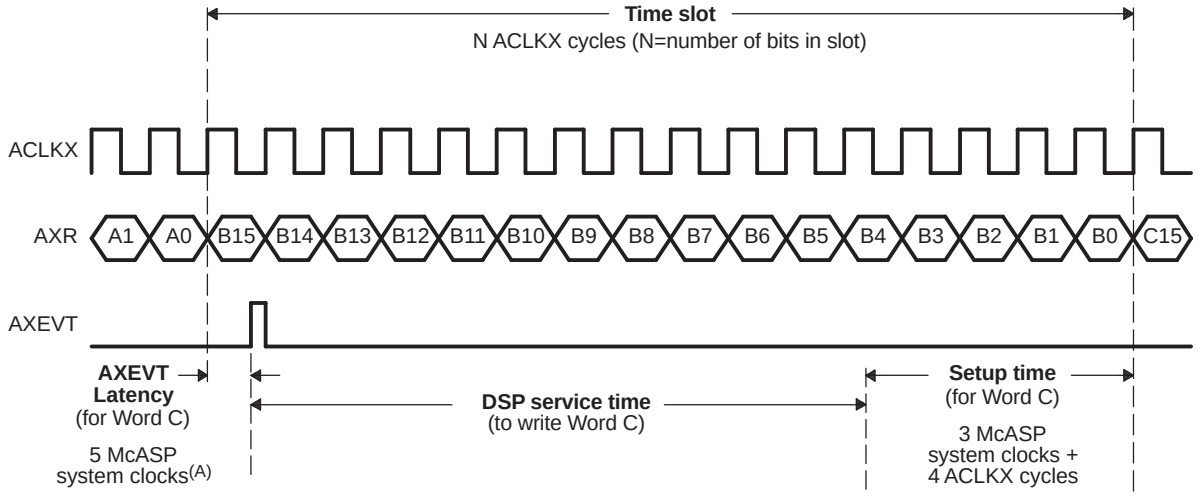
Since all serializers act in lockstep, only one DMA event is generated to indicate that all active transmit serializers are ready to be written to with new data.

Figure 23-25 shows the timing details of when AXEVT is generated at the McASP boundary. In this example, as soon as the last bit (bit A0) of Word A is transmitted, the McASP sets the XDATA flag and generates an AXEVT event. However, it takes up to 5 McASP system clocks (AXEVT Latency) before AXEVT is active at the McASP boundary. Upon AXEVT, the CPU can begin servicing the McASP by writing Word C into the XBUF (DSP Service Time). The CPU must write Word C into the XBUF no later than the setup time required by the McASP (Setup Time).

The maximum DSP Service Time (Figure 23-25) can be calculated as:

DSP Service Time = Time Slot - AXEVT Latency - Setup Time

Figure 23-25. DSP Service Time Upon Transmit DMA Event (AXEVT)



A This is not the same as AUXCLK. The CPU uses SYSCLK2 as the McASP system clock source.

Copyright © 2013–2016, Texas Instruments Incorporated

Example 23-5. DSP Service Time Calculation for Transmit DMA Event (AXEVT)

The following is an example to show how to calculate DSP Service Time. Assume the following setup:

- Device: CPU at 300 MHz
- McASP transmits in I2S format at 192 kHz frame rate. Assume slot size is 32 bit

With the above setup, we obtain the following parameters corresponding to Figure 23-25:

- Calculation of McASP system clock cycle:
  - CPU uses SYSCLK2 as the McASP system clock. It runs at 150 MHz (half of device frequency)
  - Therefore, McASP system clock cycle = 1/150 MHz = 6.7 ns
- Calculation of ACLKX clock cycle:
  - This example has two 32-bit slots per frame, for a total of 64 bits per frame
  - ACLKX clock cycle is (1/192 kHz)/64 = 81.4 ns
- Time Slot between AXEVT events:
  - For I2S format, McASP generates two AXEVT events per 192 kHz frame
  - Therefore, Time Slot between AXEVT events is (1/192 kHz)/2 = 2604 ns
- AXEVT Latency
  = 5 McASP system clocks
  = 6.7 ns × 5 = 33.5 ns
- Setup Time
  = 3 McASP system clocks + 4 ACLKX cycles
  = (6.7 ns × 3) + (81.4 ns × 4)
  = 345.7 ns
- DSP Service Time
  = Time Slot - AXEVT Latency - Setup Time
  = 2604 ns - 33.5 ns - 345.7 ns
  = 2225 ns

Copyright © 2013–2016, Texas Instruments Incorporated

23.0.21.3.1.2 Receive Data Ready

Similarly, the receive data ready flag RDATA bit in the RSTAT reflects the status of the RBUF register. The RDATA flag is set when data is transferred from the XRSR[n] shift registers to the XRBUF[n] buffers, indicating that the RBUF contains received data and is ready to have the CPU read the data. This flag is cleared when the RDATA bit is written with a 1, or when all the serializers configured as receivers are read.

Whenever RDATA is set, an DMA event AREVT is automatically generated to notify the DMA of the RBUF ready status. An interrupt ARINT is also generated if RDATA interrupt is enabled in the RINTCTL register (See Section 23.0.21.5.2 for details).

For DMA requests, the McASP does not require RSTAT to be read between DMA events. This means that even if RSTAT already has the RDATA flag set to 1 from a previous request, the next transfer triggers another DMA request.

Since all serializers act in lockstep, only one DMA event is generated to indicate that all active receive serializers are ready to receive new data.

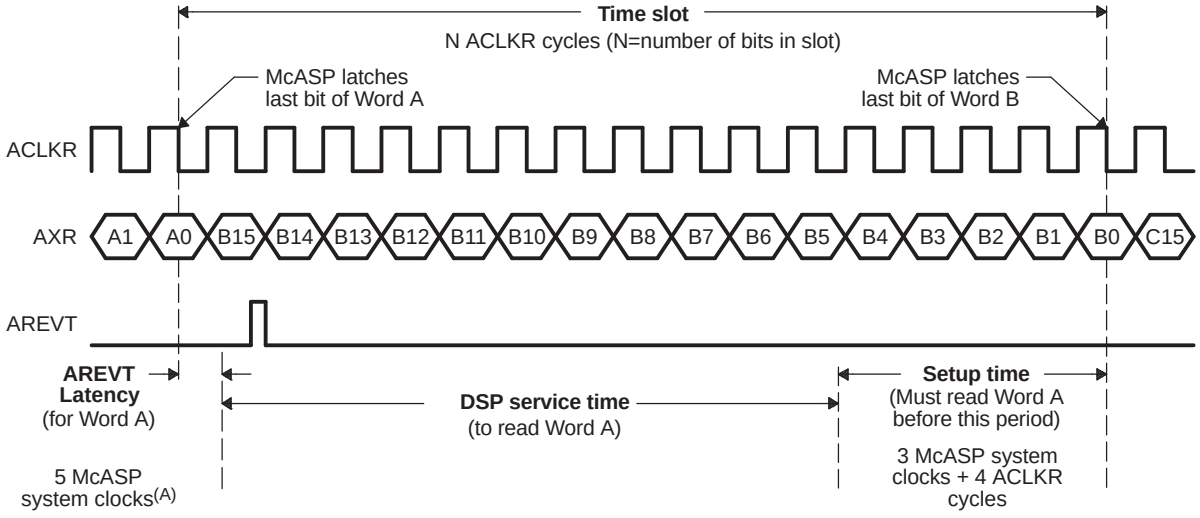
Figure 23-26 shows the timing details of when AREVT is generated at the McASP boundary. In this example, as soon as the last bit (bit A0) of Word A is received, the McASP sets the RDATA flag and generates an AREVT event. However, it takes up to 5 McASP system clocks (AREVT Latency) before AREVT is active at the McASP boundary. Upon AREVT, the CPU can begin servicing the McASP by reading Word A from the RBUF (DSP Service Time). The CPU must read Word A from the XBUF no later than the setup time required by the McASP (Setup Time).

The maximum DSP Service Time (Figure 23-26) can be calculated as:

DSP Service Time = Time Slot - AREVT Latency - Setup Time

The DSP Service Time calculation for receive is similar to the calculation for transmit. See Example 23-5 for DSP Service Time calculation using transmit as an example.

Figure 23-26. DSP Service Time Upon Receive DMA Event (AREVT)



A This is not the same as AUXCLK. The CPU uses SYSCLK2 as the McASP system clock source.

Copyright © 2013–2016, Texas Instruments Incorporated

23.0.21.3.2 Transfers through the DMA Port

CAUTION

To perform internal transfers through the DMA port, clear XBUSEL/RBUSEL bit to 0 in the respective XFMT/RFMT registers. Failure to do so will result in software malfunction.

Typically, you will access the McASP XRBUF registers through the DMA port. To access through the DMA port, simply have the CPU or DMA access the XRBUF through its DMA port location. See your device-specific data manual for the exact memory address. Through the DMA port, the DMA/CPU can service all the serializers through a single address. The McASP automatically cycles through the appropriate serializers.

For transmit operations through the DMA port, the DMA/CPU should write to the same XBUF DMA port address to service all of the active transmit serializers. In addition, the DMA/CPU should write to the XBUF for all active transmit serializers in incremental (although not necessarily consecutive) order. For example, if serializers 0, 4, 5, and 7 are set up as active transmitters, the DMA/CPU should write to the XBUF DMA port address four times with data for serializers 0, 4, 5, and 7 upon each transmit data ready event. This exact servicing order must be followed so that data appears in the appropriate serializers.

Similarly, for receive operations through the DMA port, the DMA/CPU should read from the same RBUF DMA port address to service all of the active receive serializers. In addition, reads from the active receive serializers through the DMA port return data in incremental (although not necessarily consecutive) order. For example, if serializers 1, 2, 3, and 6 are set up as active receivers, the DMA/CPU should read from the RBUF DMA port address four times to obtain data for serializers 1, 2, 3, and 6 in this exact order, upon each receive data ready event.

When transmitting, the DMA/CPU must write data to each serializer configured as "active" and "transmit" within each time slot. Failure to do so results in a buffer underrun condition (Section 23.0.21.6.2). Similarly, when receiving, data must be read from each serializer configured as "active" and "receive" within each time slot. Failure to do results in a buffer overrun condition (Section 23.0.21.6.3).

To perform internal transfers through the DMA port, clear XBUSEL/RBUSEL bit to 0 in the respective XFMT/RFMT registers.

23.0.21.3.3 Transfers Through the Peripheral Configuration Bus

CAUTION

The CPU does not support the emulation suspend signal. Therefore, if a data window is open in the Code Composer Studio™ integrated development environment to observe the XRBUF locations, the emulation read from the XRBUF locations causes an undesirable side effect of clearing the RDATA bit in RSTAT. Furthermore, if you write to the XRBUF through the Code Composer Studio™ integrated development environment, the emulation write to the XRBUF locations causes the XDATA bit in XSTAT to be cleared.

To perform internal transfers through the peripheral configuration bus, set XBUSEL/RBUSEL bit to 1 in the respective XFMT/RFMT registers. Failure to do so will result in software malfunction.

In this method, the DMA/CPU accesses the XRBUF through the peripheral configuration bus address. The exact XRBUF address for any particular serializer is determined by adding the offset for that particular serializer to the base address for the particular McASP (found in the device-specific data manual). XRBUF for the serializers configured as transmitters is given the name XBUF*n*. For example, the XRBUF associated with transmit serializer 2 is named XBUF2. Similarly, XRBUF for the serializers configured as receivers is given the name RBUF*n*.

Copyright © 2013–2016, Texas Instruments Incorporated

Accessing the XRBUF through the DMA port is different because the CPU/DMA only needs to access one single address. When accessing through the peripheral configuration bus, the CPU/DMA must provide the exact XBUF*n* or RBUF*n* address for each access.

When transmitting, DMA/CPU must write data to each serializer configured as "active" and "transmit" within each time slot. Failure to do so results in a buffer underrun condition (Section 23.0.21.6.2). Similarly when receiving, data must be read from each serializer configured as "active" and "receive" within each time slot. Failure to do results in a buffer overrun condition (Section 23.0.21.6.3).

23.0.21.3.4 Using the CPU for McASP Servicing

The CPU can be used to service the McASP through interrupt (upon AXINT/ARINT interrupts) or through polling the XDATA bit in the XSTAT register. As discussed in Section 23.0.21.3.2 and Section 23.0.21.3.3, the CPU can access either through the DMA port or through the peripheral configuration port.

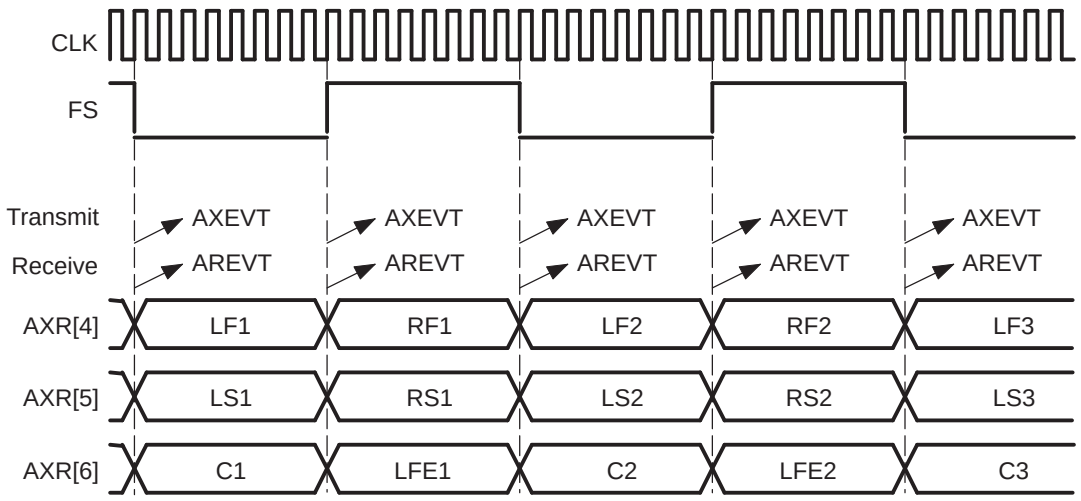
To use the CPU to service the McASP through interrupts, the XSTAT/RSTAT bit must be enabled in the respective XINTCTL/RINTCTL registers, to generate interrupts AXINT/ARINT to the CPU upon data ready.

23.0.21.3.5 Using the DMA for McASP Servicing

The most typical scenario is to use the DMA to service the McASP through the DMA port, although the DMA can also service the McASP through the peripheral configuration port. Use AXEVT/AREVT that is triggered upon each XDATA/RDATA transition from 0 to 1.

Figure 23-27 shows an example audio system with six audio channels (LF, RF, LS, RS, C, and LFE) transmitted from three AXR[n] pins on the McASP and shows when events AXEVT and AREVT are triggered.

Figure 23-27. DMA Events in an Audio Example–Two Events



In Figure 23-27, a DMA event AXEVT/AREVT is triggered on each time slot. In the example, AXEVT is triggered for each of the transmit audio channel time slot (time slot for channels LF, LS, and C; and time slot for channels RF, RS, LFE). Similarly, AREVT is triggered for each of the receive audio channel time slot. This allows for the use of a single DMA to transmit all audio channels, and a single DMA to receive all audio channels.

Note the difference between DMA event generation and the CPU interrupt generation. DMA events are generated automatically upon data ready; whereas CPU interrupt generation needs to be enabled in the XINTCTL/RINTCTL register.

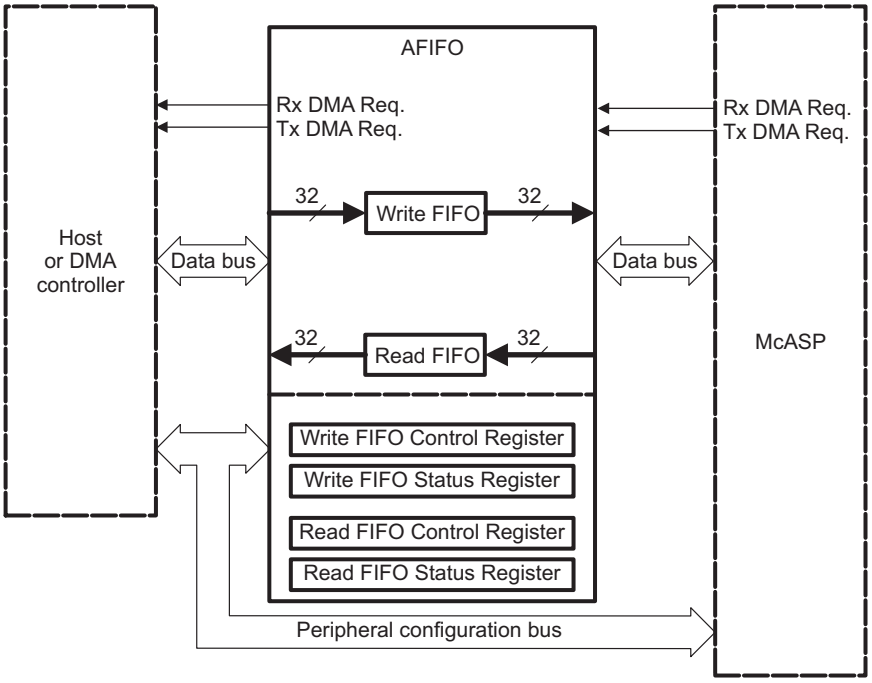
Copyright © 2013–2016, Texas Instruments Incorporated

McASP Audio FIFO (AFIFO)

The AFIFO contains two FIFOs: one Read FIFO (RFIFO), and one Write FIFO (WFIFO). To ensure backward compatibility with existing software, both the Read and Write FIFOs are disabled by default. See Figure 23-28 for a high-level block diagram of the AFIFO.

The AFIFO may be enabled/disabled and configured via the WFIFOCTL and RFIFOCTL registers. Note that if the Read or Write FIFO is to be enabled, it must be enabled prior to initializing the receive/transmit section of the McASP (see Section 23.0.21.1.2 for details).

Figure 23-28. McASP Audio FIFO (AFIFO) Block Diagram



AFIFO Data Transmission

When the Write FIFO is disabled, transmit DMA requests pass through directly from the McASP to the host/DMA controller. Whether the WFIFO is enabled or disabled, the McASP generates transmit DMA requests as needed; the AFIFO is “invisible” to the McASP.

When the Write FIFO is enabled, transmit DMA requests from the McASP are sent to the AFIFO, which in turn generates transmit DMA requests to the host/DMA controller.

If the Write FIFO is enabled, upon a transmit DMA request from the McASP, the WFIFO writes *WNUMDMA* 32-bit words to the McASP if and when there are at least *WNUMDMA* words in the Write FIFO. If there are not, the WFIFO waits until this condition has been satisfied. At that point, it writes *WNUMDMA* words to the McASP. (See description for WFIFOCTL.WNUMDMA in Section 23.1.45.)

If the host CPU writes to the Write FIFO, independent of a transmit DMA request, the WFIFO will accept host writes until full. After this point, excess data will be discarded.

Note that when the WFIFO is first enabled, it will immediately issue a transmit DMA request to the host. This is because it begins in an empty state, and is therefore ready to accept data.

Copyright © 2013–2016, Texas Instruments Incorporated

Transmit DMA Event Pacer

The AFIFO may be configured to delay making a transmit DMA request to the host until the Write FIFO has enough space for a specified number of words. In this situation, the number of transmit DMA requests to the host or DMA controller is reduced.

If the Write FIFO has space to accept *WNUMEVT* 32-bit words, it generates a transmit DMA request to the host and then waits for a response. Once *WNUMEVT* words have been written to the FIFO, it checks again to see if there is space for *WNUMEVT* 32-bit words. If there is space, it generates another transmit DMA request to the host, and so on. In this fashion, the Write FIFO will attempt to stay filled.

Note that if transmit DMA event pacing is desired, WFIFOCTL.WNUMEVT should be set to a non-zero integer multiple of the value in WFIFOCTL.WNUMDMA. If transmit DMA event pacing is not desired, then the value in WFIFOCTL.WNUMEVT should be set equal to the value in WFIFOCTL.WNUMDMA.

AFIFO Data Reception

When the Read FIFO is disabled, receive DMA requests pass through directly from McASP to the host/DMA controller. Whether the RFIFO is enabled or disabled, the McASP generates receive DMA requests as needed; the AFIFO is “invisible” to the McASP.

When the Read FIFO is enabled, receive DMA requests from the McASP are sent to the AFIFO, which in turn generates receive DMA requests to the host/DMA controller.

If the Read FIFO is enabled and the McASP makes a receive DMA request, the RFIFO reads *RNUMDMA* 32-bit words from the McASP, if and when the RFIFO has space for *RNUMDMA* words. If it does not, the RFIFO waits until this condition has been satisfied; at that point, it reads *RNUMDMA* words from the McASP. (See description for RFIFOCTL.RNUMDMA in Section 23.1.47.)

If the host CPU reads the Read FIFO, independent of a receive DMA request, and the RFIFO at that time contains less than *RNUMEVT* words, those words will be read correctly, emptying the FIFO.

Receive DMA Event Pacer

The AFIFO may be configured to delay making a receive DMA request to the host until the Read FIFO contains a specified number of words. In this situation, the number of receive DMA requests to the host or DMA controller is reduced.

If the Read FIFO contains at least *RNUMEVT* 32-bit words, it generates a receive DMA request to the host and then waits for a response. Once *RNUMEVT* 32-bit words have been read from the RFIFO, the RFIFO checks again to see if it contains at least another *RNUMEVT* words. If it does, it generates another receive DMA request to the host, and so on. In this fashion, the Read FIFO will attempt to stay empty.

Note that if receive DMA event pacing is desired, RFIFOCTL.RNUMEVT should be set to a non-zero integer multiple of the value in RFIFOCTL.RNUMDMA. If receive DMA event pacing is not desired, then the value in RFIFOCTL.RNUMEVT should be set equal to the value in RFIFOCTL.RNUMDMA.

Arbitration Between Transmit and Receive DMA Requests

If both the WFIFO and the RFIFO are enabled and a transmit DMA request and receive DMA request occur simultaneously, priority is given to the transmit DMA request. Once a transfer is in progress, it is allowed to complete.

If only the WFIFO is enabled and a transmit DMA request and receive DMA request occur simultaneously, priority is given to the transmit DMA request. Once a transfer is in progress, it is allowed to complete.

If only the RFIFO is enabled and a transmit DMA request and receive DMA request occur simultaneously, priority is given to the receive DMA request. Once a transfer is in progress, it is allowed to complete.

Copyright © 2013–2016, Texas Instruments Incorporated

23.0.21.4 Formatter

23.0.21.4.1 Transmit Bit Stream Data Alignment

The McASP transmitter supports serial formats of:

- Slot (or Time slot) size = 8, 12, 16, 20, 24, 28, 32 bits
- Word size ≤ Slot size
- Alignment: when more bits/slot than bits/words, then:
  - Left aligned = word shifted first, remaining bits are pad
  - Right aligned = pad bits are shifted first, word occupies the last bits in slot
- Order: order of bits shifted out:
  - MSB: most-significant bit of word is shifted out first, last bit is LSB
  - LSB: least-significant bit of word is shifted out last, last bit is MSB

Hardware support for these serial formats comes from the programmable options in the transmit bitstream format register (XFMT):

- XRVRS: bit reverse (1) or no bit reverse (0)
- XROT: rotate right by 0, 4, 8, 12, 16, 20, 24, or 28 bits
- XSSZ: transmit slot size of 8, 12, 16, 20, 24, 28, or 32 bits

XSSZ should always be programmed to match the slot size of the serial stream. The word size is not directly programmed into the McASP, but rather is used to determine the rotation needed in the XROT field.

Table 23-4 and Figure 23-29 show the XRVRS and XROT fields for each serial format and for both integer and Q31 fractional internal representations.

This discussion assumes that all slot size (SLOT in Table 23-4) and word size (WORD in Table 23-4) options are multiples of 4, since the transmit rotate right unit only supports rotation by multiples of 4. However, the bit mask/pad unit does allow for any number of significant digits. For example, a Q31 number may have 19 significant digits (word) and be transmitted in a 24-bit slot; this would be formatted as a word size of 20 bits and a slot size of 24 bits. However, it is possible to set the bit mask unit to only pass the 19 most-significant digits (program the mask value to FFFF E000h). The digits that are not significant can be set to a selected pad value, which can be any one of the significant digits, a fixed value of 0, or a fixed value of 1.

The transmit bit mask/pad unit operates on data as an initial step of the transmit format unit (see Figure 23-20), and the data is aligned in the same representation as it is written to the transmitter by the CPU (typically Q31 or integer).

Table 23-4. Transmit Bitstream Data Alignment

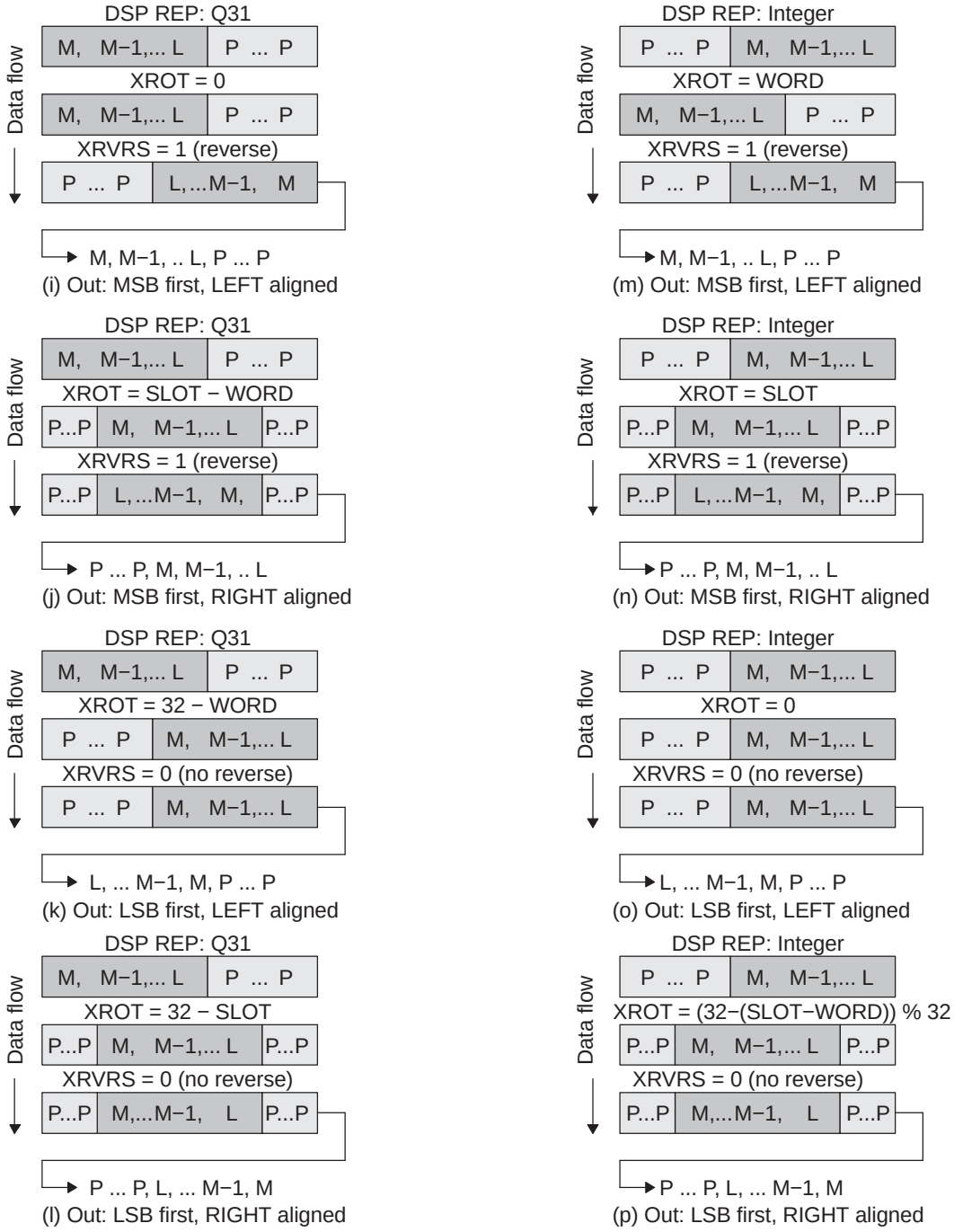
Figure 23-29	Bit Stream Order	Bit Stream Alignment	Internal Numeric Representation	XFMT Bit	
				XROT[1]	XRVRS
(a)[2]	MSB first	Left aligned	Q31 fraction	0	1
(b)	MSB first	Right aligned	Q31 fraction	SLOT - WORD	1
(c)	LSB first	Left aligned	Q31 fraction	32 - WORD	0
(d)	LSB first	Right aligned	Q31 fraction	32 - SLOT	0
(e)[2]	MSB first	Left aligned	Integer	WORD	1
(f)	MSB first	Right aligned	Integer	SLOT	1
(g)	LSB first	Left aligned	Integer	0	0
(h)	LSB first	Right aligned	Integer	(32 - (SLOT - WORD)) % 32	0

(1) WORD = Word size rounded up to the nearest multiple of 4; SLOT = slot size; % = modulo operator

(2) To transmit in I2S format, use MSB first, left aligned, and also select XDATDLY = 01 (1 bit delay)

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 23-29. Data Flow Through Transmit Format Unit



Copyright © 2013–2016, Texas Instruments Incorporated

23.0.21.4.2 Receive Bit Stream Data Alignment

The McASP receiver supports serial formats of:

- Slot or time slot size = 8, 12, 16, 20, 24, 28, 32 bits
- Word size ≤ Slot size
- Alignment when more bits/slot than bits/words, then:
  - Left aligned = word shifted first, remaining bits are pad
  - Right aligned = pad bits are shifted first, word occupies the last bits in slot
- Order of bits shifted out:
  - MSB: most-significant bit of word is shifted out first, last bit is LSB
  - LSB: least-significant bit of word is shifted out last, last bit is MSB

Hardware support for these serial formats comes from the programmable options in the receive bitstream format register (RFMT):

- RRVRS: bit reverse (1) or no bit reverse (0)
- RROT: rotate right by 0, 4, 8, 12, 16, 20, 24, or 28 bits
- RSSZ: receive slot size of 8, 12, 16, 20, 24, 28, or 32 bits

RSSZ should always be programmed to match the slot size of the serial stream. The word size is not directly programmed into the McASP, but rather is used to determine the rotation needed in the RROT field.

Table 23-5 and Figure 23-30 show the RRVRS and RROT fields for each serial format and for both integer and Q31 fractional internal representations.

This discussion assumes that all slot size and word size options are multiples of 4; since the receive rotate right unit only supports rotation by multiples of 4. However, the bit mask/pad unit does allow for any number of significant digits. For example, a Q31 number may have 19 significant digits (word) and be transmitted in a 24-bit slot; this would be formatted as a word size of 20 bits and a slot size of 24 bits. However, it is possible to set the bit mask unit to only pass the 19 most-significant digits (program the mask value to FFFF E000h). The digits that are not significant can be set to a selected pad value, which can be any one of the significant digits, a fixed value of 0, or a fixed value of 1.

The receive bit mask/pad unit operates on data as the final step of the receive format unit (see Figure 23-19), and the data is aligned in the same representation as it is read from the receiver by the CPU (typically Q31 or integer).

Table 23-5. Receive Bitstream Data Alignment

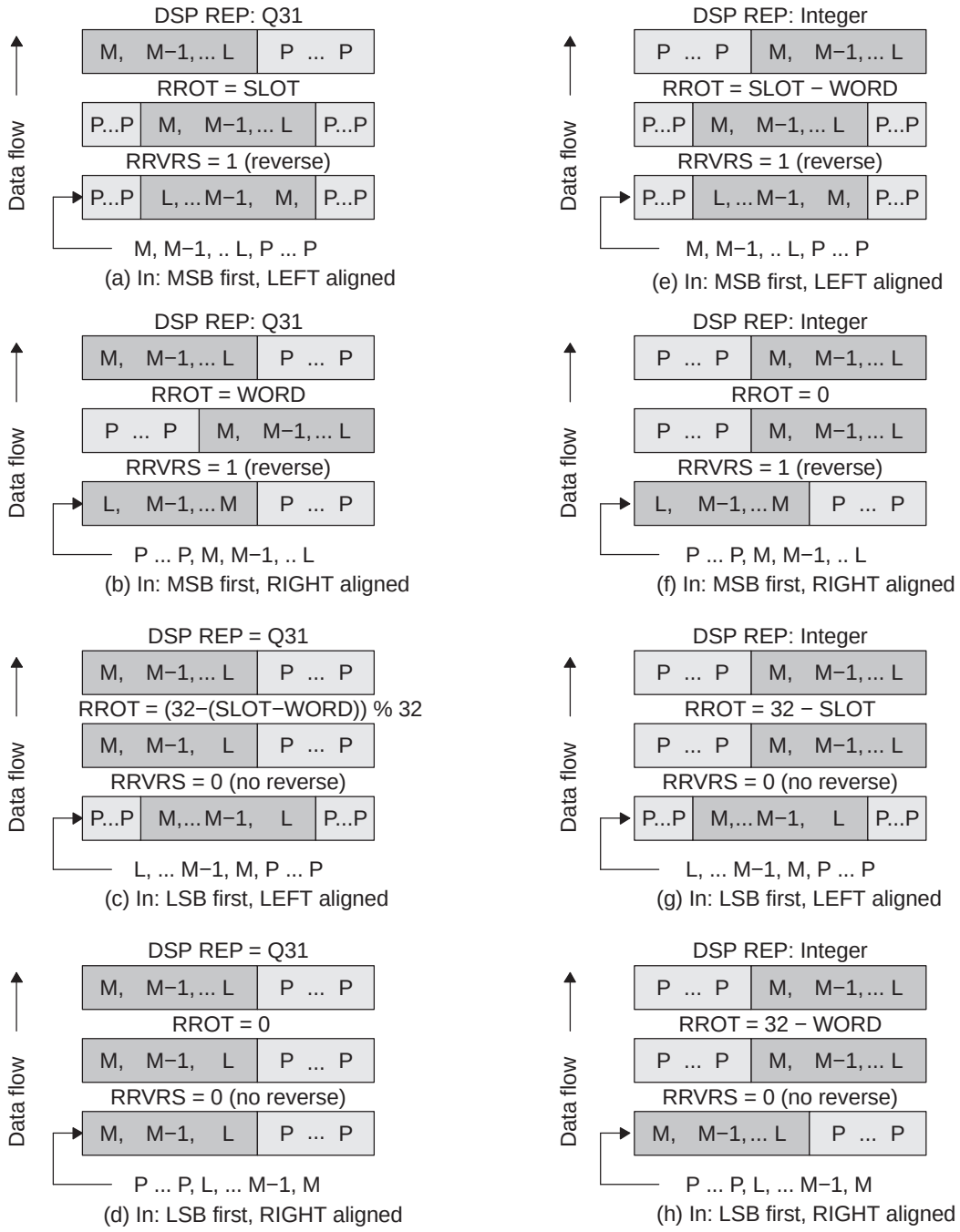
Figure 23-30	Bit Stream Order	Bit Stream Alignment	Internal Numeric Representation	RFMT Bit	
				RROT[1]	RRVRS
(a)[2]	MSB first	Left aligned	Q31 fraction	SLOT	1
(b)	MSB first	Right aligned	Q31 fraction	WORD	1
(c)	LSB first	Left aligned	Q31 fraction	(32 - (SLOT - WORD)) % 32	0
(d)	LSB first	Right aligned	Q31 fraction	0	0
(e)[2]	MSB first	Left aligned	Integer	SLOT - WORD	1
(f)	MSB first	Right aligned	Integer	0	1
(g)	LSB first	Left aligned	Integer	32 - SLOT	0
(h)	LSB first	Right aligned	Integer	32 - WORD	0

(1) WORD = Word size rounded up to the nearest multiple of 4; SLOT = slot size; % = modulo operator

(2) To transmit in I2S format, select MSB first, left aligned, and also select RDATDLY = 01 (1 bit delay)

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 23-30. Data Flow Through Receive Format Unit



Copyright © 2013–2016, Texas Instruments Incorporated

23.0.21.5 Interrupts

23.0.21.5.1 Transmit Data Ready Interrupt

The transmit data ready interrupt (XDATA) is generated if XDATA is 1 in the XSTAT register and XDATA is also enabled in XINTCTL. Section 23.0.21.3.1 provides details on when XDATA is set in the XSTAT register.

A transmit start of frame interrupt (XSTAFRM) is triggered by the recognition of transmit frame sync. A transmit last slot interrupt (XLAST) is a qualified version of the data ready interrupt (XDATA). It has the same behavior as the data ready interrupt, but is further qualified by having the data requested belonging to the last slot (the slot that just ended was next-to-last TDM slot, current slot is last slot).

23.0.21.5.2 Receive Data Ready Interrupt

The receive data ready interrupt (RDATA) is generated if RDATA is 1 in the RSTAT register and RDATA is also enabled in RINTCTL. Section 23.0.21.3.2 provides details on when RDATA is set in the RSTAT register.

A receiver start of frame interrupt (RSTAFRM) is triggered by the recognition of a receiver frame sync. A receiver last slot interrupt (RLAST) is a qualified version of the data ready interrupt (RDATA). It has the same behavior as the data ready interrupt, but is further qualified by having the data in the buffer come from the last TDM time slot (the slot that just ended was last TDM slot).

23.0.21.5.3 Error Interrupts

Upon detection, the following error conditions generate interrupt flags:

- In the receive status register (RSTAT):
  - Receiver overrun (ROVRN)
  - Unexpected receive frame sync (RSYNCERR)
  - Receive clock failure (RCKFAIL)
  - Receive DMA error (RDMAERR)
- In the transmit status register (XSTAT):
  - Transmit underrun (XUNDRN)
  - Unexpected transmit frame sync (XSYNCERR)
  - Transmit clock failure (XCKFAIL)
  - Transmit DMA error (XDMAERR)

Each interrupt source also has a corresponding enable bit in the receive interrupt control register (RINTCTL) and transmit interrupt control register (XINTCTL). If the enable bit is set in RINTCTL or XINTCTL, an interrupt is requested when the interrupt flag is set in RSTAT or XSTAT. If the enable bit is not set, no interrupt request is generated. However, the interrupt flag may be polled.

23.0.21.5.4 Audio Mute (AMUTE) Function

The McASP includes an automatic audio mute function (Figure 23-31) that asserts in hardware the AMUTE device pin to a preprogrammed output state, as selected by the MUTEN bit in the audio mute control register (AMUTE). The AMUTE device pin is asserted when one of the interrupt flags is set or an external device issues an error signal on the AMUTEIN input. Typically, the AMUTEIN input is shared with a device pin.

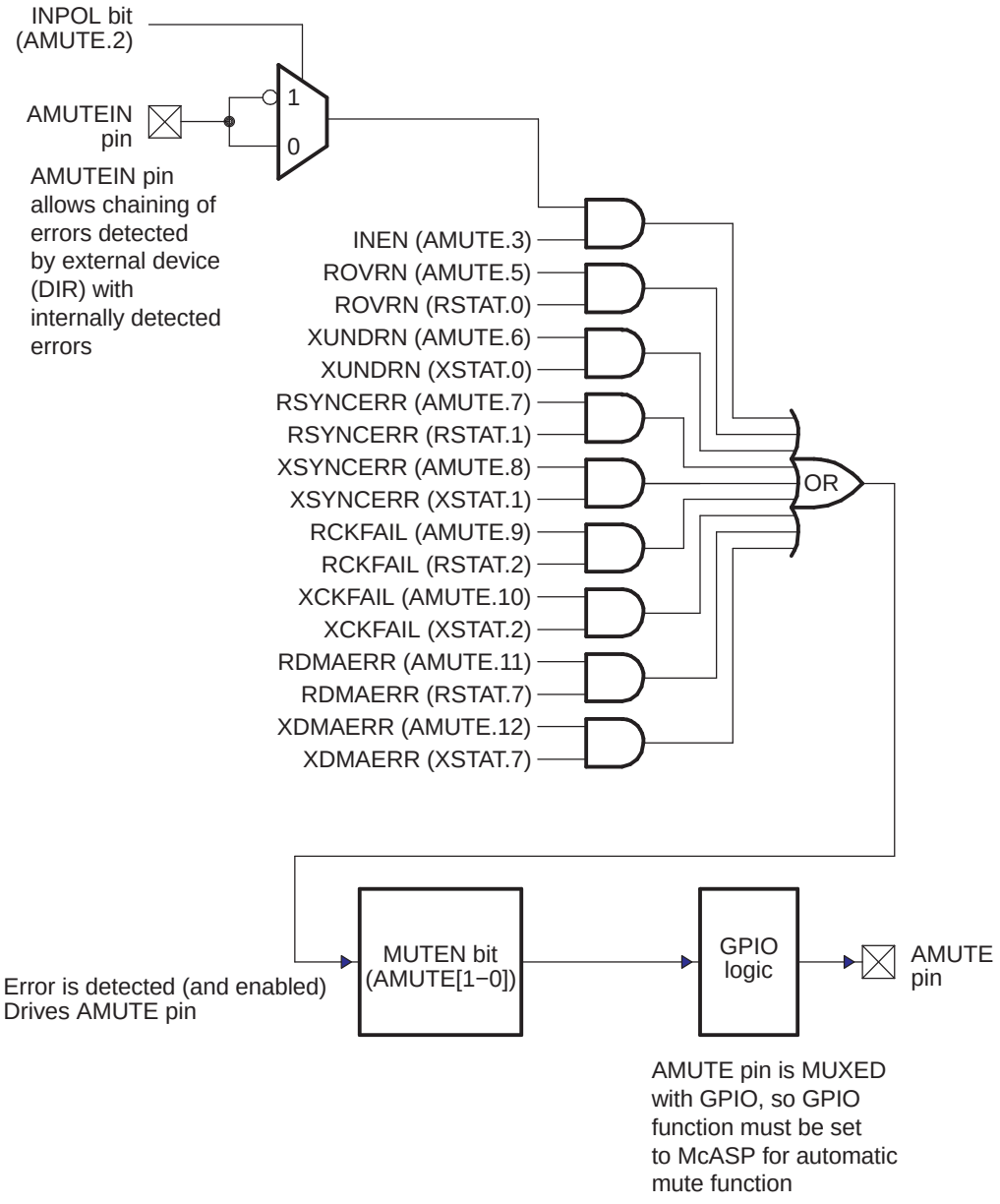
The AMUTEIN input allows the on-chip logic to consider a mute input from other devices in the system, so that all errors may be considered. The AMUTEIN input has a programmable polarity to allow it to adapt to different devices, as selected by the INPOL bit in AMUTE, and it must be enabled explicitly.

In addition to the external AMUTEIN input, the AMUTE device pin output may be asserted when one of the error interrupt flags is set and its mute function is enabled in AMUTE.

Copyright © 2013–2016, Texas Instruments Incorporated

When one or more of the errors is detected and enabled, the AMUTE device pin is driven to an active state that is selected by MUTEN in AMUTE. The active polarity of the AMUTE device pin is programmable by MUTEN (and the inactive polarity is the opposite of the active polarity). The AMUTE device pin remains driven active until software clears all the error interrupt flags that are enabled to mute, and until the AMUTEIN is inactive.

Figure 23-31. Audio Mute (AMUTE) Block Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

23.0.21.5.5 Multiple Interrupts

This only applies to interrupts and not to DMA requests. The following terms are defined:

- **Active Interrupt Request:** a flag in RSTAT or XSTAT is set and the interrupt is enabled in RINTCTL or XINTCTL.
- **Outstanding Interrupt Request:** An interrupt request has been issued on one of the McASP transmit/receive interrupt ports, but that request has not yet been serviced.
- **Serviced:** The CPU writes to RSTAT or XSTAT to clear one or more of the active interrupt request flags.

The first interrupt request to become active for the transmitter with the interrupt flag set in XSTAT and the interrupt enabled in XINTCTL generates a request on the McASP transmit interrupt port AXINT.

If more than one interrupt request becomes active in the same cycle, a single interrupt request is generated on the McASP transmit interrupt port. Subsequent interrupt requests that become active while the first interrupt request is outstanding do not immediately generate a new request pulse on the McASP transmit interrupt port.

The transmit interrupt is serviced with the CPU writing to XSTAT. If any interrupt requests are active after the write, a new request is generated on the McASP transmit interrupt port.

The receiver operates in a similar way, but using RSTAT, RINTCTL, and the McASP receive interrupt port ARINT.

One outstanding interrupt request is allowed on each port, so a transmit and a receive interrupt request may both be outstanding at the same time.

23.0.21.6 Error Handling and Management

To support the design of a robust audio system, the McASP includes error-checking capability for the serial protocol, data underrun, and data overrun. In addition, the McASP includes a timer that continually measures the high-frequency master clock every 32 AHCLKX/AHCLKR clock cycles. The timer value can be read to get a measurement of the clock frequency and has a minimum and maximum range setting that can set an error flag if the master clock goes out of a specified range.

Upon the detection of any one or more errors (software selectable), or the assertion of the AMUTEIN input pin, the AMUTE output pin may be asserted to a high or low level to immediately mute the audio output. In addition, an interrupt may be generated if desired, based on any one or more of the error sources.

23.0.21.6.1 Unexpected Frame Sync Error

An unexpected frame sync occurs when:

- In burst mode, when the next active edge of the frame sync occurs early such that the current slot will not be completed by the time the next slot is scheduled to begin.
- In TDM mode, a further constraint is that the frame sync must occur exactly during the correct bit clock (not a cycle earlier or later) and only before slot 0. An unexpected frame sync occurs if this condition is not met.

When an unexpected frame sync occurs, there are two possible actions depending upon when the unexpected frame sync occurs:

1. Early: An early unexpected frame sync occurs when the McASP is in the process of completing the current frame and a new frame sync is detected (not including overlap that occurs due to a 1 or 2 bit frame sync delay). When an early unexpected frame sync occurs:
    - Error interrupt flag is set (XSYNCERR, if an unexpected transmit frame sync occurs; RSYNCERR, if an unexpected receive frame sync occurs).
    - Current frame is not resynchronized. The number of bits in the current frame is completed. The next frame sync, which occurs after the current frame is completed, will be resynchronized.

Copyright © 2013–2016, Texas Instruments Incorporated

2. Late: A late unexpected frame sync occurs when there is a gap or delay between the last bit of the previous frame and the first bit of the next frame. When a late unexpected frame sync occurs (as soon as the gap is detected):
   - Error interrupt flag is set (XSYNCERR, if an unexpected transmit frame sync occurs; RSYNCERR, if an unexpected receive frame sync occurs).
   - Resynchronization occurs upon the arrival of the next frame sync.

Late frame sync is detected the same way in both burst mode and TDM mode; however, in burst mode, late frame sync is not meaningful and its interrupt enable should not be set.

23.0.21.6.2 Buffer Underrun Error - Transmitter

A buffer underrun can only occur for serializers programmed to be transmitters. A buffer underrun occurs when the serializer is instructed by the transmit state machine to transfer data from XRBUF[n] to XRSR[n], but XRBUF[n] has not yet been written with new data since the last time the transfer occurred. When this occurs, the transmit state machine sets the XUNDRN flag.

An underrun is checked only once per time slot. The XUNDRN flag is set when an underrun condition occurs. Once set, the XUNDRN flag remains set until the CPU explicitly writes a 1 to the XUNDRN bit to clear the XUNDRN bit.

In DIT mode, a pair of BMC zeros is shifted out when an underrun occurs (four bit times at 128 × f_s). By shifting out a pair of zeros, a clock may be recovered on the receiver. To recover, reset the McASP and start again with the proper initialization.

In TDM mode, during an underrun case, a long stream of zeros are shifted out causing the DACs to mute. To recover, reset the McASP and start again with the proper initialization.

23.0.21.6.3 Buffer Overrun Error - Receiver

A buffer overrun can only occur for serializers programmed to be receivers. A buffer overrun occurs when the serializer is instructed to transfer data from XRSR[n] to XRBUF[n], but XRBUF[n] has not yet been read by either the DMA or the CPU. When this occurs, the receiver state machine sets the ROVRN flag. However, the individual serializer writes over the data in the XRBUF[n] register (destroying the previous sample) and continues shifting.

An overrun is checked only once per time slot. The ROVRN flag is set when an overrun condition occurs. It is possible that an overrun occurs on one time slot but then the CPU catches up and does not cause an overrun on the following time slots. However, once the ROVRN flag is set, it remains set until the CPU explicitly writes a 1 to the ROVRN bit to clear the ROVRN bit.

23.0.21.6.4 DMA Error - Transmitter

A transmit DMA error, as indicated by the XDMAERR flag in the XSTAT register, occurs when the DMA (or CPU) writes more words to the DMA port of the McASP than it should. For each DMA event, the DMA should write exactly as many words as there are serializers enabled as transmitters.

XDMAERR indicates that the DMA (or CPU) wrote too many words to the McASP for a given transmit DMA event. Writing too few words results in a transmit underrun error setting XUNDRN in XSTAT.

While XDMAERR occurs infrequently, an occurrence indicates a serious loss of synchronization between the McASP and the DMA or CPU. You should reinitialize both the McASP transmitter and the DMA to resynchronize them.

Copyright © 2013–2016, Texas Instruments Incorporated

23.0.21.6.5 DMA Error - Receiver

A receive DMA error, as indicated by the RDMAERR flag in the RSTAT register, occurs when the DMA (or CPU) reads more words from the DMA port of the McASP than it should. For each DMA event, the DMA should read exactly as many words as there are serializers enabled as receivers.

RDMAERR indicates that the DMA (or CPU) read too many words from the McASP for a given receive DMA event. Reading too few words results in a receiver overrun error setting ROVRN in RSTAT.

While RDMAERR occurs infrequently, an occurrence indicates a serious loss of synchronization between the McASP and the DMA or CPU. You should reinitialize both the McASP receiver and the DMA to resynchronize them.

23.0.21.6.6 Clock Failure Detection

23.0.21.6.6.1 Clock-Failure Check Startup

It is expected, initially, that the clock-failure circuits will generate an error until at least one measurement has been taken. Therefore, the clock failure interrupts, clock switch, and mute functions should not immediately be enabled, but be enabled only after a specific startup procedure. The startup procedure is:

1. For the transmit clock failure check:
   (a) Configure transmit clock failure detect logic (XMIN, XMAX, XPS) in the transmit clock check control register (XCLKCHK).
   (b) Clear transmit clock failure flag (XCKFAIL) in the transmit status register (XSTAT).
   (c) Wait until first measurement is taken (> 32 AHCLKX clock periods).
   (d) Verify no clock failure is detected.
   (e) Repeat steps b–d until clock is running and is no longer issuing clock failure errors.
   (f) After the transmit clock is measured and falls within the acceptable range, the following may be enabled:
      (i) transmit clock failure interrupt enable bit (XCKFAIL) in the transmitter interrupt control register (XINTCTL)
      (ii) transmit clock failure detect autoswitch enable bit (XCKFAILSW) in the transmit clock check control register (XCLKCHK)
      (iii) mute option (XCKFAIL) in the mute control register (AMUTE)
2. For the receive clock failure check:
   (a) Configure receive clock failure detect logic (RMIN, RMAX, RPS) in the receive clock check control register (RCLKCHK).
   (b) Clear receive clock failure flag (RCKFAIL) in the receive status register (RSTAT).
   (c) Wait until first measurement is taken (> 32 AHCLKR clock periods).
   (d) Verify no clock failure is detected.
   (e) Repeat steps b–d until clock is running and is no longer issuing clock failure errors.
   (f) After the receive clock is measured and falls within the acceptable range, the following may be enabled:
      (i) receive clock failure interrupt enable bit (RCKFAIL) in the receiver interrupt control register (RINTCTL)
      (ii) mute option (RCKFAIL) in the mute control register (AMUTE)

Copyright © 2013–2016, Texas Instruments Incorporated

23.0.21.6.6.2 Transmit Clock Failure Check and Recovery

The transmit clock failure check circuit (Figure 23-32) works off both the internal McASP system clock and the external high-frequency serial clock (AHCLKX). It continually counts the number of system clocks for every 32 high rate serial clock (AHCLKX) periods, and stores the count in XCNT of the transmit clock check control register (XCLKCHK) every 32 high rate serial clock cycles.

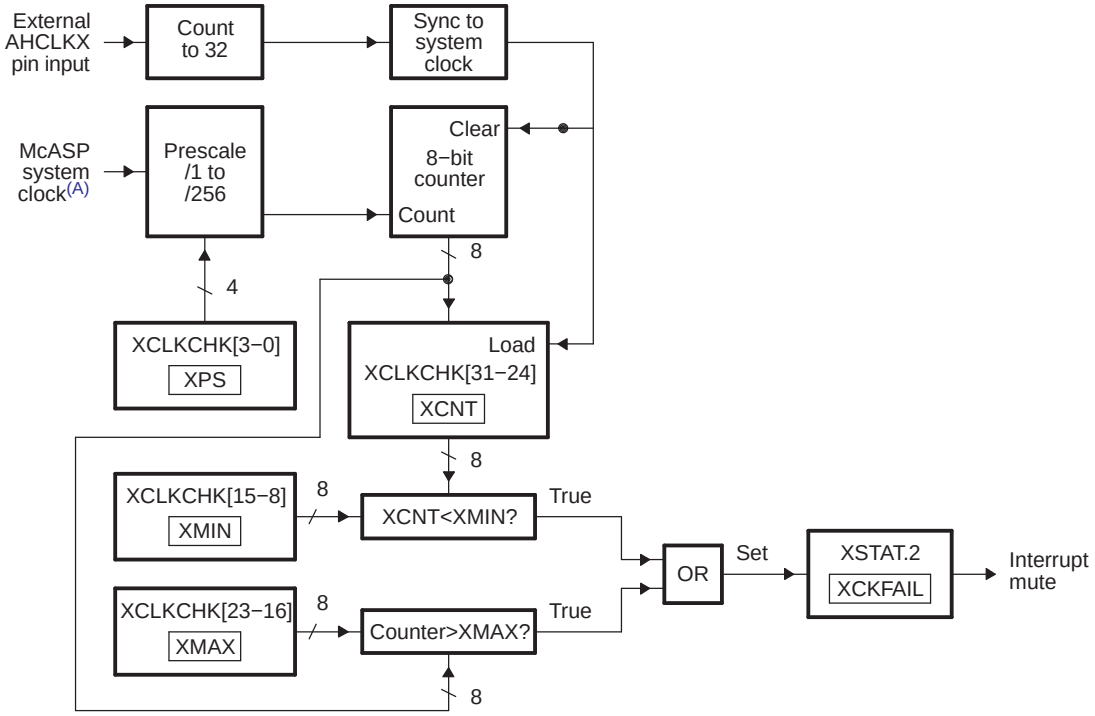
The logic compares the count against a user-defined minimum allowable boundary (XMIN), and automatically flags an interrupt (XCKFAIL in XSTST) when an out-of-range condition occurs. An out-of-range minimum condition occurs when the count is smaller than XMIN. The logic continually compares the current count (from the running system clock counter) against the maximum allowable boundary (XMAX). This is in case the external clock completely stops, so that the counter value is not copied to XCNT. An out-of-range maximum condition occurs when the count is greater than XMAX. Note that the XMIN and XMAX fields are 8-bit unsigned values, and the comparison is performed using unsigned arithmetic.

An out-of-range count may indicate either that an unstable clock was detected, or that the audio source has changed and a new sample rate is being used.

In order for the transmit clock failure check circuit to operate correctly, the high-frequency serial clock divider must be taken out of reset regardless if AHCLKX is internally generated or externally sourced.

If a clock failure is detected, the transmit clock failure flag (XCKFAIL) in XSTAT is set. This causes an interrupt, if the transmit clock failure interrupt enable bit (XCKFAIL) in XINTCTL is set.

Figure 23-32. Transmit Clock Failure Detection Circuit Block Diagram



A This is not the same as AUXCLK. The CPU uses SYSCLK2 as the McASP system clock.

Copyright © 2013–2016, Texas Instruments Incorporated

23.0.21.6.6.3 Receive Clock Failure Check and Recovery

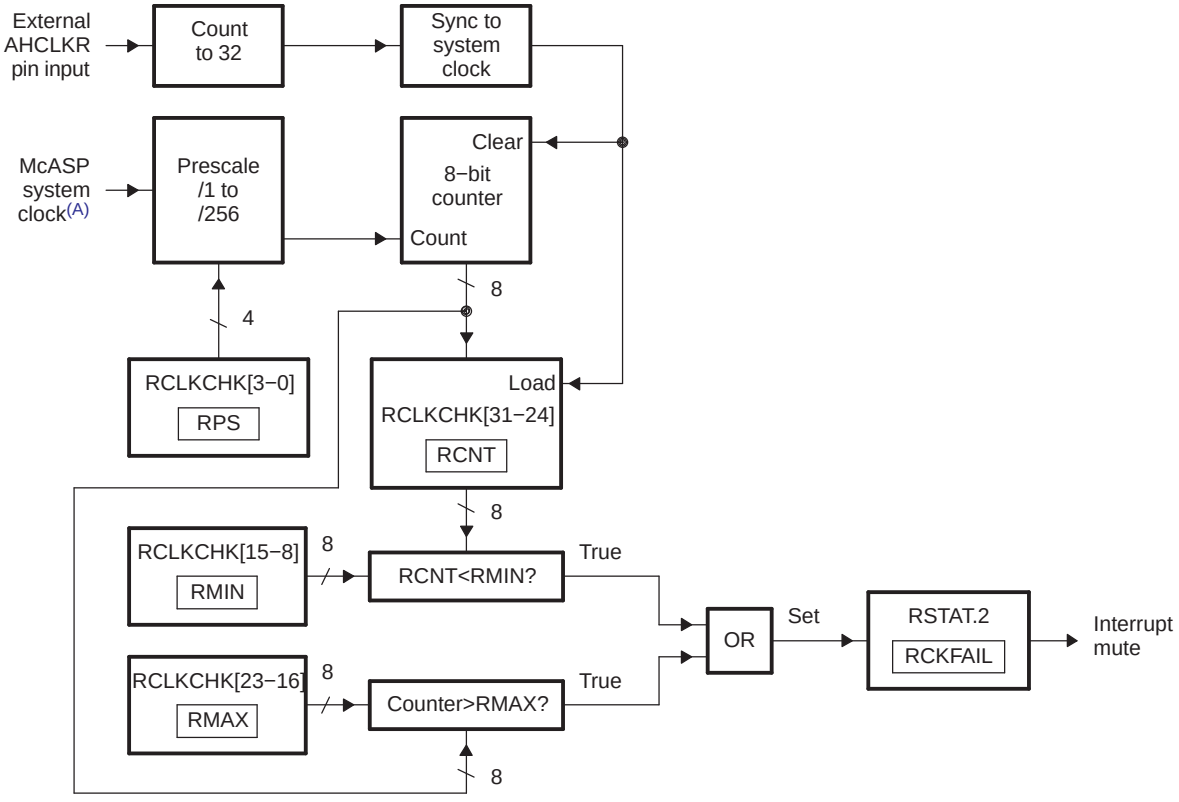
The receive clock failure check circuit (Figure 23-33) works off both the internal McASP system clock and the external high-frequency serial clock (AHCLKR). It continually counts the number of system clocks for every 32 high rate serial clock (AHCLKR) periods, and stores the count in RCNT of the receive clock check control register (RCLKCHK) every 32 high rate serial clock cycles.

The logic compares the count against a user-defined minimum allowable boundary (RMIN) and automatically flags an interrupt (RCKFAIL in RSTAT) when an out-of-range condition occurs. An out-of-range minimum condition occurs when the count is smaller than RMIN. The logic continually compares the current count (from the running system clock counter) against the maximum allowable boundary (RMAX). This is in case the external clock completely stops, so that the counter value is not copied to RCNT. An out-of-range maximum condition occurs when the count is greater than RMAX. Note that the RMIN and RMAX fields are 8-bit unsigned values, and the comparison is performed using unsigned arithmetic.

An out-of-range count may indicate either that an unstable clock was detected or that the audio source has changed and a new sample rate is being used.

In order for the receive clock failure check circuit to operate correctly, the high-frequency serial clock divider must be taken out of reset regardless if AHCLKR is internally generated or externally sourced.

Figure 23-33. Receive Clock Failure Detection Circuit Block Diagram



A This is not the same as AUXCLK. The CPU uses SYSCLK2 as the McASP system clock source.

Copyright © 2013–2016, Texas Instruments Incorporated

23.0.21.7 Loopback Modes

The McASP features a digital loopback mode (DLB) that allows testing of the McASP code in TDM mode with a single device. In loopback mode, output of the transmit serializers is connected internally to the input of the receive serializers. Therefore, you can check the receive data against the transmit data to ensure that the McASP settings are correct. Digital loopback mode applies to TDM mode only (2 to 32 slots in a frame). It does not apply to DIT mode (XMOD = 180h) or burst mode (XMOD = 0).

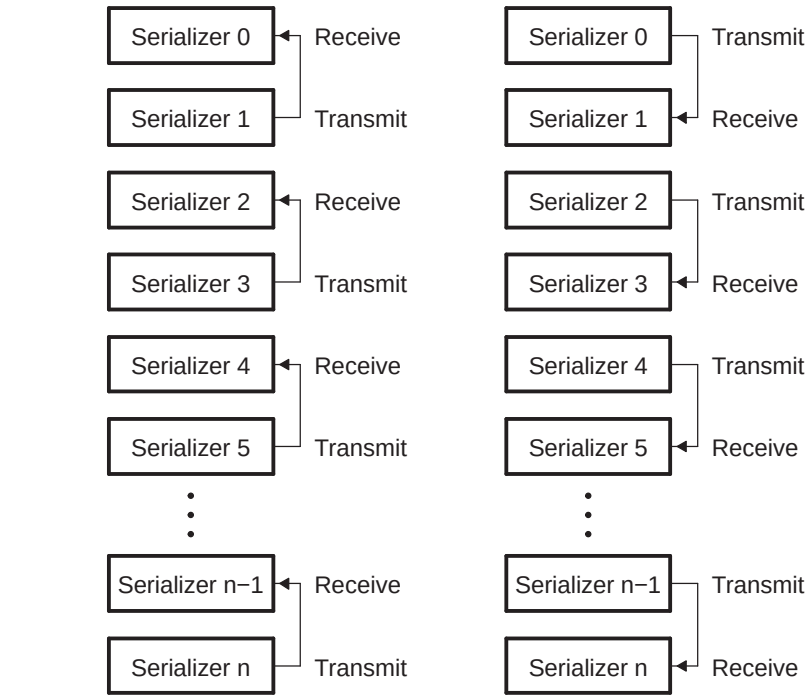
Figure 23-34 shows the basic logical connection of the serializers in loopback mode. Two types of loopback connections are possible, selected by the ORD bit in the digital loopback control register (DLBCTL) as follows:

- ORD = 0: Outputs of odd serializers are connected to inputs of even serializers. If this mode is selected, you should configure odd serializers to be transmitters and even serializers to be receivers.
- ORD = 1: Outputs of even serializers are connected to inputs of odd serializers. If this mode is selected, you should configure even serializers to be transmitters and odd serializers to be receivers.

Data can be externally visible at the I/O pin of the transmit serializer if the pin is configured as a McASP output pin by setting the corresponding PFUNC bit to 0 and PDIR bit to 1.

In loopback mode, the transmit clock and frame sync are used by both the transmit and receive sections of the McASP. The transmit and receive sections operate synchronously. This is achieved by setting the MODE bit of the DLBCTL register to 01b and the ASYNC bit of the ACLKXCTL register to 0.

Figure 23-34. Serializers in Loopback Mode



(a) DLBEN = 1 (loopback enabled) and ORD = 0 (even receive, odd transmit)

(b) DLBEN = 1 (loopback enabled) and ORD = 1 (odd receive, even transmit)

Copyright © 2013–2016, Texas Instruments Incorporated

23.0.21.7.1 Loopback Mode Configurations

This is a summary of the settings required for digital loopback mode for TDM format:

- The DLBEN bit in DLBCTL must be set to 1 to enable loopback mode.
- The MODE bits in DLBCTL must be set to 01b for both the transmit and receive sections to use the transmit clock and frame sync generator.
- The ORD bit in DLBCTL must be programmed appropriately to select odd or even serializers to be transmitters or receivers. The corresponding serializers must be configured accordingly.
- The ASYNC bit in ACLKXCTL must be cleared to 0 to ensure synchronous transmit and receive operations.
- RMOD field in AFSRCTL and XMOD field in AFSXCTL must be set to 2h to 20h to indicate TDM mode. Loopback mode does not apply to DIT or burst mode.

23.0.22 Reset Considerations

The McASP has two reset sources: software reset and hardware reset.

23.0.22.1 Software Reset Considerations

The transmitter and receiver portions of the McASP may be put in reset through the global control register (GBLCTL). Note that a valid serial clock must be supplied to the desired portion of the McASP (transmit and/or receive) in order to assert the software reset bits in GBLCTL. See Section 23.0.21.1.2 for details on how to ensure reset has occurred.

The entire McASP module may also be reset through the Power and Sleep Controller (PSC). Note that from the McASP perspective, this reset appears as a hardware reset to the entire module.

23.0.22.2 Hardware Reset Considerations

When the McASP is reset due to device reset, the entire serial port (including the transmitter and receiver state machines, and other registers) is reset.

23.0.23 EDMA Event Support

The McASP-related EDMA events are shown in Table 23-6.

Table 23-6. EDMA Events - McASP

Channel	Event Name	Event Description
0	AREVT0	McASP0 Receive Event
1	AXEVT0	McASP0 Transmit Event

23.0.24 Power Management

The McASP can be placed in reduced power modes to conserve power during periods of low activity. The power management of the peripheral is controlled by the processor Power and Sleep Controller (PSC). The PSC acts as a master controller for power management for all of the peripherals on the device. For information on power management procedures using the PSC, see the *Power and Sleep Controller (PSC)* chapter.

Copyright © 2013–2016, Texas Instruments Incorporated

23.1 Registers

Control registers for the McASP are summarized in Table 23-7. The control registers are accessed through the peripheral configuration port. The receive buffer registers (RBUF) and transmit buffer registers (XBUF) can also be accessed through the DMA port, as listed in Table 23-8. See your device-specific data manual for the memory address of these registers.

Control registers for the McASP Audio FIFO (AFIFO) are summarized in Table 23-9. Note that the AFIFO Write FIFO (WFIFO) and Read FIFO (RFIFO) have independent control and status registers. The AFIFO control registers are accessed through the peripheral configuration port. See your device-specific data manual for the memory address of these registers.

Table 23-7. McASP Registers Accessed by CPU/EDMA Through Peripheral Configuration Port

Offset	Acronym	Register Description	Section
0h	REV	Revision identification register	Section 23.1.2
10h	PFUNC	Pin function register	Section 23.1.3
14h	PDIR	Pin direction register	Section 23.1.4
18h	PDOUT	Pin data output register	Section 23.1.5
1Ch	PDIN	Read returns: Pin data input register	Section 23.1.6
1Ch	PDSET	Writes affect: Pin data set register (alternate write address: PDOUT)	Section 23.1.7
20h	PDCLR	Pin data clear register (alternate write address: PDOUT)	Section 23.1.8
44h	GBLCTL	Global control register	Section 23.1.9
48h	AMUTE	Audio mute control register	Section 23.1.10
4Ch	DLBCTL	Digital loopback control register	Section 23.1.11
50h	DITCTL	DIT mode control register	Section 23.1.12
60h	RGBLCTL	Receiver global control register: Alias of GBLCTL, only receive bits are affected - allows receiver to be reset independently from transmitter	Section 23.1.13
64h	RMASK	Receive format unit bit mask register	Section 23.1.14
68h	RFMT	Receive bit stream format register	Section 23.1.15
6Ch	AFSRCTL	Receive frame sync control register	Section 23.1.16
70h	ACLKRCTL	Receive clock control register	Section 23.1.17
74h	AHCLKRCTL	Receive high-frequency clock control register	Section 23.1.18
78h	RTDM	Receive TDM time slot 0-31 register	Section 23.1.19
7Ch	RINTCTL	Receiver interrupt control register	Section 23.1.20
80h	RSTAT	Receiver status register	Section 23.1.21
84h	RSLOT	Current receive TDM time slot register	Section 23.1.22
88h	RCLKCHK	Receive clock check control register	Section 23.1.23
8Ch	REVTCTL	Receiver DMA event control register	Section 23.1.24
A0h	XGBLCTL	Transmitter global control register. Alias of GBLCTL, only transmit bits are affected - allows transmitter to be reset independently from receiver	Section 23.1.25
A4h	XMASK	Transmit format unit bit mask register	Section 23.1.26
A8h	XFMT	Transmit bit stream format register	Section 23.1.27
ACh	AFSXCTL	Transmit frame sync control register	Section 23.1.28
B0h	ACLKXCTL	Transmit clock control register	Section 23.1.29
B4h	AHCLKXCTL	Transmit high-frequency clock control register	Section 23.1.30
B8h	XTDM	Transmit TDM time slot 0-31 register	Section 23.1.31
BCh	XINTCTL	Transmitter interrupt control register	Section 23.1.32
C0h	XSTAT	Transmitter status register	Section 23.1.33
C4h	XSLOT	Current transmit TDM time slot register	Section 23.1.34
C8h	XCLKCHK	Transmit clock check control register	Section 23.1.35
CCh	XEVTCTL	Transmitter DMA event control register	Section 23.1.36

Copyright © 2013–2016, Texas Instruments Incorporated

Table 23-7. McASP Registers Accessed by CPU/EDMA Through Peripheral Configuration Port (continued)

Offset	Acronym	Register Description	Section
100h	DITCSRA0	Left (even TDM time slot) channel status register (DIT mode) 0	Section 23.1.38
104h	DITCSRA1	Left (even TDM time slot) channel status register (DIT mode) 1	Section 23.1.38
108h	DITCSRA2	Left (even TDM time slot) channel status register (DIT mode) 2	Section 23.1.38
10Ch	DITCSRA3	Left (even TDM time slot) channel status register (DIT mode) 3	Section 23.1.38
110h	DITCSRA4	Left (even TDM time slot) channel status register (DIT mode) 4	Section 23.1.38
114h	DITCSRA5	Left (even TDM time slot) channel status register (DIT mode) 5	Section 23.1.38
118h	DITCSRB0	Right (odd TDM time slot) channel status register (DIT mode) 0	Section 23.1.39
11Ch	DITCSRB1	Right (odd TDM time slot) channel status register (DIT mode) 1	Section 23.1.39
120h	DITCSRB2	Right (odd TDM time slot) channel status register (DIT mode) 2	Section 23.1.39
124h	DITCSRB3	Right (odd TDM time slot) channel status register (DIT mode) 3	Section 23.1.39
128h	DITCSRB4	Right (odd TDM time slot) channel status register (DIT mode) 4	Section 23.1.39
12Ch	DITCSRB5	Right (odd TDM time slot) channel status register (DIT mode) 5	Section 23.1.39
130h	DITUDRA0	Left (even TDM time slot) channel user data register (DIT mode) 0	Section 23.1.40
134h	DITUDRA1	Left (even TDM time slot) channel user data register (DIT mode) 1	Section 23.1.40
138h	DITUDRA2	Left (even TDM time slot) channel user data register (DIT mode) 2	Section 23.1.40
13Ch	DITUDRA3	Left (even TDM time slot) channel user data register (DIT mode) 3	Section 23.1.40
140h	DITUDRA4	Left (even TDM time slot) channel user data register (DIT mode) 4	Section 23.1.40
144h	DITUDRA5	Left (even TDM time slot) channel user data register (DIT mode) 5	Section 23.1.40
148h	DITUDRB0	Right (odd TDM time slot) channel user data register (DIT mode) 0	Section 23.1.41
14Ch	DITUDRB1	Right (odd TDM time slot) channel user data register (DIT mode) 1	Section 23.1.41
150h	DITUDRB2	Right (odd TDM time slot) channel user data register (DIT mode) 2	Section 23.1.41
154h	DITUDRB3	Right (odd TDM time slot) channel user data register (DIT mode) 3	Section 23.1.41
158h	DITUDRB4	Right (odd TDM time slot) channel user data register (DIT mode) 4	Section 23.1.41
15Ch	DITUDRB5	Right (odd TDM time slot) channel user data register (DIT mode) 5	Section 23.1.41
180h	SRCTL0	Serializer control register 0	Section 23.1.37
184h	SRCTL1	Serializer control register 1	Section 23.1.37
188h	SRCTL2	Serializer control register 2	Section 23.1.37
18Ch	SRCTL3	Serializer control register 3	Section 23.1.37
190h	SRCTL4	Serializer control register 4	Section 23.1.37
194h	SRCTL5	Serializer control register 5	Section 23.1.37
198h	SRCTL6	Serializer control register 6	Section 23.1.37
19Ch	SRCTL7	Serializer control register 7	Section 23.1.37
1A0h	SRCTL8	Serializer control register 8	Section 23.1.37
1A4h	SRCTL9	Serializer control register 9	Section 23.1.37
1A8h	SRCTL10	Serializer control register 10	Section 23.1.37
1ACh	SRCTL11	Serializer control register 11	Section 23.1.37
1B0h	SRCTL12	Serializer control register 12	Section 23.1.37
1B4h	SRCTL13	Serializer control register 13	Section 23.1.37
1B8h	SRCTL14	Serializer control register 14	Section 23.1.37
1BCh	SRCTL15	Serializer control register 15	Section 23.1.37

Copyright © 2013–2016, Texas Instruments Incorporated

Table 23-7. McASP Registers Accessed by CPU/EDMA Through Peripheral Configuration Port (continued)

Offset	Acronym	Register Description	Section
200h	XBUF0[1]	Transmit buffer register for serializer 0	Section 23.1.42
204h	XBUF1[1]	Transmit buffer register for serializer 1	Section 23.1.42
208h	XBUF2[1]	Transmit buffer register for serializer 2	Section 23.1.42
20Ch	XBUF3[1]	Transmit buffer register for serializer 3	Section 23.1.42
210h	XBUF4[1]	Transmit buffer register for serializer 4	Section 23.1.42
214h	XBUF5[1]	Transmit buffer register for serializer 5	Section 23.1.42
218h	XBUF6[1]	Transmit buffer register for serializer 6	Section 23.1.42
21Ch	XBUF7[1]	Transmit buffer register for serializer 7	Section 23.1.42
220h	XBUF8[1]	Transmit buffer register for serializer 8	Section 23.1.42
224h	XBUF9[1]	Transmit buffer register for serializer 9	Section 23.1.42
228h	XBUF10[1]	Transmit buffer register for serializer 10	Section 23.1.42
22Ch	XBUF11[1]	Transmit buffer register for serializer 11	Section 23.1.42
230h	XBUF12[1]	Transmit buffer register for serializer 12	Section 23.1.42
234h	XBUF13[1]	Transmit buffer register for serializer 13	Section 23.1.42
238h	XBUF14[1]	Transmit buffer register for serializer 14	Section 23.1.42
23Ch	XBUF15[1]	Transmit buffer register for serializer 15	Section 23.1.42
280h	RBUF0[2]	Receive buffer register for serializer 0	Section 23.1.43
284h	RBUF1[2]	Receive buffer register for serializer 1	Section 23.1.43
288h	RBUF2[2]	Receive buffer register for serializer 2	Section 23.1.43
28Ch	RBUF3[2]	Receive buffer register for serializer 3	Section 23.1.43
290h	RBUF4[2]	Receive buffer register for serializer 4	Section 23.1.43
294h	RBUF5[2]	Receive buffer register for serializer 5	Section 23.1.43
298h	RBUF6[2]	Receive buffer register for serializer 6	Section 23.1.43
29Ch	RBUF7[2]	Receive buffer register for serializer 7	Section 23.1.43
2A0h	RBUF8[2]	Receive buffer register for serializer 8	Section 23.1.43
2A4h	RBUF9[2]	Receive buffer register for serializer 9	Section 23.1.43
2A8h	RBUF10[2]	Receive buffer register for serializer 10	Section 23.1.43
2ACh	RBUF11[2]	Receive buffer register for serializer 11	Section 23.1.43
2B0h	RBUF12[2]	Receive buffer register for serializer 12	Section 23.1.43
2B4h	RBUF13[2]	Receive buffer register for serializer 13	Section 23.1.43
2B8h	RBUF14[2]	Receive buffer register for serializer 14	Section 23.1.43
2BCh	RBUF15[2]	Receive buffer register for serializer 15	Section 23.1.43

(1) Writes to XRBUF[n] by way of XBUF*n* by the CPU/EDMA can only occur through the peripheral configuration port when XBUSEL = 1 in XFMT.

(2) Reads from XRBUF[n] by way of RBUF*n* by the CPU/EDMA can only occur through the peripheral configuration port when RBUSEL = 1 in RFMT.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 23-8. McASP Registers Accessed by CPU/EDMA Through DMA Port

Offset[1]	Access	Acronym	Register Description
2000h	Read Accesses	RBUF	Receive buffer DMA port address. Cycles through receive serializers, skipping over transmit serializers and inactive serializers. Starts at the lowest serializer at the beginning of each time slot. Reads from XRBUF[n] by way of RBUF by the CPU/EDMA can only occur through the DMA port when RBUSEL = 0 in RFMT.
2000h	Write Accesses	XBUF	Transmit buffer DMA port address. Cycles through transmit serializers, skipping over receive and inactive serializers. Starts at the lowest serializer at the beginning of each time slot. Writes to XRBUF[n] by way of XBUF by the CPU/EDMA can only occur through the DMA port when XBUSEL = 0 in XFMT.

(1) RBUF and XBUF are at the same address location. Reads access RBUF and writes access XBUF.

Table 23-9. McASP AFIFO Registers Accessed Through Peripheral Configuration Port[1]

Offset	Acronym	Register Description	Section
1000h	AFIFOREV	AFIFO revision identification register	Section 23.1.44
1010h	WFIFOCTL	Write FIFO control register	Section 23.1.45
1014h	WFIFOSTS	Write FIFO status register	Section 23.1.46
1018h	RFIFOCTL	Read FIFO control register	Section 23.1.47
101Ch	RFIFOSTS	Read FIFO status register	Section 23.1.48

(1) The AFIFO cannot be used with the peripheral configuration port. Only the DMA port has access to the AFIFO.

23.1.1 Register Bit Restrictions

Some bit fields (see Bits With Restrictions on When They May be Changed) have restrictions on when they may be changed. These restrictions take the form of certain registers that must be asserted in GBLCTL. Once these registers have been asserted, the user may then, and only then, change the desired bit field.

Bits With Restrictions on When They May be Changed

To Change Register:	To Change Bit Field:	... these registers must be asserted in GBLCTL									
		HCLKRRST	RGRST	RSRCLR	RSMRST	RFRST	HCLKXRST	XGRST	XSRCLR	XSMRST	XFRST
DITCTL	DITEN									x	x
XFMT	XSSZ									x	
XFMT	XDATDLY				x					x	
RFMT	RSSZ				x						
RFMT	RDATDLY				x						
AFSXCTL	FSXP									x	x
AFSXCTL	FSXM									x	x
AFSXCTL	FXWID									x	x
AFSXCTL	XMOD									x	x
AFSRCTL	FSRP				x	x					
AFSRCTL	FSRM				x	x					
AFSRCTL	FRWID				x	x					
AFSRCTL	RMOD				x	x					
ACLKXCTL	CLKXDIV							x	x	x	x
ACLKXCTL	CLKXM								x	x	x
ACLKXCTL	ASYNC				x	x					
ACLKXCTL	CLKXP								x	x	x
ACLKRCTL	CLKRDIV		x	x	x	x					
ACLKRCTL	CLKRM			x	x	x					
ACLKRCTL	CLKRP			x	x	x					

Copyright © 2013–2016, Texas Instruments Incorporated

Bits With Restrictions on When They May be Changed (continued)

To Change Register:	To Change Bit Field:	... these registers must be asserted in GBLCTL									
		HCLKRRST	RGRST	RSRCLR	RSMRST	RFRST	HCLKXRST	XGRST	XSRCLR	XSMRST	XFRST
AHCLKXCTL	HCLKXDIV						x	x	x	x	x
AHCLKXCTL	HCLKXP						x	x	x	x	x
AHCLKXCTL	HCLKXM						x	x	x	x	x
AHCLKRCTL	HCLKRDIV	x	x	x	x	x					
AHCLKRCTL	HCLKRP	x	x	x	x	x					
AHCLKRCTL	HCLKRM	x	x	x	x	x					
DLBCTL	DLBEN			x	x	x			x	x	x
DLBCTL	ORD			x	x	x			x	x	x
DLBCTL	MODE			x	x	x			x	x	x

23.1.2 Revision Identification Register (REV)

The revision identification register (REV) contains revision data for the peripheral. The REV is shown in Figure 23-35 and described in Table 23-10.

Figure 23-35. Revision Identification Register (REV)

31	0
REV	
R-4430 0A02h	

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

Table 23-10. Revision Identification Register (REV) Field Descriptions

Bit	Field	Value	Description
31-0	REV	4430 0A02h	Identifies revision of peripheral.

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.3 Pin Function Register (PFUNC)

The pin function register (PFUNC) specifies the function of AXR[n], ACLKX, AHCLKX, AFSX, ACLKR, AHCLKR, and AFSR pins as either a McASP pin or a general-purpose input/output (GPIO) pin. The PFUNC is shown in Figure 23-36 and described in Table 23-11.

CAUTION

Writing to Reserved Bits

Writing a value other than 0 to reserved bits in this register may cause improper device operation.

Figure 23-36. Pin Function Register (PFUNC)

31	30	29	28	27	26	25	24
AFSR	AHCLKR	ACLKR	AFSX	AHCLKX	ACLKX	AMUTE	Reserved (A)
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

23	16
Reserved (A)	
R-0	

15	14	13	12	11	10	9	8
AXR15	AXR14	AXR13	AXR12	AXR11	AXR10	AXR9	AXR8
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
AXR7	AXR6	AXR5	AXR4	AXR3	AXR2	AXR1	AXR0
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

A If writing to this field, always write the default value for future device compatibility.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 23-11. Pin Function Register (PFUNC) Field Descriptions

Bit	Field	Value	Description
31	AFSR		Determines if AFSR pin functions as McASP or GPIO.
		0	Pin functions as McASP pin.
		1	Pin functions as GPIO pin.
30	AHCLKR		Determines if AHCLKR pin functions as McASP or GPIO.
		0	Pin functions as McASP pin.
		1	Pin functions as GPIO pin.
29	ACLKR		Determines if ACLKR pin functions as McASP or GPIO.
		0	Pin functions as McASP pin.
		1	Pin functions as GPIO pin.
28	AFSX		Determines if AFSX pin functions as McASP or GPIO.
		0	Pin functions as McASP pin.
		1	Pin functions as GPIO pin.
27	AHCLKX		Determines if AHCLKX pin functions as McASP or GPIO.
		0	Pin functions as McASP pin.
		1	Pin functions as GPIO pin.
26	ACLKX		Determines if ACLKX pin functions as McASP or GPIO.
		0	Pin functions as McASP pin.
		1	Pin functions as GPIO pin.
25	AMUTE		Determines if AMUTE pin functions as McASP or GPIO.
		0	Pin functions as McASP pin.
		1	Pin functions as GPIO pin.
24-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15-0	AXR[15-0]		Determines if AXR[n] pin functions as McASP or GPIO.
		0	Pin functions as McASP pin.
		1	Pin functions as GPIO pin.

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.4 Pin Direction Register (PDIR)

The pin direction register (PDIR) specifies the direction of AXR[n], ACLKX, AHCLKX, AFSX, ACLKR, AHCLKR, and AFSR pins as either an input or an output pin. The PDIR is shown in Figure 23-37 and described in Table 23-12.

Regardless of the pin function register (PFUNC) setting, each PDIR bit must be set to 1 for the specified pin to be enabled as an output and each PDIR bit must be cleared to 0 for the specified pin to be an input.

For example, if the McASP is configured to use an internally-generated bit clock and the clock is to be driven out to the system, the PFUNC bit must be cleared to 0 (McASP function) and the PDIR bit must be set to 1 (an output).

When AXR[n] is configured to transmit, the PFUNC bit must be cleared to 0 (McASP function) and the PDIR bit must be set to 1 (an output). Similarly, when AXR[n] is configured to receive, the PFUNC bit must be cleared to 0 (McASP function) and the PDIR bit must be cleared to 0 (an input).

CAUTION

Writing to Reserved Bits

Writing a value other than 0 to reserved bits in this register may cause improper device operation.

Figure 23-37. Pin Direction Register (PDIR)

31	30	29	28	27	26	25	24
AFSR	AHCLKR	ACLKR	AFSX	AHCLKX	ACLKX	AMUTE	Reserved (A)
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

23	16
Reserved (A)	
R-0	

15	14	13	12	11	10	9	8
AXR15	AXR14	AXR13	AXR12	AXR11	AXR10	AXR9	AXR8
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
AXR7	AXR6	AXR5	AXR4	AXR3	AXR2	AXR1	AXR0
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

A If writing to this field, always write the default value for future device compatibility.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 23-12. Pin Direction Register (PDIR) Field Descriptions

Bit	Field	Value	Description
31	AFSR		Determines if AFSR pin functions as an input or output.
		0	Pin functions as input.
		1	Pin functions as output.
30	AHCLKR		Determines if AHCLKR pin functions as an input or output.
		0	Pin functions as input.
		1	Pin functions as output.
29	ACLKR		Determines if ACLKR pin functions as an input or output.
		0	Pin functions as input.
		1	Pin functions as output.
28	AFSX		Determines if AFSX pin functions as an input or output.
		0	Pin functions as input.
		1	Pin functions as output.
27	AHCLKX		Determines if AHCLKX pin functions as an input or output.
		0	Pin functions as input.
		1	Pin functions as output.
26	ACLKX		Determines if ACLKX pin functions as an input or output.
		0	Pin functions as input.
		1	Pin functions as output.
25	AMUTE		Determines if AMUTE pin functions as an input or output.
		0	Pin functions as input.
		1	Pin functions as output.
24-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15-0	AXR[15-0]		Determines if AXR[n] pin functions as an input or output.
		0	Pin functions as input.
		1	Pin functions as output.

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.5 Pin Data Output Register (PDOUT)

The pin data output register (PDOUT) holds a value for data out at all times, and may be read back at all times. The value held by PDOUT is not affected by writing to PDIR and PFUNC. However, the data value in PDOUT is driven out onto the McASP pin only if the corresponding bit in PFUNC is set to 1 (GPIO function) and the corresponding bit in PDIR is set to 1 (output). When reading data, returns the corresponding bit value in PDOUT[n], does not return input from I/O pin; when writing data, writes to the corresponding PDOUT[n] bit. The PDOUT is shown in Figure 23-38 and described in Table 23-13.

PDOUT has these aliases or alternate addresses:

- PDSET - when written to at this address, writing a 1 to a bit in PDSET sets the corresponding bit in PDOUT to 1; writing a 0 has no effect and keeps the bits in PDOUT unchanged.
- PDCLR - when written to at this address, writing a 1 to a bit in PDCLR clears the corresponding bit in PDOUT to 0; writing a 0 has no effect and keeps the bits in PDOUT unchanged.

There is only one set of data out bits, PDOUT[31-0]. The other registers, PDSET and PDCLR, are just different addresses for the same control bits, with different behaviors during writes.

CAUTION

Writing to Reserved Bits

Writing a value other than 0 to reserved bits in this register may cause improper device operation.

Figure 23-38. Pin Data Output Register (PDOUT)

31	30	29	28	27	26	25	24
AFSR	AHCLKR	ACLKR	AFSX	AHCLKX	ACLKX	AMUTE	Reserved (A)
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

23	16
Reserved (A)	
R-0	

15	14	13	12	11	10	9	8
AXR15	AXR14	AXR13	AXR12	AXR11	AXR10	AXR9	AXR8
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
AXR7	AXR6	AXR5	AXR4	AXR3	AXR2	AXR1	AXR0
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

A If writing to this field, always write the default value for future device compatibility.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 23-13. Pin Data Output Register (PDOUT) Field Descriptions

Bit	Field	Value	Description
31	AFSR		Determines drive on AFSR output pin when the corresponding PFUNC[31] and PDIR[31] bits are set to 1.
		0	Pin drives low.
		1	Pin drives high.
30	AHCLKR		Determines drive on AHCLKR output pin when the corresponding PFUNC[30] and PDIR[30] bits are set to 1.
		0	Pin drives low.
		1	Pin drives high.
29	ACLKR		Determines drive on ACLKR output pin when the corresponding PFUNC[29] and PDIR[29] bits are set to 1.
		0	Pin drives low.
		1	Pin drives high.
28	AFSX		Determines drive on AFSX output pin when the corresponding PFUNC[28] and PDIR[28] bits are set to 1.
		0	Pin drives low.
		1	Pin drives high.
27	AHCLKX		Determines drive on AHCLKX output pin when the corresponding PFUNC[27] and PDIR[27] bits are set to 1.
		0	Pin drives low.
		1	Pin drives high.
26	ACLKX		Determines drive on ACLKX output pin when the corresponding PFUNC[26] and PDIR[26] bits are set to 1.
		0	Pin drives low.
		1	Pin drives high.
25	AMUTE		Determines drive on AMUTE output pin when the corresponding PFUNC[25] and PDIR[25] bits are set to 1.
		0	Pin drives low.
		1	Pin drives high.
24-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15-0	AXR[15-0]		Determines drive on AXR[n] output pin when the corresponding PFUNC[n] and PDIR[n] bits are set to 1.
		0	Pin drives low.
		1	Pin drives high.

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.6 Pin Data Input Register (PDIN)

The pin data input register (PDIN) holds the I/O pin state of each of the McASP pins. PDIN allows the actual value of the pin to be read, regardless of the state of PFUNC and PDIR. The value after reset for registers 1 through 15 and 24 through 31 depends on how the pins are being driven. The PDIN is shown in Figure 23-39 and described in Table 23-14.

CAUTION

Writing to Reserved Bits

Writing a value other than 0 to reserved bits in this register may cause improper device operation.

Figure 23-39. Pin Data Input Register (PDIN)

31	30	29	28	27	26	25	24
AFSR	AHCLKR	ACLKR	AFSX	AHCLKX	ACLKX	AMUTE	Reserved (A)
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

23	16
Reserved (A)	
R-0	

15	14	13	12	11	10	9	8
AXR15	AXR14	AXR13	AXR12	AXR11	AXR10	AXR9	AXR8
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
AXR7	AXR6	AXR5	AXR4	AXR3	AXR2	AXR1	AXR0
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

A If writing to this field, always write the default value for future device compatibility.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 23-14. Pin Data Input Register (PDIN) Field Descriptions

Bit	Field	Value	Description
31	AFSR		Logic level on AFSR pin.
		0	Pin is logic low.
		1	Pin is logic high.
30	AHCLKR		Logic level on AHCLKR pin.
		0	Pin is logic low.
		1	Pin is logic high.
29	ACLKR		Logic level on ACLKR pin.
		0	Pin is logic low.
		1	Pin is logic high.
28	AFSX		Logic level on AFSX pin.
		0	Pin is logic low.
		1	Pin is logic high.
27	AHCLKX		Logic level on AHCLKX pin.
		0	Pin is logic low.
		1	Pin is logic high.
26	ACLKX		Logic level on ACLKX pin.
		0	Pin is logic low.
		1	Pin is logic high.
25	AMUTE		Logic level on AMUTE pin.
		0	Pin is logic low.
		1	Pin is logic high.
24-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15-0	AXR[15-0]		Logic level on AXR[n] pin.
		0	Pin is logic low.
		1	Pin is logic high.

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.7 Pin Data Set Register (PDSET)

The pin data set register (PDSET) is an alias of the pin data output register (PDOUT) for writes only. Writing a 1 to the PDSET bit sets the corresponding bit in PDOUT and, if PFUNC = 1 (GPIO function) and PDIR = 1 (output), drives a logic high on the pin. PDSET is useful for a multitasking system because it allows you to set to a logic high only the desired pin(s) within a system without affecting other I/O pins controlled by the same McASP. The PDSET is shown in Figure 23-40 and described in Table 23-15.

CAUTION

Writing to Reserved Bits

Writing a value other than 0 to reserved bits in this register may cause improper device operation.

Figure 23-40. Pin Data Set Register (PDSET)

31	30	29	28	27	26	25	24
AFSR	AHCLKR	ACLKR	AFSX	AHCLKX	ACLKX	AMUTE	Reserved (A)
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

23	16
Reserved (A)	
R-0	

15	14	13	12	11	10	9	8
AXR15	AXR14	AXR13	AXR12	AXR11	AXR10	AXR9	AXR8
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
AXR7	AXR6	AXR5	AXR4	AXR3	AXR2	AXR1	AXR0
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

A If writing to this field, always write the default value for future device compatibility.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 23-15. Pin Data Set Register (PDSET) Field Descriptions

Bit	Field	Value	Description
31	AFSR		Allows the corresponding AFSR bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[31] bit is set to 1.
30	AHCLKR		Allows the corresponding AHCLKR bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[30] bit is set to 1.
29	ACLKR		Allows the corresponding ACLKR bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[29] bit is set to 1.
28	AFSX		Allows the corresponding AFSX bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[28] bit is set to 1.
27	AHCLKX		Allows the corresponding AHCLKX bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[27] bit is set to 1.
26	ACLKX		Allows the corresponding ACLKX bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[26] bit is set to 1.
25	AMUTE		Allows the corresponding AMUTE bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[25] bit is set to 1.
24-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15-0	AXR[15-0]		Allows the corresponding AXR[n] bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[n] bit is set to 1.

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.8 Pin Data Clear Register (PDCLR)

The pin data clear register (PDCLR) is an alias of the pin data output register (PDOUT) for writes only. Writing a 1 to the PDCLR bit clears the corresponding bit in PDOUT and, if PFUNC = 1 (GPIO function) and PDIR = 1 (output), drives a logic low on the pin. PDCLR is useful for a multitasking system because it allows you to clear to a logic low only the desired pin(s) within a system without affecting other I/O pins controlled by the same McASP. The PDCLR is shown in Figure 23-41 and described in Table 23-16.

CAUTION

Writing to Reserved Bits

Writing a value other than 0 to reserved bits in this register may cause improper device operation.

Figure 23-41. Pin Data Clear Register (PDCLR)

31	30	29	28	27	26	25	24
AFSR	AHCLKR	ACLKR	AFSX	AHCLKX	ACLKX	AMUTE	Reserved (A)
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

23	16
Reserved (A)	
R-0	

15	14	13	12	11	10	9	8
AXR15	AXR14	AXR13	AXR12	AXR11	AXR10	AXR9	AXR8
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
AXR7	AXR6	AXR5	AXR4	AXR3	AXR2	AXR1	AXR0
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

A If writing to this field, always write the default value for future device compatibility.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 23-16. Pin Data Clear Register (PDCLR) Field Descriptions

Bit	Field	Value	Description
31	AFSR		Allows the corresponding AFSR bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[31] bit is cleared to 0.
30	AHCLKR		Allows the corresponding AHCLKR bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[30] bit is cleared to 0.
29	ACLKR		Allows the corresponding ACLKR bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[29] bit is cleared to 0.
28	AFSX		Allows the corresponding AFSX bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[28] bit is cleared to 0.
27	AHCLKX		Allows the corresponding AHCLKX bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[27] bit is cleared to 0.
26	ACLKX		Allows the corresponding ACLKX bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[26] bit is cleared to 0.
25	AMUTE		Allows the corresponding AMUTE bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[25] bit is cleared to 0.
24-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15-0	AXR[15-0]		Allows the corresponding AXR[n] bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port.
		0	No effect.
		1	PDOUT[n] bit is cleared to 0.

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.9 Global Control Register (GBLCTL)

The global control register (GBLCTL) provides initialization of the transmit and receive sections. The GBLCTL is shown in Figure 23-42 and described in Table 23-17.

The bit fields in GBLCTL are synchronized and latched by the corresponding clocks (ACLKX for bits 12-8 and ACLKR for bits 4-0). Before GBLCTL is programmed, you must ensure that serial clocks are running. If the corresponding external serial clocks, ACLKX and ACLKR, are not yet running, you should select the internal serial clock source in AHCLKXCTL, AHCLKRCTL, ACLKXCTL, and ACLKRCTL before GBLCTL is programmed. Also, after programming any bits in GBLCTL you should not proceed until you have read back from GBLCTL and verified that the bits are latched in GBLCTL.

Figure 23-42. Global Control Register (GBLCTL)

31	16
Reserved (A)	
R-0	

15	13	12	11	10	9	8
Reserved (A)		XFRST	XSMRST	XSRCLR	XHCLKRST	XCLKRST
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	5	4	3	2	1	0
Reserved (A)		RFRST	RSMRST	RSRCLR	RHCLKRST	RCLKRST
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

A If writing to this field, always write the default value for future device compatibility.

Table 23-17. Global Control Register (GBLCTL) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
12	XFRST		Transmit frame sync generator reset enable bit.
		0	Transmit frame sync generator is reset.
		1	Transmit frame sync generator is active. When released from reset, the transmit frame sync generator begins counting serial clocks and generating frame sync as programmed.
11	XSMRST		Transmit state machine reset enable bit.
		0	Transmit state machine is held in reset. AXR[n] pin state: If PFUNC[n] = 0 and PDIR[n] = 1; then the serializer drives the AXR[n] pin to the state specified for inactive time slot (as determined by DISMOD bits in SRCTL).
		1	Transmit state machine is released from reset. When released from reset, the transmit state machine immediately transfers data from XRBUF[n] to XRSR[n]. The transmit state machine sets the underrun flag (XUNDRN) in XSTAT, if XRBUF[n] have not been preloaded with data before reset is released. The transmit state machine also immediately begins detecting frame sync and is ready to transmit. Transmit TDM time slot begins at slot 0 after reset is released.
10	XSRCLR		Transmit serializer clear enable bit. By clearing then setting this bit, the transmit buffer is flushed to an empty state (XDATA = 1). If XSMRST = 1, XSRCLR = 1, XDATA = 1, and XBUF is not loaded with new data before the start of the next active time slot, an underrun will occur.
		0	Transmit serializers are cleared.
		1	Transmit serializers are active. When the transmit serializers are first taken out of reset (XSRCLR changes from 0 to 1), the transmit data ready bit (XDATA) in XSTAT is set to indicate XBUF is ready to be written.
9	XHCLKRST		Transmit high-frequency clock divider reset enable bit.
		0	Transmit high-frequency clock divider is held in reset.
		1	Transmit high-frequency clock divider is running.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 23-17. Global Control Register (GBLCTL) Field Descriptions (continued)

Bit	Field	Value	Description
8	XCLKRST		Transmit clock divider reset enable bit.
		0	Transmit clock divider is held in reset. When the clock divider is in reset, it passes through a divide-by-1 of its input.
		1	Transmit clock divider is running.
7-5	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
4	RFRST		Receive frame sync generator reset enable bit.
		0	Receive frame sync generator is reset.
		1	Receive frame sync generator is active. When released from reset, the receive frame sync generator begins counting serial clocks and generating frame sync as programmed.
3	RSMRST		Receive state machine reset enable bit.
		0	Receive state machine is held in reset.
		1	Receive state machine is released from reset. When released from reset, the receive state machine immediately begins detecting frame sync and is ready to receive. Receive TDM time slot begins at slot 0 after reset is released.
2	RSRCLR		Receive serializer clear enable bit. By clearing then setting this bit, the receive buffer is flushed.
		0	Receive serializers are cleared.
		1	Receive serializers are active.
1	RHCLKRST		Receive high-frequency clock divider reset enable bit.
		0	Receive high-frequency clock divider is held in reset.
		1	Receive high-frequency clock divider is running.
0	RCLKRST		Receive clock divider reset enable bit.
		0	Receive clock divider is held in reset. When the clock divider is in reset, it passes through a divide-by-1 of its input.
		1	Receive clock divider is running.

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.10 Audio Mute Control Register (AMUTE)

The audio mute control register (AMUTE) controls the McASP audio mute (AMUTE) output pin. The value after reset for register 4 depends on how the pins are being driven. The AMUTE is shown in Figure 23-43 and described in Table 23-18.

Figure 23-43. Audio Mute Control Register (AMUTE)

31							16
Reserved [A]							
R-0							
15		13	12	11	10	9	8
Reserved [A]			XDMAERR	RDMAERR	XCKFAIL	RCKFAIL	XSYNCERR
R-0			R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
RSYNCERR	XUNDRN	ROVRN	INSTAT	INEN	INPOL	MUTEN	
R/W-0	R/W-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

A If writing to this field, always write the default value for future device compatibility.

Table 23-18. Audio Mute Control Register (AMUTE) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
12	XDMAERR		If transmit DMA error (XDMAERR), drive AMUTE active enable bit.
		0	Drive is disabled. Detection of transmit DMA error is ignored by AMUTE.
		1	Drive is enabled (active). Upon detection of transmit DMA error, AMUTE is active and is driven according to MUTEN bit.
11	RDMAERR		If receive DMA error (RDMAERR), drive AMUTE active enable bit.
		0	Drive is disabled. Detection of receive DMA error is ignored by AMUTE.
		1	Drive is enabled (active). Upon detection of receive DMA error, AMUTE is active and is driven according to MUTEN bit.
10	XCKFAIL		If transmit clock failure (XCKFAIL), drive AMUTE active enable bit.
		0	Drive is disabled. Detection of transmit clock failure is ignored by AMUTE.
		1	Drive is enabled (active). Upon detection of transmit clock failure, AMUTE is active and is driven according to MUTEN bit
9	RCKFAIL		If receive clock failure (RCKFAIL), drive AMUTE active enable bit.
		0	Drive is disabled. Detection of receive clock failure is ignored by AMUTE.
		1	Drive is enabled (active). Upon detection of receive clock failure, AMUTE is active and is driven according to MUTEN bit.
8	XSYNCERR		If unexpected transmit frame sync error (XSYNCERR), drive AMUTE active enable bit.
		0	Drive is disabled. Detection of unexpected transmit frame sync error is ignored by AMUTE.
		1	Drive is enabled (active). Upon detection of unexpected transmit frame sync error, AMUTE is active and is driven according to MUTEN bit.
7	RSYNCERR		If unexpected receive frame sync error (RSYNCERR), drive AMUTE active enable bit.
		0	Drive is disabled. Detection of unexpected receive frame sync error is ignored by AMUTE.
		1	Drive is enabled (active). Upon detection of unexpected receive frame sync error, AMUTE is active and is driven according to MUTEN bit.
6	XUNDRN		If transmit underrun error (XUNDRN), drive AMUTE active enable bit.
		0	Drive is disabled. Detection of transmit underrun error is ignored by AMUTE.
		1	Drive is enabled (active). Upon detection of transmit underrun error, AMUTE is active and is driven according to MUTEN bit.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 23-18. Audio Mute Control Register (AMUTE) Field Descriptions (continued)

Bit	Field	Value	Description
5	ROVRN		If receiver overrun error (ROVRN), drive AMUTE active enable bit.
		0	Drive is disabled. Detection of receiver overrun error is ignored by AMUTE.
		1	Drive is enabled (active). Upon detection of receiver overrun error, AMUTE is active and is driven according to MUTEN bit.
4	INSTAT		Determines drive on AXR[n] pin when PFUNC[n] and PDIR[n] bits are set to 1.
		0	AMUTEIN pin is inactive.
		1	AMUTEIN pin is active. Audio mute in error is detected.
3	INEN		Drive AMUTE active when AMUTEIN error is active (INSTAT = 1).
		0	Drive is disabled. AMUTEIN is ignored by AMUTE.
		1	Drive is enabled (active). INSTAT = 1 drives AMUTE active.
2	INPOL		Audio mute in (AMUTEIN) polarity select bit.
		0	Polarity is active high. A high on AMUTEIN sets INSTAT to 1.
		1	Polarity is active low. A low on AMUTEIN sets INSTAT to 1.
1-0	MUTEN	0-3h	AMUTE pin enable bit (unless overridden by GPIO registers).
		0	AMUTE pin is disabled, pin goes to tri-state condition.
		1h	AMUTE pin is driven high if error is detected.
		2h	AMUTE pin is driven low if error is detected.
		3h	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.11 Digital Loopback Control Register (DLBCTL)

The digital loopback control register (DLBCTL) controls the internal loopback settings of the McASP in TDM mode. The DLBCTL is shown in Figure 23-44 and described in Table 23-19.

Figure 23-44. Digital Loopback Control Register (DLBCTL)

31	16
Reserved (A)	
R-0	

15	4	3	2	1	0
Reserved (A)		MODE		ORD	DLBEN
R-0		R/W-0		R/W-0	R/W-0

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

A If writing to this field, always write the default value for future device compatibility.

Table 23-19. Digital Loopback Control Register (DLBCTL) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
3-2	MODE	0-3h	Loopback generator mode bits. Applies only when loopback mode is enabled (DLBEN = 1).
		0	Default and reserved on loopback mode (DLBEN = 1). When in non-loopback mode (DLBEN = 0), MODE should be left at default (00). When in loopback mode (DLBEN = 1), MODE = 00 is reserved and not applicable.
		1h	Transmit clock and frame sync generators used by both transmit and receive sections. When in loopback mode (DLBEN = 1), MODE must be 01.
		2h-3h	Reserved.
1	ORD		Loopback order bit when loopback mode is enabled (DLBEN = 1).
		0	Odd serializers N + 1 transmit to even serializers N that receive. The corresponding serializers must be programmed properly.
		1	Even serializers N transmit to odd serializers N+1 that receive. The corresponding serializers must be programmed properly.
0	DLBEN		Loopback mode enable bit.
		0	Loopback mode is disabled.
		1	Loopback mode is enabled.

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.12 Digital Mode Control Register (DITCTL)

The DIT mode control register (DITCTL) controls DIT operations of the McASP. The DITCTL is shown in Figure 23-45 and described in Table 23-20.

Figure 23-45. Digital Mode Control Register (DITCTL)

31	16
Reserved [A]	
R-0	

15	4	3	2	1	0
Reserved [A]		VB	VA	Rsvd [A]	DITEN
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

A If writing to this field, always write the default value for future device compatibility.

Table 23-20. Digital Mode Control Register (DITCTL) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
3	VB		Valid bit for odd time slots (DIT right subframe).
		0	V bit is 0 during odd DIT subframes.
		1	V bit is 1 during odd DIT subframes.
2	VA		Valid bit for even time slots (DIT left subframe).
		0	V bit is 0 during even DIT subframes.
		1	V bit is 1 during even DIT subframes.
1	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
0	DITEN		DIT mode enable bit. DITEN should only be changed while the XSMRST bit in GBLCTL is in reset (and for startup, XSRCLR also in reset). However, it is not necessary to reset the XCLKRST or XHCLKRST bits in GBLCTL to change DITEN.
		0	DIT mode is disabled. Transmitter operates in TDM or burst mode.
		1	DIT mode is enabled. Transmitter operates in DIT encoded mode.

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.13 Receiver Global Control Register (RGBLCTL)

Alias of the global control register (GBLCTL). Writing to the receiver global control register (RGBLCTL) affects only the receive bits of GBLCTL (bits 4-0). Reads from RGBLCTL return the value of GBLCTL. RGBLCTL allows the receiver to be reset independently from the transmitter. The RGBLCTL is shown in Figure 23-46 and described in Table 23-21. See Section 23.1.9 for a detailed description of GBLCTL.

Figure 23-46. Receiver Global Control Register (RGBLCTL)

31	16
Reserved (A)	
R-0	

15	13	12	11	10	9	8
Reserved (A)		XFRST	XSMRST	XSRCLR	XHCLKRST	XCLKRST
R-0		R-0	R-0	R-0	R-0	R-0

7	5	4	3	2	1	0
Reserved (A)		RFRST	RSMRST	RSRCLR	RHCLKRST	RCLKRST
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

A If writing to this field, always write the default value for future device compatibility.

Table 23-21. Receiver Global Control Register (RGBLCTL) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
12	XFRST	x	Transmit frame sync generator reset enable bit. A read of this bit returns the XFRST bit value of GBLCTL. Writes have no effect.
11	XSMRST	x	Transmit state machine reset enable bit. A read of this bit returns the XSMRST bit value of GBLCTL. Writes have no effect.
10	XSRCLR	x	Transmit serializer clear enable bit. A read of this bit returns the XSRCLR bit value of GBLCTL. Writes have no effect.
9	XHCLKRST	x	Transmit high-frequency clock divider reset enable bit. A read of this bit returns the XHCLKRST bit value of GBLCTL. Writes have no effect.
8	XCLKRST	x	Transmit clock divider reset enable bit. a read of this bit returns the XCLKRST bit value of GBLCTL. Writes have no effect.
7-5	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
4	RFRST		Receive frame sync generator reset enable bit. A write to this bit affects the RFRST bit of GBLCTL.
		0	Receive frame sync generator is reset.
		1	Receive frame sync generator is active.
3	RSMRST		Receive state machine reset enable bit. A write to this bit affects the RSMRST bit of GBLCTL.
		0	Receive state machine is held in reset.
		1	Receive state machine is released from reset.
2	RSRCLR		Receive serializer clear enable bit. A write to this bit affects the RSRCLR bit of GBLCTL.
		0	Receive serializers are cleared.
		1	Receive serializers are active.
1	RHCLKRST		Receive high-frequency clock divider reset enable bit. A write to this bit affects the RHCLKRST bit of GBLCTL.
		0	Receive high-frequency clock divider is held in reset.
		1	Receive high-frequency clock divider is running.
0	RCLKRST		Receive clock divider reset enable bit. A write to this bit affects the RCLKRST bit of GBLCTL.
		0	Receive clock divider is held in reset.
		1	Receive clock divider is running.

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.14 Receive Format Unit Bit Mask Register (RMASK)

The receive format unit bit mask register (RMASK) determines which bits of the received data are masked off and padded with a known value before being read by the CPU or DMA. The RMASK is shown in Figure 23-47 and described in Table 23-22.

Figure 23-47. Receive Format Unit Bit Mask Register (RMASK)

31	30	29	28	27	26	25	24
RMASK31	RMASK30	RMASK29	RMASK28	RMASK27	RMASK26	RMASK25	RMASK24
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
RMASK23	RMASK22	RMASK21	RMASK20	RMASK19	RMASK18	RMASK17	RMASK16
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
RMASK15	RMASK14	RMASK13	RMASK12	RMASK11	RMASK10	RMASK9	RMASK8
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
RMASK7	RMASK6	RMASK5	RMASK4	RMASK3	RMASK2	RMASK1	RMASK0
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 23-22. Receive Format Unit Bit Mask Register (RMASK) Field Descriptions

Bit	Field	Value	Description
31-0	RMASK[31-0]		Receive data mask enable bit.
		0	Corresponding bit of receive data (after passing through reverse and rotate units) is masked out and then padded with the selected bit pad value (RPAD and RPBIT bits in RFMT).
		1	Corresponding bit of receive data (after passing through reverse and rotate units) is returned to CPU or DMA.

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.15 Receive Bit Stream Format Register (RFMT)

The receive bit stream format register (RFMT) configures the receive data format. The RFMT is shown in Figure 23-48 and described in Table 23-23.

Figure 23-48. Receive Bit Stream Format Register (RFMT)

31																		18	17	16
Reserved [A]																			RDATDLY	
R-0																			R/W-0	

15	14	13	12	8	7	4	3	2	0
RRVRS	RPAD		RPBIT		RSSZ		RBUSEL	RROT	
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

A If writing to this field, always write the default value for future device compatibility.

Table 23-23. Receive Bit Stream Format Register (RFMT) Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
17-16	RDATDLY	0-3h	Receive bit delay.
		0	0-bit delay. The first receive data bit, AXR[n], occurs in same ACLKR cycle as the receive frame sync (AFSR).
		1h	1-bit delay. The first receive data bit, AXR[n], occurs one ACLKR cycle after the receive frame sync (AFSR).
		2h	2-bit delay. The first receive data bit, AXR[n], occurs two ACLKR cycles after the receive frame sync (AFSR).
		3h	Reserved.
15	RRVRS		Receive serial bitstream order.
		0	Bitstream is LSB first. No bit reversal is performed in receive format bit reverse unit.
		1	Bitstream is MSB first. Bit reversal is performed in receive format bit reverse unit.
14-13	RPAD	0-3h	Pad value for extra bits in slot not belonging to the word. This field only applies to bits when RMASK[n] = 0.
		0	Pad extra bits with 0.
		1h	Pad extra bits with 1.
		2h	Pad extra bits with one of the bits from the word as specified by RPBIT bits.
		3h	Reserved.
12-8	RPBIT	0-1Fh	RPBIT value determines which bit (as read by the CPU or DMA from RBUF[n]) is used to pad the extra bits. This field only applies when RPAD = 2h.
		0	Pad with bit 0 value.
		1h-1Fh	Pad with bit 1 to bit 31 value.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 23-23. Receive Bit Stream Format Register (RFMT) Field Descriptions (continued)

Bit	Field	Value	Description
7-4	RSSZ	0-Fh	Receive slot size.
		0-2h	Reserved
		3h	Slot size is 8 bits.
		4h	Reserved
		5h	Slot size is 12 bits.
		6h	Reserved
		7h	Slot size is 16 bits.
		8h	Reserved
		9h	Slot size is 20 bits.
		Ah	Reserved
		Bh	Slot size is 24 bits
		Ch	Reserved
		Dh	Slot size is 28 bits.
		Eh	Reserved
		Fh	Slot size is 32 bits.
3	RBUSEL		Selects whether reads from serializer buffer XRBUF[n] by way of RBUF*n* by the CPU/EDMA occur through the peripheral configuration port or the DMA port.
		0	Reads from XRBUF[n] originate on the DMA port. Reads from XRBUF[n] on the peripheral configuration port are ignored.
		1	Reads from XRBUF[n] originate on the peripheral configuration port. Reads from XRBUF[n] on the DMA port are ignored.
2-0	RROT	0-7h	Right-rotation value for receive rotate right format unit.
		0	Rotate right by 0 (no rotation).
		1h	Rotate right by 4 bit positions.
		2h	Rotate right by 8 bit positions.
		3h	Rotate right by 12 bit positions.
		4h	Rotate right by 16 bit positions.
		5h	Rotate right by 20 bit positions.
		6h	Rotate right by 24 bit positions.
		7h	Rotate right by 28 bit positions.

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.16 Receive Frame Sync Control Register (AFSRCTL)

The receive frame sync control register (AFSRCTL) configures the receive frame sync (AFSR). The AFSRCTL is shown in Figure 23-49 and described in Table 23-24.

Figure 23-49. Receive Frame Sync Control Register (AFSRCTL)

31	16
Reserved [A]	
R-0	

15	7	6	5	4	3	2	1	0
RMOD		Reserved [A]		FRWID	Reserved [A]		FSRM	FSRP
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

A If writing to this field, always write the default value for future device compatibility.

Table 23-24. Receive Frame Sync Control Register (AFSRCTL) Field Descriptions

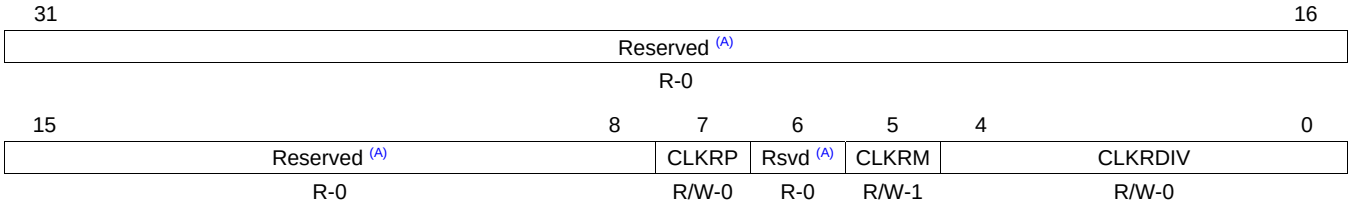
Bit	Field	Value	Description
31-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15-7	RMOD	0-1FFh	Receive frame sync mode select bits.
		0	Burst mode
		1h	Reserved
		2h-20h	2-slot TDM (I2S mode) to 32-slot TDM
		21h-17Fh	Reserved
		180h	384-slot TDM (external DIR IC inputting 384-slot DIR frames to McASP over I2S interface)
		181h-1FFh	Reserved
6-5	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
4	FRWID		Receive frame sync width select bit indicates the width of the receive frame sync (AFSR) during its active period.
		0	Single bit
		1	Single word
3-2	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
1	FSRM		Receive frame sync generation select bit.
		0	Externally-generated receive frame sync
		1	Internally-generated receive frame sync
0	FSRP		Receive frame sync polarity select bit.
		0	A rising edge on receive frame sync (AFSR) indicates the beginning of a frame.
		1	A falling edge on receive frame sync (AFSR) indicates the beginning of a frame.

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.17 Receive Clock Control Register (ACLKRCTL)

The receive clock control register (ACLKRCTL) configures the receive bit clock (ACLKR) and the receive clock generator. The ACLKRCTL is shown in Figure 23-50 and described in Table 23-25.

Figure 23-50. Receive Clock Control Register (ACLKRCTL)



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

A If writing to this field, always write the default value for future device compatibility.

Table 23-25. Receive Clock Control Register (ACLKRCTL) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
7	CLKRP		Receive bitstream clock polarity select bit. Note that this bitfield does not have any effect, if ACLKXCTL.ASYNC = 0 (see Section 23.1.29 for a description for the ASYNC bit).
		0	Falling edge. Receiver samples data on the falling edge of the serial clock, so the external transmitter driving this receiver must shift data out on the rising edge of the serial clock.
		1	Rising edge. Receiver samples data on the rising edge of the serial clock, so the external transmitter driving this receiver must shift data out on the falling edge of the serial clock.
6	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
5	CLKRM		Receive bit clock source bit. Note that this bitfield does not have any effect, if ACLKXCTL.ASYNC = 0 (see Section 23.1.29 for a description for the ASYNC bit).
		0	External receive clock source from ACLKR pin.
		1	Internal receive clock source from output of programmable bit clock divider.
4-0	CLKRDIV	0-1Fh	Receive bit clock divide ratio bits determine the divide-down ratio from AHCLKR to ACLKR. Note that this bitfield does not have any effect, if ACLKXCTL.ASYNC = 0 (see Section 23.1.29 for a description for the ASYNC bit).
		0	Divide-by-1
		1h	Divide-by-2
		2h-1Fh	Divide-by-3 to divide-by-32

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.18 Receive High-Frequency Clock Control Register (AHCLKRCTL)

The receive high-frequency clock control register (AHCLKRCTL) configures the receive high-frequency master clock (AHCLKR) and the receive clock generator. The AHCLKRCTL is shown in Figure 23-51 and described in Table 23-26.

Figure 23-51. Receive High-Frequency Clock Control Register (AHCLKRCTL)

31					16
Reserved [A]					
R-0					

15	14	13	12	11	0
HCLKRM	HCLKRP	Reserved [A]		HCLKRDIV	
R/W-1	R/W-0	R/W-0		R/W-0	

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

A If writing to this field, always write the default value for future device compatibility.

Table 23-26. Receive High-Frequency Clock Control Register (AHCLKRCTL) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15	HCLKRM		Receive high-frequency clock source bit.
		0	External receive high-frequency clock source from AHCLKR pin.
		1	Internal receive high-frequency clock source from output of programmable high clock divider.
14	HCLKRP		Receive bitstream high-frequency clock polarity select bit.
		0	Not inverted. AHCLKR is not inverted before programmable bit clock divider. In the special case where the receive bit clock (ACLKR) is internally generated and the programmable bit clock divider is set to divide-by-1 (CLKRDIV = 0 in ACLKRCTL), AHCLKR is directly passed through to the ACLKR pin.
		1	Inverted. AHCLKR is inverted before programmable bit clock divider. In the special case where the receive bit clock (ACLKR) is internally generated and the programmable bit clock divider is set to divide-by-1 (CLKRDIV = 0 in ACLKRCTL), AHCLKR is directly passed through to the ACLKR pin.
13-12	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
11-0	HCLKRDIV	0-FFFh	Receive high-frequency clock divide ratio bits determine the divide-down ratio from AUXCLK to AHCLKR.
		0	Divide-by-1
		1h	Divide-by-2
		2h-FFFh	Divide-by-3 to divide-by-4096

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.19 Receive TDM Time Slot Register (RTDM)

The receive TDM time slot register (RTDM) specifies which TDM time slot the receiver is active. The RTDM is shown in Figure 23-52 and described in Table 23-27.

Figure 23-52. Receive TDM Time Slot Register (RTDM)

31	30	29	28	27	26	25	24
RTDMS31	RTDMS30	RTDMS29	RTDMS28	RTDMS27	RTDMS26	RTDMS25	RTDMS24
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
RTDMS23	RTDMS22	RTDMS21	RTDMS20	RTDMS19	RTDMS18	RTDMS17	RTDMS16
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
RTDMS15	RTDMS14	RTDMS13	RTDMS12	RTDMS11	RTDMS10	RTDMS9	RTDMS8
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
RTDMS7	RTDMS6	RTDMS5	RTDMS4	RTDMS3	RTDMS2	RTDMS1	RTDMS0
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 23-27. Receive TDM Time Slot Register (RTDM) Field Descriptions

Bit	Field	Value	Description
31-0	RTDMS[31-0]		Receiver mode during TDM time slot *n*.
		0	Receive TDM time slot *n* is inactive. The receive serializer does not shift in data during this slot.
		1	Receive TDM time slot *n* is active. The receive serializer shifts in data during this slot.

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.20 Receiver Interrupt Control Register (RINTCTL)

The receiver interrupt control register (RINTCTL) controls generation of the McASP receive interrupt (RINT). When the register bit(s) is set to 1, the occurrence of the enabled McASP condition(s) generates RINT. The RINTCTL is shown in Figure 23-53 and described in Table 23-28. See Section 23.1.21 for a description of the interrupt conditions.

Figure 23-53. Receiver Interrupt Control Register (RINTCTL)

31							8
Reserved [A]							
R-0							
7	6	5	4	3	2	1	0
RSTAFRM	Reserved [A]	RDATA	RLAST	RDMAERR	RCKFAIL	RSYNCERR	ROVRN
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; R = Read only; -*n* = value after reset

A If writing to this field, always write the default value for future device compatibility.

Table 23-28. Receiver Interrupt Control Register (RINTCTL) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
7	RSTAFRM		Receive start of frame interrupt enable bit.
		0	Interrupt is disabled. A receive start of frame interrupt does not generate a McASP receive interrupt (RINT).
		1	Interrupt is enabled. A receive start of frame interrupt generates a McASP receive interrupt (RINT).
6	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
5	RDATA		Receive data ready interrupt enable bit.
		0	Interrupt is disabled. A receive data ready interrupt does not generate a McASP receive interrupt (RINT).
		1	Interrupt is enabled. A receive data ready interrupt generates a McASP receive interrupt (RINT).
4	RLAST		Receive last slot interrupt enable bit.
		0	Interrupt is disabled. A receive last slot interrupt does not generate a McASP receive interrupt (RINT).
		1	Interrupt is enabled. A receive last slot interrupt generates a McASP receive interrupt (RINT).
3	RDMAERR		Receive DMA error interrupt enable bit.
		0	Interrupt is disabled. A receive DMA error interrupt does not generate a McASP receive interrupt (RINT).
		1	Interrupt is enabled. A receive DMA error interrupt generates a McASP receive interrupt (RINT).
2	RCKFAIL		Receive clock failure interrupt enable bit.
		0	Interrupt is disabled. A receive clock failure interrupt does not generate a McASP receive interrupt (RINT).
		1	Interrupt is enabled. A receive clock failure interrupt generates a McASP receive interrupt (RINT).
1	RSYNCERR		Unexpected receive frame sync interrupt enable bit.
		0	Interrupt is disabled. An unexpected receive frame sync interrupt does not generate a McASP receive interrupt (RINT).
		1	Interrupt is enabled. An unexpected receive frame sync interrupt generates a McASP receive interrupt (RINT).
0	ROVRN		Receiver overrun interrupt enable bit.
		0	Interrupt is disabled. A receiver overrun interrupt does not generate a McASP receive interrupt (RINT).
		1	Interrupt is enabled. A receiver overrun interrupt generates a McASP receive interrupt (RINT).

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.21 Receiver Status Register (RSTAT)

The receiver status register (RSTAT) provides the receiver status and receive TDM time slot number. If the McASP logic attempts to set an interrupt flag in the same cycle that the CPU writes to the flag to clear it, the McASP logic has priority and the flag remains set. This also causes a new interrupt request to be generated. The RSTAT is shown in Figure 23-54 and described in Table 23-29.

Figure 23-54. Receiver Status Register (RSTAT)

31							9	8
Reserved (A)								RERR
R-0								R/W-0
7	6	5	4	3	2	1		0
RDMAERR	RSTAFRM	RDATA	RLAST	RTDMSLOT	RCKFAIL	RSYNCERR		ROVRN
R/W-0	R/W-0	R/W-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

A If writing to this field, always write the default value for future device compatibility.

Table 23-29. Receiver Status Register (RSTAT) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
8	RERR		RERR bit always returns a logic-OR of: ROVRN | RSYNCERR | RCKFAIL | RDMAERR Allows a single bit to be checked to determine if a receiver error interrupt has occurred.
		0	No errors have occurred.
		1	An error has occurred.
7	RDMAERR		Receive DMA error flag. RDMAERR is set when the CPU or DMA reads more serializers through the DMA port in a given time slot than were programmed as receivers. Causes a receive interrupt (RINT), if this bit is set and RDMAERR in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect.
		0	Receive DMA error did not occur.
		1	Receive DMA error did occur.
6	RSTAFRM		Receive start of frame flag. Causes a receive interrupt (RINT), if this bit is set and RSTAFRM in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect.
		0	No new receive frame sync (AFSR) is detected.
		1	A new receive frame sync (AFSR) is detected.
5	RDATA		Receive data ready flag. Causes a receive interrupt (RINT), if this bit is set and RDATA in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect.
		0	No new data in RBUF.
		1	Data is transferred from XRSR to RBUF and ready to be serviced by the CPU or DMA. When RDATA is set, it always causes a DMA event (AREVT).
4	RLAST		Receive last slot flag. RLAST is set along with RDATA, if the current slot is the last slot in a frame. Causes a receive interrupt (RINT), if this bit is set and RLAST in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect.
		0	Current slot is not the last slot in a frame.
		1	Current slot is the last slot in a frame. RDATA is also set.
3	RTDMSLOT		Returns the LSB of RSLOT. Allows a single read of RSTAT to determine whether the current TDM time slot is even or odd.
		0	Current TDM time slot is odd.
		1	Current TDM time slot is even.
2	RCKFAIL		Receive clock failure flag. RCKFAIL is set when the receive clock failure detection circuit reports an error (see Section 23.0.21.6.6). Causes a receive interrupt (RINT), if this bit is set and RCKFAIL in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect.
		0	Receive clock failure did not occur.
		1	Receive clock failure did occur.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 23-29. Receiver Status Register (RSTAT) Field Descriptions (continued)

Bit	Field	Value	Description
1	RSYNCERR		Unexpected receive frame sync flag. RSYNCERR is set when a new receive frame sync (AFSR) occurs before it is expected. Causes a receive interrupt (RINT), if this bit is set and RSYNCERR in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect.
		0	Unexpected receive frame sync did not occur.
		1	Unexpected receive frame sync did occur.
0	ROVRN		Receiver overrun flag. ROVRN is set when the receive serializer is instructed to transfer data from XRSR to RBUF, but the former data in RBUF has not yet been read by the CPU or DMA. Causes a receive interrupt (RINT), if this bit is set and ROVRN in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect.
		0	Receiver overrun did not occur.
		1	Receiver overrun did occur.

23.1.22 Current Receive TDM Time Slot Registers (RSLOT)

The current receive TDM time slot register (RSLOT) indicates the current time slot for the receive data frame. The RSLOT is shown in Figure 23-55 and described in Table 23-30.

Figure 23-55. Current Receive TDM Time Slot Registers (RSLOT)

31	16
Reserved (A)	
R-0	

15	9	8	0
Reserved (A)		RSLOTCNT	
R-0		R-0	

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

A If writing to this field, always write the default value for future device compatibility.

Table 23-30. Current Receive TDM Time Slot Registers (RSLOT) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
8-0	RSLOTCNT	0-17Fh	Current receive time slot count. Legal values: 0 to 383 (17Fh). TDM function is not supported for > 32 time slots. However, TDM time slot counter may count to 383 when used to receive a DIR block (transferred over TDM format).

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.23 Receive Clock Check Control Register (RCLKCHK)

The receive clock check control register (RCLKCHK) configures the receive clock failure detection circuit. The RCLKCHK is shown in Figure 23-56 and described in Table 23-31.

Figure 23-56. Receive Clock Check Control Register (RCLKCHK)

31	24	23	16
RCNT		RMAX	
R-0		R/W-0	

15	8	7	4	3	0
RMIN		Reserved (A)		RPS	
R/W-0		R-0		R/W-0	

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

A If writing to this field, always write the default value for future device compatibility.

Table 23-31. Receive Clock Check Control Register (RCLKCHK) Field Descriptions

Bit	Field	Value	Description
31-24	RCNT	0-FFh	Receive clock count value (from previous measurement). The clock circuit continually counts the number of CPU system clocks for every 32 receive high-frequency master clock (AHCLKR) signals, and stores the count in RCNT until the next measurement is taken.
23-16	RMAX	0-FFh	Receive clock maximum boundary. This 8-bit unsigned value sets the maximum allowed boundary for the clock check counter after 32 receive high-frequency master clock (AHCLKR) signals have been received. If the current counter value is greater than RMAX after counting 32 AHCLKR signals, RCKFAIL in RSTAT is set. The comparison is performed using unsigned arithmetic.
15-8	RMIN	0-FFh	Receive clock minimum boundary. This 8-bit unsigned value sets the minimum allowed boundary for the clock check counter after 32 receive high-frequency master clock (AHCLKR) signals have been received. If RCNT is less than RMIN after counting 32 AHCLKR signals, RCKFAIL in RSTAT is set. The comparison is performed using unsigned arithmetic.
7-4	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
3-0	RPS	0-Fh	Receive clock check prescaler value.
		0	McASP system clock divided by 1
		1h	McASP system clock divided by 2
		2h	McASP system clock divided by 4
		3h	McASP system clock divided by 8
		4h	McASP system clock divided by 16
		5h	McASP system clock divided by 32
		6h	McASP system clock divided by 64
		7h	McASP system clock divided by 128
		8h	McASP system clock divided by 256
		9h-Fh	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

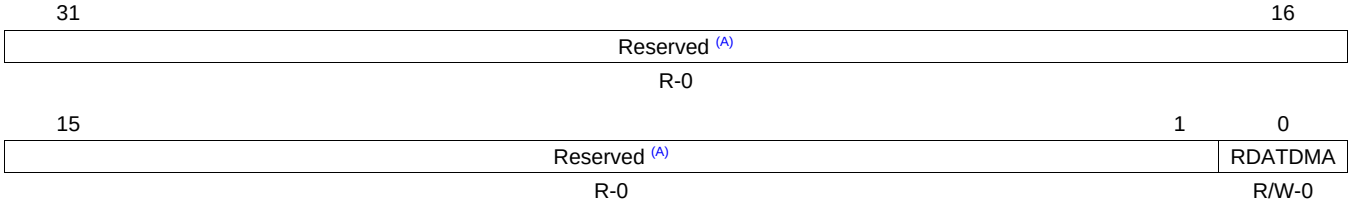
23.1.24 Receiver DMA Event Control Register (REVTCTL)

The receiver DMA event control register (REVTCTL) is shown in Figure 23-57 and described in Table 23-32.

CAUTION

Accessing REVTCTL not implemented on a specific CPU may cause improper device operation.

Figure 23-57. Receiver DMA Event Control Register (REVTCTL)



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

A If writing to this field, always write the default value for future device compatibility.

Table 23-32. Receiver DMA Event Control Register (REVTCTL) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
0	RDATDMA		Receive data DMA request enable bit. If writing to this field, always write the default value of 0.
		0	Receive data DMA request is enabled.
		1	Reserved.

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.25 Transmitter Global Control Register (XGBLCTL)

Alias of the global control register (GBLCTL). Writing to the transmitter global control register (XGBLCTL) affects only the transmit bits of GBLCTL (bits 12-8). Reads from XGBLCTL return the value of GBLCTL. XGBLCTL allows the transmitter to be reset independently from the receiver. The XGBLCTL is shown in Figure 23-58 and described in Table 23-33. See Section 23.1.9 for a detailed description of GBLCTL.

Figure 23-58. Transmitter Global Control Register (XGBLCTL)

31	16
Reserved (A)	
R-0	

15	13	12	11	10	9	8
Reserved (A)		XFRST	XSMRST	XSRCLR	XHCLKRST	XCLKRST
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	5	4	3	2	1	0
Reserved (A)		RFRST	RSMRST	RSRCLR	RHCLKRST	RCLKRST
R-0		R-0	R-0	R-0	R-0	R-0

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

A If writing to this field, always write the default value for future device compatibility.

Table 23-33. Transmitter Global Control Register (XGBLCTL) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0-FFh	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
12	XFRST		Transmit frame sync generator reset enable bit. A write to this bit affects the XFRST bit of GBLCTL.
		0	Transmit frame sync generator is reset.
		1	Transmit frame sync generator is active.
11	XSMRST		Transmit state machine reset enable bit. A write to this bit affects the XSMRST bit of GBLCTL.
		0	Transmit state machine is held in reset.
		1	Transmit state machine is released from reset.
10	XSRCLR		Transmit serializer clear enable bit. A write to this bit affects the XSRCLR bit of GBLCTL.
		0	Transmit serializers are cleared.
		1	Transmit serializers are active.
9	XHCLKRST		Transmit high-frequency clock divider reset enable bit. A write to this bit affects the XHCLKRST bit of GBLCTL.
		0	Transmit high-frequency clock divider is held in reset.
		1	Transmit high-frequency clock divider is running.
8	XCLKRST		Transmit clock divider reset enable bit. A write to this bit affects the XCLKRST bit of GBLCTL.
		0	Transmit clock divider is held in reset.
		1	Transmit clock divider is running.
7-5	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
4	RFRST	x	Receive frame sync generator reset enable bit. A read of this bit returns the RFRST bit value of GBLCTL. Writes have no effect.
3	RSMRST	x	Receive state machine reset enable bit. A read of this bit returns the RSMRST bit value of GBLCTL. Writes have no effect.
2	RSRCLR	x	Receive serializer clear enable bit. A read of this bit returns the RSRSCLR bit value of GBLCTL. Writes have no effect.
1	RHCLKRST	x	Receive high-frequency clock divider reset enable bit. A read of this bit returns the RHCLKRST bit value of GBLCTL. Writes have no effect.
0	RCLKRST	x	Receive clock divider reset enable bit. A read of this bit returns the RCLKRST bit value of GBLCTL. Writes have no effect.

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.26 Transmit Format Unit Bit Mask Register (XMASK)

The transmit format unit bit mask register (XMASK) determines which bits of the transmitted data are masked off and padded with a known value before being shifted out the McASP. The XMASK is shown in Figure 23-59 and described in Table 23-34.

Figure 23-59. Transmit Format Unit Bit Mask Register (XMASK)

31	30	29	28	27	26	25	24
XMASK31	XMASK30	XMASK29	XMASK28	XMASK27	XMASK26	XMASK25	XMASK24
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
XMASK23	XMASK22	XMASK21	XMASK20	XMASK19	XMASK18	XMASK17	XMASK16
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
XMASK15	XMASK14	XMASK13	XMASK12	XMASK11	XMASK10	XMASK9	XMASK8
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
XMASK7	XMASK6	XMASK5	XMASK4	XMASK3	XMASK2	XMASK1	XMASK0
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 23-34. Transmit Format Unit Bit Mask Register (XMASK) Field Descriptions

Bit	Field	Value	Description
31-0	XMASK[31-0]		Transmit data mask enable bit.
		0	Corresponding bit of transmit data (before passing through reverse and rotate units) is masked out and then padded with the selected bit pad value (XPAD and XPBIT bits in XFMT), which is transmitted out the McASP in place of the original bit.
		1	Corresponding bit of transmit data (before passing through reverse and rotate units) is transmitted out the McASP.

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.27 Transmit Bit Stream Format Register (XFMT)

The transmit bit stream format register (XFMT) configures the transmit data format. The XFMT is shown in Figure 23-60 and described in Table 23-35.

Figure 23-60. Transmit Bit Stream Format Register (XFMT)

31																		18	17	16
Reserved [A]																			XDATDLY	
R-0																			R/W-0	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XRVRS	XPAD		XPBIT					XSSZ				XBUSEL	XROT		
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

A If writing to this field, always write the default value for future device compatibility.

Table 23-35. Transmit Bit Stream Format Register (XFMT) Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
17-16	XDATDLY	0-3h	Transmit sync bit delay.
		0	0-bit delay. The first transmit data bit, AXR[n], occurs in same ACLKX cycle as the transmit frame sync (AFSX).
		1h	1-bit delay. The first transmit data bit, AXR[n], occurs one ACLKX cycle after the transmit frame sync (AFSX).
		2h	2-bit delay. The first transmit data bit, AXR[n], occurs two ACLKX cycles after the transmit frame sync (AFSX).
		3h	Reserved.
15	XRVRS		Transmit serial bitstream order.
		0	Bitstream is LSB first. No bit reversal is performed in transmit format bit reverse unit.
		1	Bitstream is MSB first. Bit reversal is performed in transmit format bit reverse unit.
14-13	XPAD	0-3h	Pad value for extra bits in slot not belonging to word defined by XMASK. This field only applies to bits when XMASK[n] = 0.
		0	Pad extra bits with 0.
		1h	Pad extra bits with 1.
		2h	Pad extra bits with one of the bits from the word as specified by XPBIT bits.
		3h	Reserved
12-8	XPBIT	0-1Fh	XPBIT value determines which bit (as written by the CPU or DMA to XBUF[n]) is used to pad the extra bits before shifting. This field only applies when XPAD = 2h.
		0	Pad with bit 0 value.
		1-1Fh	Pad with bit 1 to bit 31 value.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 23-35. Transmit Bit Stream Format Register (XFMT) Field Descriptions (continued)

Bit	Field	Value	Description
7-4	XSSZ	0-Fh	Transmit slot size.
		0-2h	Reserved
		3h	Slot size is 8 bits.
		4h	Reserved
		5h	Slot size is 12 bits.
		6h	Reserved.
		7h	Slot size is 16 bits.
		8h	Reserved.
		9h	Slot size is 20 bits.
		Ah	Reserved.
		Bh	Slot size is 24 bits.
		Ch	Reserved.
		Dh	Slot size is 28 bits.
		Eh	Reserved.
		Fh	Slot size is 32 bits.
3	XBUSEL		Selects whether writes to serializer buffer XRBUF[n] by way of XBUF*n* by the CPU/EDMA occur through the peripheral configuration port or the DMA port.
		0	Writes to XRBUF[n] originate from the DMA port. Writes to XRBUF[n] from the peripheral configuration port are ignored with no effect to the McASP.
		1	Writes to XRBUF[n] originate from the peripheral configuration port. Writes to XRBUF[n] from the DMA port are ignored with no effect to the McASP.
2-0	XROT	0-7h	Right-rotation value for transmit rotate right format unit.
		0	Rotate right by 0 (no rotation).
		1h	Rotate right by 4 bit positions.
		2h	Rotate right by 8 bit positions.
		3h	Rotate right by 12 bit positions.
		4h	Rotate right by 16 bit positions.
		5h	Rotate right by 20 bit positions.
		6h	Rotate right by 24 bit positions.
		7h	Rotate right by 28 bit positions.

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.28 Transmit Frame Sync Control Register (AFSXCTL)

The transmit frame sync control register (AFSXCTL) configures the transmit frame sync (AFSX). The AFSXCTL is shown in Figure 23-61 and described in Table 23-36.

Figure 23-61. Transmit Frame Sync Control Register (AFSXCTL)

31	16
Reserved [A]	
R-0	

15	7	6	5	4	3	2	1	0
XMOD		Reserved [A]		FXWID	Reserved [A]		FSXM	FSXP
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

A If writing to this field, always write the default value for future device compatibility.

Table 23-36. Transmit Frame Sync Control Register (AFSXCTL) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15-7	XMOD	0-1FFh	Transmit frame sync mode select bits.
		0	Burst mode
		1h	Reserved
		2h-20h	2-slot TDM (I2S mode) to 32-slot TDM
		21h-17Fh	Reserved
		180h	384-slot DIT mode
		181h-1FFh	Reserved
6-5	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
4	FXWID		Transmit frame sync width select bit indicates the width of the transmit frame sync (AFSX) during its active period.
		0	Single bit
		1	Single word
3-2	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
1	FSXM		Transmit frame sync generation select bit.
		0	Externally-generated transmit frame sync
		1	Internally-generated transmit frame sync
0	FSXP		Transmit frame sync polarity select bit.
		0	A rising edge on transmit frame sync (AFSX) indicates the beginning of a frame.
		1	A falling edge on transmit frame sync (AFSX) indicates the beginning of a frame.

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.29 Transmit Clock Control Register (ACLKXCTL)

The transmit clock control register (ACLKXCTL) configures the transmit bit clock (ACLKX) and the transmit clock generator. The ACLKXCTL is shown in Figure 23-62 and described in Table 23-37.

Figure 23-62. Transmit Clock Control Register (ACLKXCTL)

31	16
Reserved (A)	
R-0	

15	8	7	6	5	4	0
Reserved (A)		CLKXP	ASYNC	CLKXM	CLKXDIV	
R-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

A If writing to this field, always write the default value for future device compatibility.

Table 23-37. Transmit Clock Control Register (ACLKXCTL) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
7	CLKXP		Transmit bitstream clock polarity select bit.
		0	Rising edge. External receiver samples data on the falling edge of the serial clock, so the transmitter must shift data out on the rising edge of the serial clock.
		1	Falling edge. External receiver samples data on the rising edge of the serial clock, so the transmitter must shift data out on the falling edge of the serial clock.
6	ASYNC		Transmit/receive operation asynchronous enable bit.
		0	Synchronous. Transmit clock and frame sync provides the source for both the transmit and receive sections. Note that in this mode, the receive bit clock is an inverted version of the transmit bit clock. See Section 23.0.21.1.5 for more details.
		1	Asynchronous. Separate clock and frame sync used by transmit and receive sections.
5	CLKXM		Transmit bit clock source bit.
		0	External transmit clock source from ACLKX pin.
		1	Internal transmit clock source from output of programmable bit clock divider.
4-0	CLKXDIV	0-1Fh	Transmit bit clock divide ratio bits determine the divide-down ratio from AHCLKX to ACLKX.
		0	Divide-by-1
		1h	Divide-by-2
		2h-1Fh	Divide-by-3 to divide-by-32

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.30 Transmit High-Frequency Clock Control Register (AHCLKXCTL)

The transmit high-frequency clock control register (AHCLKXCTL) configures the transmit high-frequency master clock (AHCLKX) and the transmit clock generator. The AHCLKXCTL is shown in Figure 23-63 and described in Table 23-38.

Figure 23-63. Transmit High-Frequency Clock Control Register (AHCLKXCTL)

31					16
Reserved [A]					
R-0					

15	14	13	12	11	0
HCLKXM	HCLKXP	Reserved [A]		HCLKXDIV	
R/W-1	R/W-0	R/W-0		R/W-0	

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

A If writing to this field, always write the default value for future device compatibility.

Table 23-38. Transmit High-Frequency Clock Control Register (AHCLKXCTL) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15	HCLKXM		Transmit high-frequency clock source bit.
		0	External transmit high-frequency clock source from AHCLKX pin.
		1	Internal transmit high-frequency clock source from output of programmable high clock divider.
14	HCLKXP		Transmit bitstream high-frequency clock polarity select bit.
		0	Not inverted. AHCLKX is not inverted before programmable bit clock divider. In the special case where the transmit bit clock (ACLKX) is internally generated and the programmable bit clock divider is set to divide-by-1 (CLKXDIV = 0 in ACLKXCTL), AHCLKX is directly passed through to the ACLKX pin.
		1	Inverted. AHCLKX is inverted before programmable bit clock divider. In the special case where the transmit bit clock (ACLKX) is internally generated and the programmable bit clock divider is set to divide-by-1 (CLKXDIV = 0 in ACLKXCTL), AHCLKX is directly passed through to the ACLKX pin.
13-12	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
11-0	HCLKXDIV	0-FFFh	Transmit high-frequency clock divide ratio bits determine the divide-down ratio from AUXCLK to AHCLKX.
		0	Divide-by-1
		1h	Divide-by-2
		2h-FFFh	Divide-by-3 to divide-by-4096

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.31 Transmit TDM Time Slot Register (XTDM)

The transmit TDM time slot register (XTDM) specifies in which TDM time slot the transmitter is active. TDM time slot counter range is extended to 384 slots (to support SPDIF blocks of 384 subframes). XTDM operates modulo 32, that is, XTDMS specifies the TDM activity for time slots 0, 32, 64, 96, 128, etc. The XTDM is shown in Figure 23-64 and described in Table 23-39.

Figure 23-64. Transmit TDM Time Slot Register (XTDM)

31	30	29	28	27	26	25	24
XTDMS31	XTDMS30	XTDMS29	XTDMS28	XTDMS27	XTDMS26	XTDMS25	XTDMS24
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
XTDMS23	XTDMS22	XTDMS21	XTDMS20	XTDMS19	XTDMS18	XTDMS17	XTDMS16
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
XTDMS15	XTDMS14	XTDMS13	XTDMS12	XTDMS11	XTDMS10	XTDMS9	XTDMS8
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
XTDMS7	XTDMS6	XTDMS5	XTDMS4	XTDMS3	XTDMS2	XTDMS1	XTDMS0
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 23-39. Transmit TDM Time Slot Register (XTDM) Field Descriptions

Bit	Field	Value	Description
31-0	XTDMS[31-0]		Transmitter mode during TDM time slot *n*.
		0	Transmit TDM time slot *n* is inactive. The transmit serializer does not shift out data during this slot.
		1	Transmit TDM time slot *n* is active. The transmit serializer shifts out data during this slot according to the serializer control register (SRCTL).

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.32 Transmitter Interrupt Control Register (XINTCTL)

The transmitter interrupt control register (XINTCTL) controls generation of the McASP transmit interrupt (XINT). When the register bit(s) is set to 1, the occurrence of the enabled McASP condition(s) generates XINT. The XINTCTL is shown in Figure 23-65 and described in Table 23-40. See Section 23.1.33 for a description of the interrupt conditions.

Figure 23-65. Transmitter Interrupt Control Register (XINTCTL)

31							8
Reserved (A)							
R-0							
7	6	5	4	3	2	1	0
XSTAFRM	Reserved (A)	XDATA	XLAST	XDMAERR	XCKFAIL	XSYNCERR	XUNDRN
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; R = Read only; -*n* = value after reset

A If writing to this field, always write the default value for future device compatibility.

Table 23-40. Transmitter Interrupt Control Register (XINTCTL) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
7	XSTAFRM		Transmit start of frame interrupt enable bit.
		0	Interrupt is disabled. A transmit start of frame interrupt does not generate a McASP transmit interrupt (XINT).
		1	Interrupt is enabled. A transmit start of frame interrupt generates a McASP transmit interrupt (XINT).
6	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
5	XDATA		Transmit data ready interrupt enable bit.
		0	Interrupt is disabled. A transmit data ready interrupt does not generate a McASP transmit interrupt (XINT).
		1	Interrupt is enabled. A transmit data ready interrupt generates a McASP transmit interrupt (XINT).
4	XLAST		Transmit last slot interrupt enable bit.
		0	Interrupt is disabled. A transmit last slot interrupt does not generate a McASP transmit interrupt (XINT).
		1	Interrupt is enabled. A transmit last slot interrupt generates a McASP transmit interrupt (XINT).
3	XDMAERR		Transmit DMA error interrupt enable bit.
		0	Interrupt is disabled. A transmit DMA error interrupt does not generate a McASP transmit interrupt (XINT).
		1	Interrupt is enabled. A transmit DMA error interrupt generates a McASP transmit interrupt (XINT).
2	XCKFAIL		Transmit clock failure interrupt enable bit.
		0	Interrupt is disabled. A transmit clock failure interrupt does not generate a McASP transmit interrupt (XINT).
		1	Interrupt is enabled. A transmit clock failure interrupt generates a McASP transmit interrupt (XINT).
1	XSYNCERR		Unexpected transmit frame sync interrupt enable bit.
		0	Interrupt is disabled. An unexpected transmit frame sync interrupt does not generate a McASP transmit interrupt (XINT).
		1	Interrupt is enabled. An unexpected transmit frame sync interrupt generates a McASP transmit interrupt (XINT).
0	XUNDRN		Transmitter underrun interrupt enable bit.
		0	Interrupt is disabled. A transmitter underrun interrupt does not generate a McASP transmit interrupt (XINT).
		1	Interrupt is enabled. A transmitter underrun interrupt generates a McASP transmit interrupt (XINT).

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.33 Transmitter Status Register (XSTAT)

The transmitter status register (XSTAT) provides the transmitter status and transmit TDM time slot number. If the McASP logic attempts to set an interrupt flag in the same cycle that the CPU writes to the flag to clear it, the McASP logic has priority and the flag remains set. This also causes a new interrupt request to be generated. The XSTAT is shown in Figure 23-66 and described in Table 23-41.

Figure 23-66. Transmitter Status Register (XSTAT)

31							9	8
Reserved (A)								XERR
R-0								R/W-0
7	6	5	4	3	2	1	0	
XDMAERR	XSTAFRM	XDATA	XLAST	XTDMSLOT	XCKFAIL	XSYNCERR	XUNDRN	
R/W-0	R/W-0	R/W-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

A If writing to this field, always write the default value for future device compatibility.

Table 23-41. Transmitter Status Register (XSTAT) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
8	XERR		XERR bit always returns a logic-OR of: XUNDRN | XSYNCERR | XCKFAIL | XDMAERR Allows a single bit to be checked to determine if a transmitter error interrupt has occurred.
		0	No errors have occurred.
		1	An error has occurred.
7	XDMAERR		Transmit DMA error flag. XDMAERR is set when the CPU or DMA writes more serializers through the DMA port in a given time slot than were programmed as transmitters. Causes a transmit interrupt (XINT), if this bit is set and XDMAERR in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect.
		0	Transmit DMA error did not occur.
		1	Transmit DMA error did occur.
6	XSTAFRM		Transmit start of frame flag. Causes a transmit interrupt (XINT), if this bit is set and XSTAFRM in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect.
		0	No new transmit frame sync (AFSX) is detected.
		1	A new transmit frame sync (AFSX) is detected.
5	XDATA		Transmit data ready flag. Causes a transmit interrupt (XINT), if this bit is set and XDATA in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect.
		0	XBUF is written and is full.
		1	Data is copied from XBUF to XRSR. XBUF is empty and ready to be written. XDATA is also set when the transmit serializers are taken out of reset. When XDATA is set, it always causes a DMA event (AXEVT).
4	XLAST		Transmit last slot flag. XLAST is set along with XDATA, if the current slot is the last slot in a frame. Causes a transmit interrupt (XINT), if this bit is set and XLAST in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect.
		0	Current slot is not the last slot in a frame.
		1	Current slot is the last slot in a frame. XDATA is also set.
3	XTDMSLOT		Returns the LSB of XSLOT. Allows a single read of XSTAT to determine whether the current TDM time slot is even or odd.
		0	Current TDM time slot is odd.
		1	Current TDM time slot is even.
2	XCKFAIL		Transmit clock failure flag. XCKFAIL is set when the transmit clock failure detection circuit reports an error (see Section 23.0.21.6.6). Causes a transmit interrupt (XINT), if this bit is set and XCKFAIL in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect.
		0	Transmit clock failure did not occur.
		1	Transmit clock failure did occur.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 23-41. Transmitter Status Register (XSTAT) Field Descriptions (continued)

Bit	Field	Value	Description
1	XSYNCERR		Unexpected transmit frame sync flag. XSYNCERR is set when a new transmit frame sync (AFSX) occurs before it is expected. Causes a transmit interrupt (XINT), if this bit is set and XSYNCERR in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect.
		0	Unexpected transmit frame sync did not occur.
		1	Unexpected transmit frame sync did occur.
0	XUNDRN		Transmitter underrun flag. XUNDRN is set when the transmit serializer is instructed to transfer data from XBUF to XRSR, but XBUF has not yet been serviced with new data since the last transfer. Causes a transmit interrupt (XINT), if this bit is set and XUNDRN in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect.
		0	Transmitter underrun did not occur.
		1	Transmitter underrun did occur. See Section 23.0.21.6.2 for details on McASP action upon underrun conditions.

23.1.34 Current Transmit TDM Time Slot Register (XSLOT)

The current transmit TDM time slot register (XSLOT) indicates the current time slot for the transmit data frame. The XSLOT is shown in Figure 23-67 and described in Table 23-42.

Figure 23-67. Current Transmit TDM Time Slot Register (XSLOT)

31	16
Reserved (A)	
R-0	

15	9	8	0
Reserved (A)		XSLOTCNT	
R-0		R-17Fh	

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

A If writing to this field, always write the default value for future device compatibility.

Table 23-42. Current Transmit TDM Time Slot Register (XSLOT) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
8-0	XSLOTCNT	0-17Fh	Current transmit time slot count. Legal values: 0 to 383 (17Fh). During reset, this counter value is 383 so the next count value, which is used to encode the first DIT group of data, will be 0 and encodes the B preamble. TDM function is not supported for > 32 time slots. However, TDM time slot counter may count to 383 when used to transmit a DIT block.

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.35 Transmit Clock Check Control Register (XCLKCHK)

The transmit clock check control register (XCLKCHK) configures the transmit clock failure detection circuit. The XCLKCHK is shown in Figure 23-68 and described in Table 23-43.

Figure 23-68. Transmit Clock Check Control Register (XCLKCHK)

31	24	23	16
XCNT		XMAX	
R-0		R/W-0	

15	8	7	4	3	0
XMIN		Reserved (A)		XPS	
R/W-0		R-0		R/W-0	

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

A If writing to this field, always write the default value for future device compatibility.

Table 23-43. Transmit Clock Check Control Register (XCLKCHK) Field Descriptions

Bit	Field	Value	Description
31-24	XCNT	0	Transmit clock count value (from previous measurement). The clock circuit continually counts the number of CPU system clocks for every 32 transmit high-frequency master clock (AHCLKX) signals, and stores the count in XCNT until the next measurement is taken.
23-16	XMAX	0-FFh	Transmit clock maximum boundary. This 8-bit unsigned value sets the maximum allowed boundary for the clock check counter after 32 transmit high-frequency master clock (AHCLKX) signals have been received. If the current counter value is greater than XMAX after counting 32 AHCLKX signals, XCKFAIL in XSTAT is set. The comparison is performed using unsigned arithmetic.
15-8	XMIN	0-FFh	Transmit clock minimum boundary. This 8-bit unsigned value sets the minimum allowed boundary for the clock check counter after 32 transmit high-frequency master clock (AHCLKX) signals have been received. If XCNT is less than XMIN after counting 32 AHCLKX signals, XCKFAIL in XSTAT is set. The comparison is performed using unsigned arithmetic.
7-4	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
3-0	XPS	0-Fh	Transmit clock check prescaler value.
		0	McASP system clock divided by 1
		1h	McASP system clock divided by 2
		2h	McASP system clock divided by 4
		3h	McASP system clock divided by 8
		4h	McASP system clock divided by 16
		5h	McASP system clock divided by 32
		6h	McASP system clock divided by 64
		7h	McASP system clock divided by 128
		8h	McASP system clock divided by 256
		9h-Fh	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

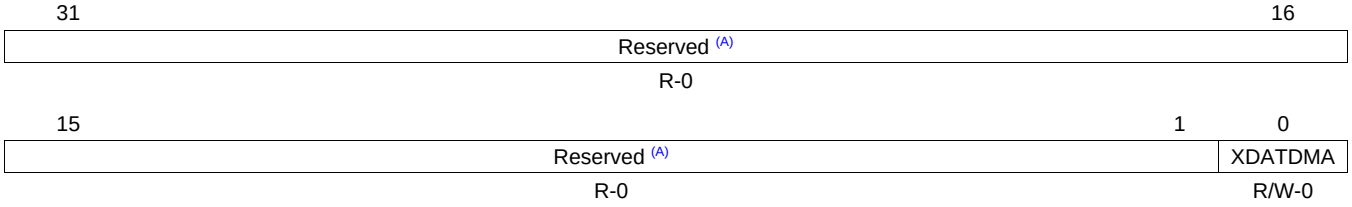
23.1.36 Transmitter DMA Event Control Register (XEVTCTL)

The transmitter DMA event control register (XEVTCTL) is shown in Figure 23-69 and described in Table 23-44.

CAUTION

Accessing XEVTCTL not implemented on a specific CPU may cause improper device operation.

Figure 23-69. Transmitter DMA Event Control Register (XEVTCTL)



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

A If writing to this field, always write the default value for future device compatibility.

Table 23-44. Transmitter DMA Event Control Register (XEVTCTL) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
0	XDATDMA		Transmit data DMA request enable bit. If writing to this field, always write the default value of 0.
		0	Transmit data DMA request is enabled.
		1	Reserved.

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.37 Serializer Control Registers (SRCTL*n*)

Each serializer on the McASP has a serializer control register (SRCTL). There are up to 16 serializers per McASP. The SRCTL is shown in Figure 23-70 and described in Table 23-45.

CAUTION

Accessing SRCTL*n* not implemented on a specific CPU may cause improper device operation.

Figure 23-70. Serializer Control Registers (SRCTL*n*)

31	16
Reserved (A)	
R-0	

15	6	5	4	3	2	1	0
Reserved (A)		RRDY	XRDY	DISMOD		SRMOD	
R-0		R-0	R-0	R/W-0		R/W-0	

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

A If writing to this field, always write the default value for future device compatibility.

Table 23-45. Serializer Control Registers (SRCTL*n*) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
5	RRDY		Receive buffer ready bit. RRDY indicates the current receive buffer state. Always reads 0 when programmed as a transmitter or as inactive. If SRMOD bit is set to receive (2h), RRDY switches from 0 to 1 whenever data is transferred from XRSR to RBUF.
		0	Receive buffer (RBUF) is empty.
		1	Receive buffer (RBUF) contains data and needs to be read before the start of the next time slot or a receiver overrun occurs.
4	XRDY		Transmit buffer ready bit. XRDY indicates the current transmit buffer state. Always reads 0 when programmed as a receiver or as inactive. If SRMOD bit is set to transmit (1h), XRDY switches from 0 to 1 when XSRCLR in GBLCTL is switched from 0 to 1 to indicate an empty transmitter. XRDY remains set until XSRCLR is forced to 0, data is written to the corresponding transmit buffer, or SRMOD bit is changed to receive (2h) or inactive (0).
		0	Transmit buffer (XBUF) contains data.
		1	Transmit buffer (XBUF) is empty and needs to be written before the start of the next time slot or a transmit underrun occurs.
3-2	DISMOD	0-3h	Serializer pin drive mode bit. Drive on pin when in inactive TDM slot of transmit mode or when serializer is inactive. This field only applies if the pin is configured as a McASP pin (PFUNC = 0).
		0	Drive on pin is 3-state.
		1h	Reserved
		2h	Drive on pin is logic low.
		3h	Drive on pin is logic high.
1-0	SRMOD	0-3h	Serializer mode bit.
		0	Serializer is inactive.
		1h	Serializer is transmitter.
		2h	Serializer is receiver.
		3h	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.38 DIT Left Channel Status Registers (DITCSRA0-DITCSRA5)

The DIT left channel status registers (DITCSRA) provide the status of each left channel (even TDM time slot). Each of the six 32-bit registers (Figure 23-71) can store 192 bits of channel status data for a complete block of transmission. The DIT reuses the same data for the next block. It is your responsibility to update the register file in time, if a different set of data need to be sent.

Figure 23-71. DIT Left Channel Status Registers (DITCSRA0-DITCSRA5)

31	0
DITCSRA*n*	
R/W-0	

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

23.1.39 DIT Right Channel Status Registers (DITCSRB0-DITCSRB5)

The DIT right channel status registers (DITCSRB) provide the status of each right channel (odd TDM time slot). Each of the six 32-bit registers (Figure 23-72) can store 192 bits of channel status data for a complete block of transmission. The DIT reuses the same data for the next block. It is your responsibility to update the register file in time, if a different set of data need to be sent.

Figure 23-72. DIT Right Channel Status Registers (DITCSRB0-DITCSRB5)

31	0
DITCSRB*n*	
R/W-0	

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

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.40 DIT Left Channel User Data Registers (DITUDRA0-DITUDRA5)

The DIT left channel user data registers (DITUDRA) provides the user data of each left channel (even TDM time slot). Each of the six 32-bit registers (Figure 23-73) can store 192 bits of user data for a complete block of transmission. The DIT reuses the same data for the next block. It is your responsibility to update the register in time, if a different set of data need to be sent.

Figure 23-73. DIT Left Channel User Data Registers (DITUDRA0-DITUDRA5)

31	0
DITUDRA*n*	
R/W-0	

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

23.1.41 DIT Right Channel User Data Registers (DITUDRB0-DITUDRB5)

The DIT right channel user data registers (DITUDRB) provides the user data of each right channel (odd TDM time slot). Each of the six 32-bit registers (Figure 23-74) can store 192 bits of user data for a complete block of transmission. The DIT reuses the same data for the next block. It is your responsibility to update the register in time, if a different set of data need to be sent.

Figure 23-74. DIT Right Channel User Data Registers (DITUDRB0-DITUDRB5)

31	0
DITUDRB*n*	
R/W-0	

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

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.42 Transmit Buffer Registers (XBUF*n*)

The transmit buffers for the serializers (XBUF) hold data from the transmit format unit. For transmit operations, the XBUF (Figure 23-75) is an alias of the XRBUF in the serializer. The XBUF can be accessed through the peripheral configuration port (Table 23-7) or through the DMA port (Table 23-8).

CAUTION

Accessing XBUF registers not implemented on a specific CPU may cause improper device operation.

Figure 23-75. Transmit Buffer Registers (XBUF*n*)

31	0
XBUF*n*	
R/W-0	

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

23.1.43 Receive Buffer Registers (RBUF*n*)

The receive buffers for the serializers (RBUF) hold data from the serializer before the data goes to the receive format unit. For receive operations, the RBUF (Figure 23-76) is an alias of the XRBUF in the serializer. The RBUF can be accessed through the peripheral configuration port (Table 23-7) or through the DMA port (Table 23-8).

CAUTION

Accessing RBUF registers not implemented on a specific CPU may cause improper device operation.

Figure 23-76. Receive Buffer Registers (RBUF*n*)

31	0
RBUF*n*	
R/W-0	

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

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.44 AFIFO Revision Identification Register (AFIFOREV)

The Audio FIFO (AFIFO) revision identification register (AFIFOREV) contains revision data for the Audio FIFO (AFIFO). The AFIFOREV is shown in Figure 23-77 and described in Table 23-46.

Figure 23-77. AFIFO Revision Identification Register (AFIFOREV)

31	0
REV	
R-4431 1100h	

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

Table 23-46. AFIFO Revision Identification Register (AFIFOREV) Field Descriptions

Bit	Field	Value	Description
31-0	REV	4431 1100h	Identifies revision of Audio FIFO.

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.45 Write FIFO Control Register (WFIFOCTL)

The Write FIFO control register (WFIFOCTL) is shown in Figure 23-78 and described in Table 23-47.

NOTE: The WNUMEVT and WNUMDMA values must be set prior to enabling the Write FIFO.

If the Write FIFO is to be enabled, it must be enabled prior to taking the McASP out of reset.

Figure 23-78. Write FIFO Control Register (WFIFOCTL)

31	17	16
Reserved		WENA
R-0		R/W-0

15	8	7	0
WNUMEVT		WNUMDMA	
R/W-10h		R/W-4h	

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

Table 23-47. Write FIFO Control Register (WFIFOCTL) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16	WENA		Write FIFO enable bit.
		0	Write FIFO is disabled. The WLVL field in the Write FIFO status register (WFIFOSTS) is reset to 0 and pointers are initialized, that is, the Write FIFO is "flushed."
		1	Write FIFO is enabled. If Write FIFO is to be enabled, it must be enabled prior to taking McASP out of reset.
15-8	WNUMEVT	0-FFh	Write word count per DMA event (32-bit). When the Write FIFO has space for at least WNUMEVT words of data, then an AXEVT (transmit DMA event) is generated to the host/DMA controller. This value should be set to a non-zero integer multiple of the number of serializers enabled as transmitters. This value must be set prior to enabling the Write FIFO.
		0	0 words
		1h	1 word
		2h	2 words
		3h-40h	3 to 64 words
		41h-FFh	Reserved
7-0	WNUMDMA	0-FFh	Write word count per transfer (32-bit words). Upon a transmit DMA event from the McASP, WNUMDMA words are transferred from the Write FIFO to the McASP. This value must equal the number of McASP serializers (not the number of channels) used as transmitters. This value must be set prior to enabling the Write FIFO.
		0	0 words
		1h	1 word
		2h	2 words
		3h-10h	3-16 words
		11h-FFh	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.46 Write FIFO Status Register (WFIFOSTS)

The Write FIFO status register (WFIFOSTS) is shown in Figure 23-79 and described in Table 23-48.

Figure 23-79. Write FIFO Status Register (WFIFOSTS)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		WLVL	
R-0		R-0	

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

Table 23-48. Write FIFO Status Register (WFIFOSTS) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	WLVL	0-FFh	Write level (read-only). Number of 32-bit words currently in the Write FIFO.
		0	0 words currently in Write FIFO.
		1h	1 word currently in Write FIFO.
		2h	2 words currently in Write FIFO.
		3h-40h	3 to 64 words currently in Write FIFO.
		41h-FFh	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.47 Read FIFO Control Register (RFIFOCTL)

The Read FIFO control register (RFIFOCTL) is shown in Figure 23-80 and described in Table 23-49.

NOTE: The RNUMEVT and RNUMDMA values must be set prior to enabling the Read FIFO.

If the Read FIFO is to be enabled, it must be enabled prior to taking the McASP out of reset.

Figure 23-80. Read FIFO Control Register (RFIFOCTL)

31	17	16
Reserved		RENA
R-0		R/W-0

15	8	7	0
RNUMEVT		RNUMDMA	
R/W-10h		R/W-4h	

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

Table 23-49. Read FIFO Control Register (RFIFOCTL) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16	RENA		Read FIFO enable bit.
		0	Read FIFO is disabled. The RLVL bit in the Read FIFO status register (RFIFOSTS) is reset to 0 and pointers are initialized, that is, the Read FIFO is “flushed.”
		1	Read FIFO is enabled. If Read FIFO is to be enabled, it must be enabled prior to taking McASP out of reset.
15-8	RNUMEVT	0-FFh	Read word count per DMA event (32-bit). When the Read FIFO contains at least RNUMEVT words of data, then an AREVT (receive DMA event) is generated to the host/DMA controller. This value should be set to a non-zero integer multiple of the number of serializers enabled as receivers. This value must be set prior to enabling the Read FIFO.
		0	0 words
		1h	1 word
		2h	2 words
		3h-40h	3 to 64 words
		41h-FFh	Reserved
7-0	RNUMDMA	0-FFh	Read word count per transfer (32-bit words). Upon a receive DMA event from the McASP, the Read FIFO reads RNUMDMA words from the McASP. This value must equal the number of McASP serializers used as receivers. This value must be set prior to enabling the Read FIFO.
		0	0 words
		1	1 word
		2	2 words
		3h-10h	3-16 words
		11h-FFh	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

23.1.48 Read FIFO Status Register (RFIFOSTS)

The Read FIFO status register (RFIFOSTS) is shown in Figure 23-81 and described in Table 23-50.

Figure 23-81. Read FIFO Status Register (RFIFOSTS)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		RLVL	
R-0		R-0	

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

Table 23-50. Read FIFO Status Register (RFIFOSTS) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	RLVL	0-FFh	Read level (read-only). Number of 32-bit words currently in the Read FIFO.
		0	0 words currently in Read FIFO.
		1h	1 word currently in Read FIFO.
		2h	2 words currently in Read FIFO.
		3h-40h	3 to 64 words currently in Read FIFO.
		41h-FFh	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

Multichannel Buffered Serial Port (McBSP)

This chapter describes the multichannel buffered serial port (McBSP). See your device-specific data manual to determine how many McBSPs are available on your device.

Topic **Page**

24.1 Introduction **1055**
24.2 Architecture **1057**
24.3 Registers **1104**

Copyright © 2013–2016, Texas Instruments Incorporated

24.1 Introduction

24.1.1 Purpose of the Peripheral

The primary use for the multichannel buffered serial port (McBSP) is for audio interface purposes. The McBSP is a specialized version of the McBSP peripheral used on other TI DSPs. The primary audio modes that are supported are the AC97 and IIS modes. In addition to the primary audio modes, the McBSP can be programmed to support other serial formats but is not intended to be used as a high-speed interface.

The McBSP consists of a data path and a control path that connect to external devices. Separate pins for transmission and reception communicate data to these external devices. The CPU communicates to the McBSP using 32-bit-wide control registers accessible via the internal peripheral bus.

24.1.2 Features

The McBSP provides the following functions:

- Full-duplex communication
- Double-buffered data registers, which allow a continuous data stream
- Independent framing and clocking for receive and transmit
- Direct interface to industry-standard codecs, analog interface chips (AICs), and other serially connected analog-to-digital (A/D) and digital-to-analog (D/A) devices
- External shift clock or an internal, programmable frequency shift clock for data transfer

In addition, the McBSP has the following capabilities:

- Direct interface to:
  - T1/E1 framers
  - MVIP switching compatible and ST-BUS compliant devices including:
    - MVIP framers
    - H.100 framers
    - SCSA framers
  - IOM-2 compliant devices
  - AC97 compliant devices (the necessary multiphase frame synchronization capability is provided)
  - IIS compliant devices
- Multichannel transmit and receive of up to 128 channels
- A wide selection of data sizes, including 8, 12, 16, 20, 24, and 32 bits
- μ-Law and A-Law companding
- 8-bit data transfers with the option of LSB or MSB first
- Programmable polarity for both frame synchronization and data clocks
- Highly programmable internal clock and frame generation
- Additional McBSP Buffer FIFO (BFIFO):
  - Provides additional data buffering
  - Provides added tolerance to variations in host/DMA controller response times
  - May be used as a DMA event pacer
  - Independent Read FIFO and Write FIFO
  - 256 bytes of RAM for each FIFO (read and write)
  - Option to bypass Write FIFO and/or Read FIFO, independently

Copyright © 2013–2016, Texas Instruments Incorporated

24.1.3 Functional Block Diagram

The McBSP consists of a data path and control path, as shown in Figure 24-1

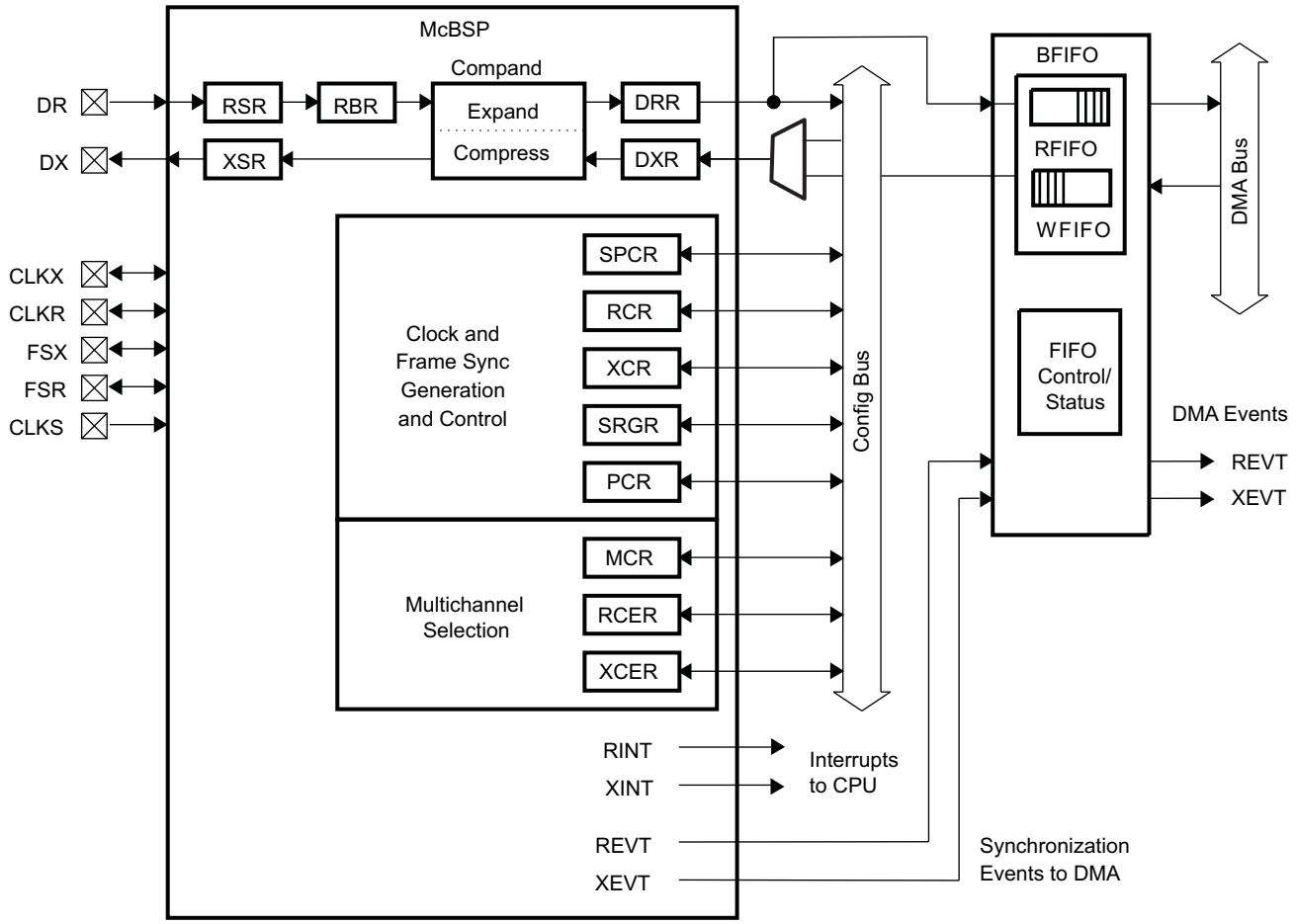
24.1.4 Industry Standard Compliance Statement

The McBSP supports the following industry standard interfaces:

AC97— The AC97 standard specifies a 5-wire digital serial link between an audio codec device and its digital controller.

IIS— IIS is a protocol for transmitting two channels of digital audio data over a single serial connection. The IIS bus is an industry standard three-wire interface for streaming stereo audio between devices, typically between a CPU/DSP and a DAC/ADC.

Figure 24-1. McBSP Block Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

24.2 Architecture

This section describes the architecture of the McBSP.

24.2.1 Clock Control

The McBSP can use an internal or external clock source. Either clock source can be divided-down inside the McBSP to generate the actual interface bit clock frequency. For detailed timing information, see the device-specific data manual. Detailed information about how the interface clock and frame synchronization signals are generated is provided in Section 24.2.5.

24.2.2 Signal Descriptions

The signals used on the audio interface are listed in Table 24-1.

Table 24-1. McBSP Interface Signals

Pin	I/O/Z	Description
CLKR	I/O/Z	Receive clock - supplies or receives a reference clock for the receiver; or supplies a reference clock to the sample rate generator
CLKS	I	Supplies the input clock of the sample rate generator
CLKX	I/O/Z	Transmit clock - supplies or receives a reference clock for the transmitter; or supplies a reference clock to the sample rate generator
DR	I	Received serial data
DX	O/Z	Transmitted serial data
FSR	I/O/Z	Receive frame synchronization - control signal to synchronize the start of received data
FSX	I/O/Z	Transmit frame synchronization - control signal to synchronize the start of transmitted data

24.2.3 Pin Multiplexing

Extensive pin multiplexing is used to accommodate the largest number of peripheral functions in the smallest possible package. Pin multiplexing is controlled using a combination of hardware configuration at device reset and software programmable register settings. Refer to the device-specific data manual to determine how pin multiplexing affects the McBSP.

24.2.4 Endianness Considerations

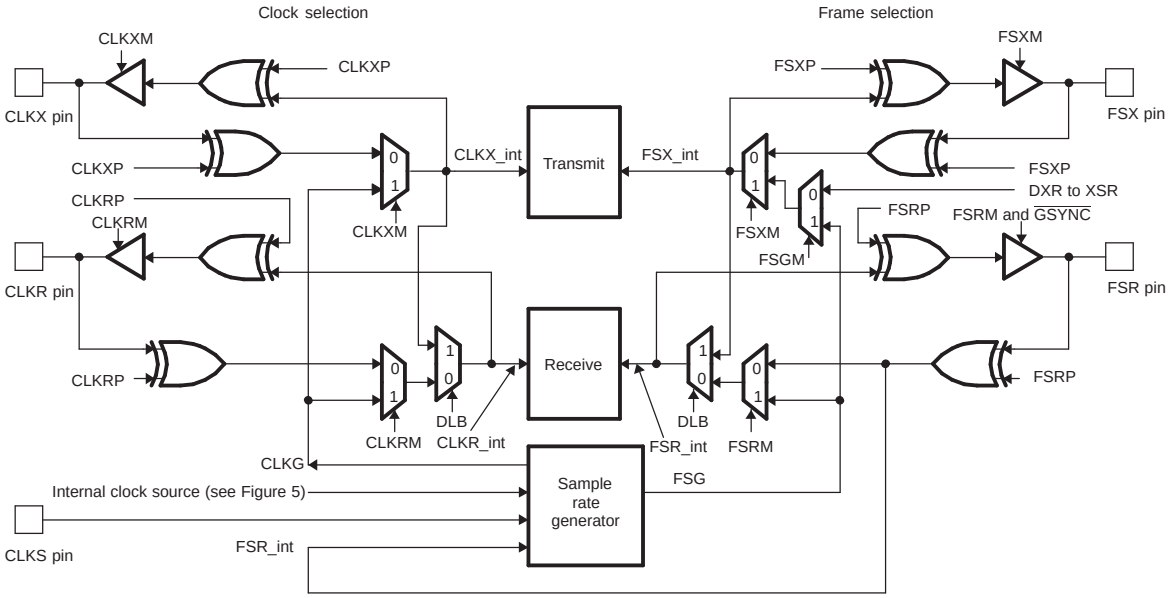
There are no endianness considerations for the McBSP.

Copyright © 2013–2016, Texas Instruments Incorporated

24.2.5 Clock, Frames, and Data

The McBSP has several ways of selecting clocking and framing for both the receiver and transmitter. Clocking and framing can be sent to both portions by the sample rate generator. Each portion can select external clocking and/or framing independently. Figure 24-2 is a block diagram of the clock and frame selection circuitry.

Figure 24-2. Clock and Frame Generation



24.2.5.1 Frame and Clock Operation

Receive and transmit frame sync pulses (FSR/X), and clocks (CLKR/X), can either be generated internally by the sample rate generator (see Section 24.2.5.2) or be driven by an external source. The source of frame sync and clock is selected by programming the mode bits, FS(R/X)M and CLK(R/X)M respectively, in the pin control register (PCR). FSR is also affected by the GSYNC bit in the sample rate generator register (SRGR), see Section 24.2.5.4.2 for details.

When FSR and FSX are inputs (FSXM = FSRM = 0), the McBSP detects them on the internal falling edge of clock, CLKR_int and CLKX_int, respectively (see Figure 24-2). The receive data arriving at the DR pin is also sampled on the falling edge of CLKR_int. These internal clock signals are either derived from an external source via the CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs 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)_int. Similarly, data on DX is output on the rising edge of CLKX_int.

The FSRP, FSXP, CLKRP, and CLKXP bits in PCR configure the polarities of FSR, FSX, CLKR, and CLKX. All frame sync signals (FSR_int and FSX_int) internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to the McBSP) and FSRP = FSXP = 1, the external active (low) frame sync signals are inverted before being sent to the receiver signal (FSR_int) and transmitter signal (FSX_int). Similarly, if internal synchronization is selected (FSR/FSX are outputs and GSYNC = 0), the internal active (high) sync signals are inverted if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin.Figure 24-2 shows this inversion using XOR gates.

Copyright © 2013–2016, Texas Instruments Incorporated

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 CLKX_int (see Figure 24-3). 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, CLKX_int, is inverted before being sent out on the CLKX pin.

Similarly, the receiver can reliably sample data that is clocked (by the transmitter) with a rising-edge clock. 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 CLKR_int (see Figure 24-4). 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 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 before being sent out on the CLKR pin.

In a system where the same clock (internal or external) is used to clock the receiver and transmitter, CLKRP = CLKXP. The receiver uses the opposite edge as the transmitter to ensure valid setup and hold times of data around this edge. Figure 24-4 shows how data clocked by an external serial device using a rising-edge clock can be sampled by the McBSP receiver with the falling edge of the same clock.

Figure 24-3. Transmit Data Clocking

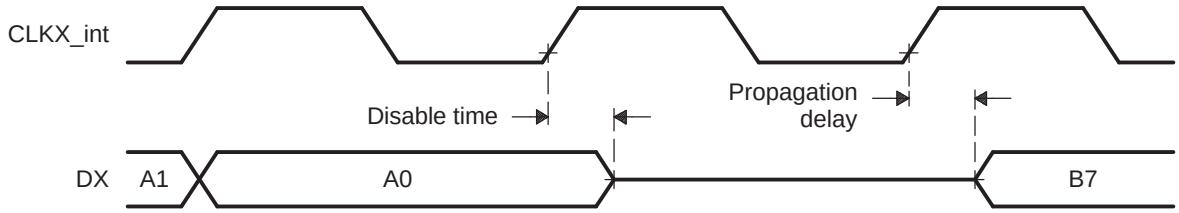
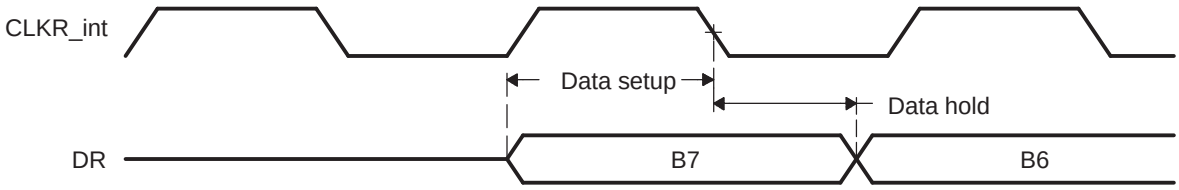


Figure 24-4. Receive Data Clocking



Copyright © 2013–2016, Texas Instruments Incorporated

24.2.5.2 Sample Rate Generator Clocking and Framing

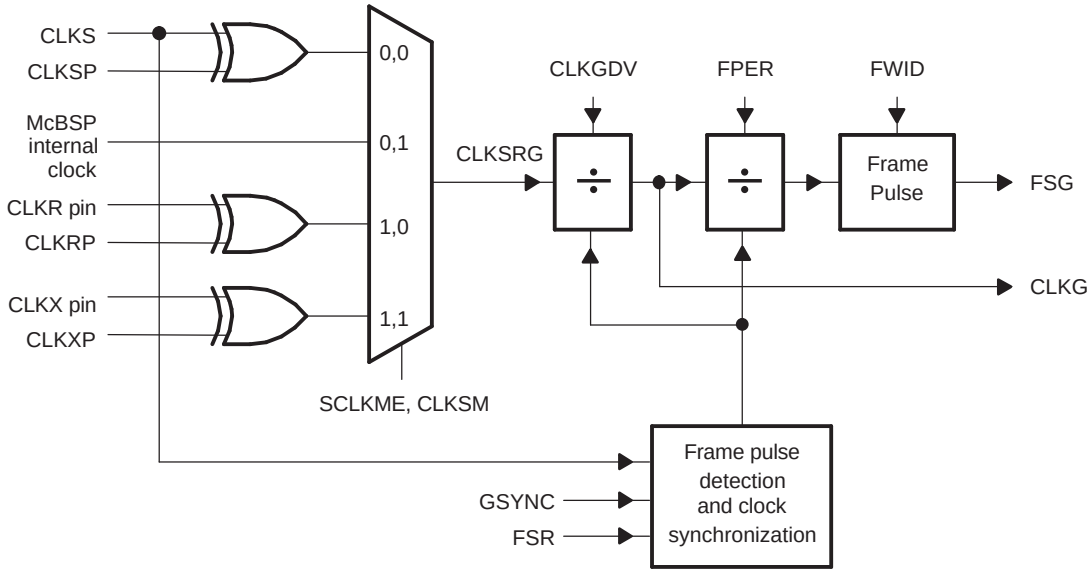
The sample rate generator is composed of a 3-stage clock divider that provides a programmable data clock (CLKG) and framing signal (FSG), as shown in Figure 24-5. CLKG and FSG are McBSP internal signals that can be programmed to drive receive and/or transmit clocking, CLK(R/X), and framing, FS(R/X). The sample rate generator can be programmed to be driven by an internal clock source or an internal clock derived from an external clock source.

The sample rate generator is not used when CLKX, FSX, CLKR, and FSR are driven by an external source. Therefore, the GRST bit in the serial port control register (SPCR) does not need to be enabled (GRST = 1) for this setup. The three stages of the sample rate generator circuit compute:

- Clock divide-down (CLKGDV): The number of input clocks per data bit clock
- Frame period (FPER): The frame period in data bit clocks
- Frame width (FWID): The width of an active frame pulse in data bit clocks

In addition, a frame pulse detection and clock synchronization module allows synchronization of the clock divide-down with an incoming frame pulse. The operation of the sample rate generator during device reset is described in Section 24.2.11.

Figure 24-5. Sample Rate Generator Block Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

24.2.5.3 Data Clock Generation

When the receive/transmit clock mode is set to 1 (CLK(R/X)M = 1 in the pin control register (PCR)), the data clocks (CLK(R/X)) are driven by the internal sample rate generator output clock, CLKG. You can select for the receiver and transmitter from a variety of data bit clocks including:

- The input clock to the sample rate generator, which can be either the internal clock source or a dedicated external clock source via the CLKX, CLKR, or CLKS pins. Internally, the McBSP clock is selected to be either PLL0_SYSCLK2 or PLL1_SYSCLK2 by configuring the ASYNC3_CLKSRC bit in the chip configuration 3 register (CFGCHIP3) of the System Configuration Module. See Section 24.2.5.3.1 for details on the source of the McBSP internal clock.
- The input clock source (internal clock source or external clock CLKX/CLKR/CLKS) to the sample rate generator can be divided-down by a programmable value (CLKGDV bit in the sample rate generator register (SRGR)) to drive CLKG.

Regardless of the source to the sample rate generator, the rising edge of CLKSRG (see Figure 24-5) generates CLKG and FSG.

24.2.5.3.1 Input Clock Source Mode: CLKSM and SCLKME

The sample rate generator input clock signal can be driven from one of four sources selectable with the SCLKME bit in the pin control register (PCR) and the CLKSM bit in the sample rate generator register (SRGR), see Table 24-2.

Table 24-2. Choosing an Input Clock for the Sample Rate Generator With the SCLKME and CLKSM Bits

SCLKME Bit in PCR	CLKSM Bit in SRGR	Input Clock for Sample Rate Generator
0	0	Signal on CLKS pin
0	1	McBSP internal input clock
1	0	Signal on CLKR pin
1	1	Signal on CLKX pin

24.2.5.3.2 Sample Rate Generator Data Bit Clock Rate: CLKGDV

The first divider stage generates the serial data bit clock from the input clock. This divider stage uses a counter that is preloaded by the CLKGDV bit in the sample rate generator register (SRGR) and that contains the divide ratio value. The output of this stage is the data bit clock that is output on the sample rate generator output, CLKG, and that serves as the input for the second and third divider stages.

CLKG has a frequency equal to 1/(CLKGDV + 1) of the sample rate generator input clock. Thus, the sample rate generator input clock frequency is divided by a value between 1 to 256. The CLKGDV value chosen must result in a clock that meets the timing requirements/limitations specified in the device-specific data manual.

When CLKGDV is an odd value or equal to 0, the CLKG duty cycle is 50%. Note that an odd CLKGDV value means an even divide down of the source clock and an even CLKGDV value means an odd divide down of the source clock. When CLKGDV is an even value (2p), the high state duration is p + 1 cycles and the low state duration is p cycles. This is illustrated in Example 24-1, Example 24-2, and Example 24-3.

In the following examples:

S_{IN} = sample generator input clock period

f_{IN} = sample generator input clock frequency

S_G = CLKG period

f_G = CLKG frequency

The following equation is given above: $f_G = f_{IN}/(CLKGDV + 1)$; therefore, $S_G = (CLKGDV + 1) \times S_{IN}$.

Copyright © 2013–2016, Texas Instruments Incorporated

Example 24-1. CLKGDV = 0

CLKGDV = 0

$S_G = (CLKGDV + 1) \times S_{IN} = (0 + 1) \times S_{IN} = S_{IN}$

Pulse width high = $S_{IN} \times (CLKGDV + 1)/2 = S_{IN} \times (0+ 1)/2 = 0.5 \times S_{IN}$

Pulse width low = $S_{IN} \times (CLKGDV + 1)/2 = S_{IN} \times (0+ 1)/2 = 0.5 \times S_{IN}$

Example 24-2. CLKGDV = 1

CLKGDV = 1

$S_G = (CLKGDV + 1) \times S_{IN} = (1 + 1) \times S_{IN} = 2 \times S_{IN}$

Pulse width high = $S_{IN} \times (CLKGDV + 1)/2 = S_{IN} \times (1 +1)/2 = S_{IN}$

Pulse width low = $S_{IN} \times (CLKGDV + 1)/2 = S_{IN} \times (1+ 1)/2 = S_{IN}$

Example 24-3. CLKGDV = 2

CLKGDV = 2

$S_G = (CLKGDV + 1) \times S_{IN} = (2 + 1) \times S_{IN} = 3 \times S_{IN}$

Pulse width high = $S_{IN} \times (CLKGDV/2 + 1) = S_{IN} \times (2/2 + 1) = 2 \times S_{IN}$

Pulse width low = $S_{IN} \times CLKGDV/2 = S_{IN} \times 2/2 = 1 \times S_{IN}$

24.2.5.3.3 Bit Clock Polarity: CLKSP

The external clock (CLKS) is selected to drive the sample rate generator clock divider by selecting CLKSM = 0 in the sample rate generator register (SRGR) and SCLKME = 0 in the pin control register (PCR). In this case, the CLKSP bit in SRGR selects the edge of CLKS on which sample rate generator data bit clock (CLKG) and frame sync signal (FSG) are generated. Since the rising edge of CLKSRG generates CLKG and FSG, the rising edge of CLKS when CLKSP = 0 or the falling edge of CLKS when CLKSP = 1 causes the transition on CLKG and FSG.

24.2.5.3.4 Bit Clock and Frame Synchronization

When the external clock (CLKS) is selected to drive the sample rate generator (CLKSM = 0 in SRGR and SCLKME = 0 in PCR), the GSYNC bit in SRGR can be used to configure the timing of CLKG relative to CLKS. GSYNC = 1 ensures that the McBSP and the external device to which it is communicating are dividing down the CLKS with the same phase relationship. If GSYNC = 0, this feature is disabled and CLKG runs freely and is not resynchronized. If GSYNC = 1, an inactive-to-active transition on FSR triggers a resynchronization of CLKG and the generation of FSG. CLKG always begins at a high state after synchronization. Also, FSR is always detected at the same edge of CLKS that generates CLKG, regardless of the length the FSR pulse. Although an external FSR is provided, FSG can still drive internal receive frame synchronization when GSYNC = 1. When GSYNC = 1, FPER does not matter, because the frame period is determined by the arrival of the external frame sync pulse.

Figure 24-6 and Figure 24-7 show this operation with various polarities of CLKS and FSR. These figures assume that FWID is 0, for a FSG = 1 CLKG wide.

These figures show what happens to CLKG when it is initially in sync and GSYNC = 1, as well as when it is not in sync with the frame synchronization and GSYNC = 1.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 24-6. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 1

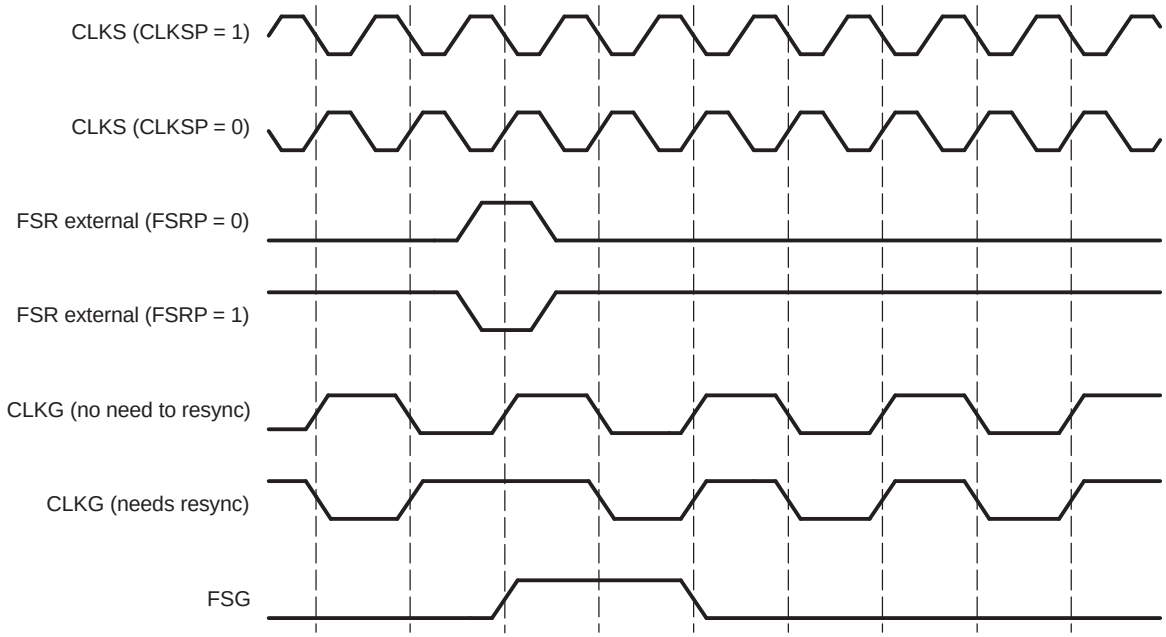
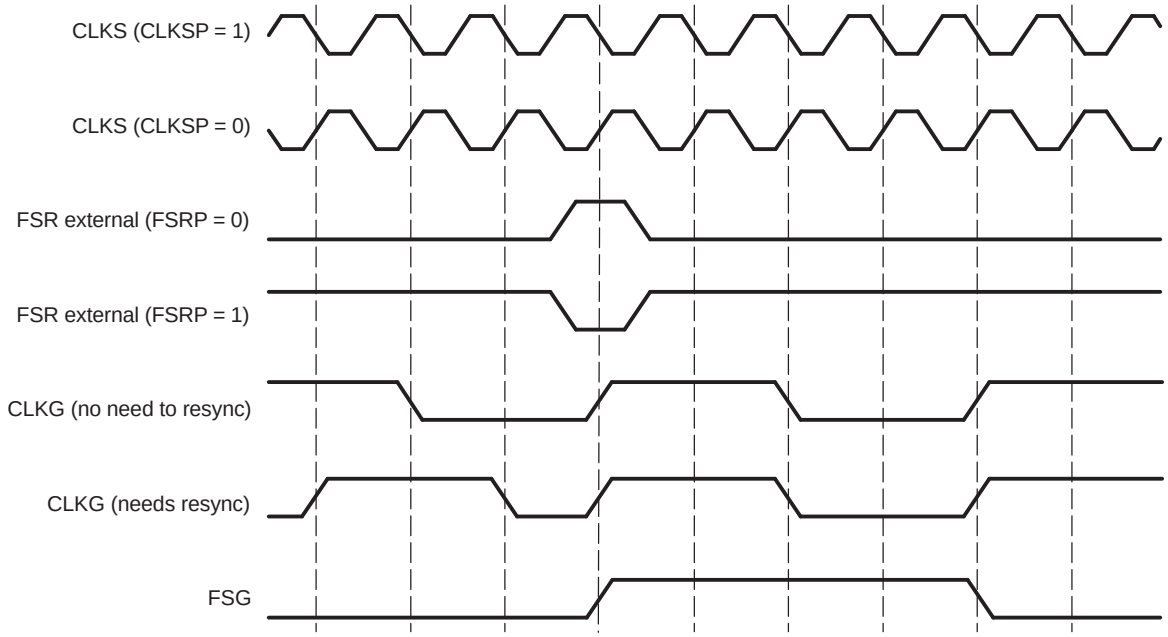


Figure 24-7. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 3



Copyright © 2013–2016, Texas Instruments Incorporated

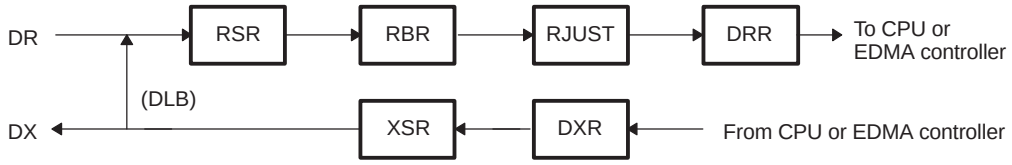
When GSYNC = 1, the transmitter can operate synchronously with the receiver, provided that the following conditions are met:

- FSX is programmed to be driven by the sample rate generator frame sync, FSG (FSGM = 1 in SRGR and FSXM = 1 in PCR).
- The sample-rate generator clock should drive the transmit and receive bit clock (CLK(R/X)M = 1 in SPCR). Therefore, the CLK(R/X) pin should not be driven by any other source.

24.2.5.3.5 Digital Loopback Mode: DLB

The DLB mode cannot be used when the McBSP is in clock stop mode (CLKSTP = 2h or 3h in the serial port control register (SPCR)). Setting DLB = 1 in SPCR enables digital loopback mode. In DLB mode, DR, FSR, and CLKR are internally connected through multiplexers to DX, FSX, and CLKX, respectively, as shown in Figure 24-2 and Figure 30-4. DLB mode allows testing of serial port code without using the external interface of the McBSP. CLKX and FSX must be enabled as outputs (CLKXM = FSXM = 1) in DLB mode.

Figure 24-8. Digital Loopback Mode



24.2.5.3.6 Receive Clock Selection: DLB, CLKRM

Table 24-3 shows how the digital loopback bit (DLB) in the serial port control register (SPCR) and the CLKRM bit in the pin control register (PCR) select the receiver clock. In digital loopback mode (DLB = 1), the transmitter clock drives the receiver. CLKRM determines whether the CLKR pin is an input or an output.

Table 24-3. Receive Clock Selection

DLB Bit in SPCR	CLKRM Bit in PCR	Source of Receive Clock	CLKR Function
0	0	CLKR acts as an input driven by the external clock and inverted as determined by CLKRP before being used.	Input.
0	1	The sample rate generator clock (CLKG) drives CLKR.	Output. CLKG inverted as determined by CLKRP before being driven out on CLKR.
1	0	CLKX_int drives the receive clock CLKR_int as selected and is inverted.	High impedance.
1	1	CLKX_int drives CLKR_int as selected and is inverted.	Output. CLKR (same as CLKX) is inverted as determined by CLKRP before being driven out.

Copyright © 2013–2016, Texas Instruments Incorporated

24.2.5.3.7 Transmit Clock Selection: CLKXM

Table 24-4 shows how the CLKXM bit in the pin control register (PCR) selects the transmit clock and whether the CLKX pin is an input or output.

Table 24-4. Transmit Clock Selection

CLKXM Bit in PCR	Source of Transmit Clock	CLKX Function
0	The external clock drives the CLKX input pin. CLKX is inverted as determined by CLKXP before being used.	Input.
1	The sample rate generator clock (CLKG) drives the transmit clock.	Output. CLKG is inverted as determined by CLKXP before being driven out on CLKX.

24.2.5.3.8 Stopping Clocks

Two methods can be used to stop serial clocks between data transfers:

- The SPI™ CLKSTP mode where clocks are stopped between single-element transfers. This mode is not supported on this device.
- The clocks are inputs to the McBSP (CLKXM or CLKRM = 0 in the pin control register (PCR)) and the McBSP operates in non-SPI mode (the clocks can be stopped between data transfers). If the external device stops the serial clock between data transfers, the McBSP interprets it as a slowed-down serial clock. Ensure that there are no glitches on the CLK(R/X) lines as the McBSP may interpret them as clock-edge transitions. Restarting the serial clock is equivalent to a normal clock transition after a slow CLK(R/X) cycle. Note that just as in normal operations, transmit under flow (XEMPTY) may occur if the DXR is not properly serviced at least three CLKX cycles before the next frame sync. Therefore, if the serial clock is stopped before DXR is properly serviced, the external device needs to restart the clock at least three CLKX cycles before the next frame sync to allow the DXR write to be properly synchronized. See Figure 24-29 for a graphical explanation on when DXR needs to be written to avoid underflow.

24.2.5.4 Frame Sync Generation

Data frame synchronization is independently programmable for the receiver and the transmitter for all data delay values. When the FRST bit in the serial port control register (SPCR) is set to 1 the frame generation logic is activated to generate frame sync signals, provided that FSGM = 1 in the sample rate generator register (SRGR). The frame sync programming options are:

- A frame pulse with a programmable period between sync pulses and a programmable active width specified in SRGR.
- The transmitter can trigger its own frame sync signal that is generated by a DXR-to-XSR copy. This causes a frame sync to occur on every DXR-to-XSR copy. The data delays can be programmed as required. However, maximum packet frequency cannot be achieved in this method for data delays of 1 and 2.
- Both the receiver and transmitter can independently select an external frame synchronization on the FSR and FSX pins, respectively.

24.2.5.4.1 Frame Period (FPER) and Frame Width (FWID)

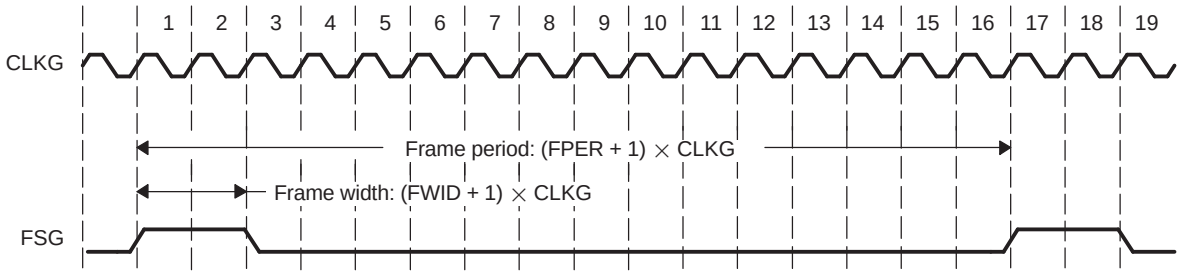
The FPER bits in the sample rate generator register (SRGR) are a 12-bit down-counter that can count down the generated data clocks from 4095 to 0. FPER controls the period of active frame sync pulses. The FWID bits in SRGR are an 8-bit down-counter. FWID controls the active width of the frame sync pulse.

Copyright © 2013–2016, Texas Instruments Incorporated

When the sample rate generator comes out of reset, FSG is in an inactive (low) state. After this, when FRST = 1 in SPCR and FSGM = 1 in SRGR, frame sync signals are generated. The frame width value (FWID + 1) is counted down on every CLKG cycle until it reaches 0 when FSG goes low. Thus, the value of FWID + 1 determines an active frame pulse width ranging from 1 to 256 data bit clocks. At the same time, the frame period value (FPER + 1) is also counting down, and when this value reaches 0, FSG goes high again, indicating a new frame is beginning. Thus, the value of FPER + 1 determines a frame length from 1 to 4096 data bits. When GSYNC = 1 in SRGR, the value of FPER does not matter. Figure 24-9 shows a frame of 16 CLKG periods (FPER = 15 or 0000 1111b).

It is recommended that FWID be programmed to a value less than (R/X)WDLEN1/2.

Figure 24-9. Programmable Frame Period and Width



24.2.5.4.2 Receive Frame Synchronization Selection: DLB and FSRM

Table 24-5 shows how you can select various sources to provide the receive frame synchronization signal. Note that in digital loopback mode (DLB = 1 in the serial port control register (SPCR)), the transmit frame sync signal is used as the receive frame sync signal and that DR is internally connected to DX.

NOTE: FSR_int and FSX_int are shown in Figure 24-2.

Table 24-5. Receive Frame Synchronization Selection

DLB Bit in SPCR	FSRM Bit in PCR	GSYNC Bit in SRGR	Source of Receive Frame Synchronization	FSR Pin Function
0	0	X	External frame sync signal drives the FSR input pin, whose signal is then inverted as determined by FSRP before being used as FSR_int.	Input.
0	1	0	Sample rate generator frame sync signal (FSG) drives FSR_int, FRST = 1.	Output. FSG is inverted as determined by FSRP before being driven out on the FSR pin.
0	1	1	Sample rate generator frame sync signal (FSG) drives FSR_int, FRST = 1.	Input. The external frame sync input on FSR is used to synchronize CLKG and generate FSG.
1	0	0	FSX_int drives FSR_int. FSX is selected as shown in Table 24-6.	High impedance.
1	X	1	FSX_int drives FSR_int and is selected as shown in Table 24-6.	Input. External FSR is not used for frame synchronization but is used to synchronize CLKG and generate FSG since GSYNC = 1.
1	1	0	FSX_int drives FSR_int and is selected as shown in Table 24-6.	Output. Receive (same as transmit) frame synchronization is inverted as determined by FSRP before being driven out.

Copyright © 2013–2016, Texas Instruments Incorporated

24.2.5.4.3 Transmit Frame Synchronization Selection: FSXM and FSGM

Table 24-6 shows how you can select the source of the transmit frame synchronization signal. The three choices are:

- External frame sync input
- The sample rate generator frame sync signal, FSG
- A signal that indicates a DXR-to-XSR copy has been made

NOTE: FSR_int and FSX_int are shown in Figure 24-2.

Table 24-6. Transmit Frame Synchronization Selection

FSXM Bit in PCR	FSGM Bit in SRGR	Source of Transmit Frame Synchronization	FSX Pin Function
0	X	External frame sync input on the FSX pin. This is inverted by FSXP before being used as FSX_int.	Input.
1	1	Sample rate generator frame sync signal (FSG) drives FSX_int. FRST = 1.	Output. FSG is inverted by FSXP before being driven out on FSX.
1	0	A DXR-to-XSR copy activates transmit frame sync signal.	Output. 1-bit-clock-wide signal inverted as determined by FSXP before being driven out on FSX.

24.2.5.4.4 Frame Detection

To facilitate detection of frame synchronization, the receive and transmit CPU interrupts (RINT and XINT) can be programmed to detect frame synchronization by setting the RINTM and XINTM bits in the serial port control register (SPCR) to 10b. The associated portion (receiver/transmitter) of the McBSP must be out of reset.

24.2.5.5 Data and Frames

24.2.5.5.1 Frame Synchronization Phases

Frame synchronization indicates the beginning of a transfer on the McBSP. The data stream following frame synchronization can have up to two phases, phase 1 and phase 2. The number of phases can be selected by the phase bit, (R/X)PHASE, in RCR and XCR. The number of elements per frame and bits per element can be independently selected for each phase via (R/X)FRLEN1/2 and (R/X)WDLEN1/2, respectively. Figure 24-10 shows a frame in which the first phase consists of two elements of 12 bits, each followed by a second phase of three elements of 8 bits each. The entire bit stream in the frame is contiguous; no gaps exist either between elements or phases. Table 24-7 shows the fields in the receive/transmit control registers (RCR/XCR) that control the frame length and element length for each phase for both the receiver and the transmitter. The maximum number of elements per frame is 128 for a single-phase frame and 256 elements in a dual-phase frame. The number of bits per element can be 8, 12, 16, 20, 24, or 32.

NOTE: For a dual-phase frame with internally generated frame synchronization, the maximum number of elements per phase depends on the word length. This is because the frame period, FPER, is only 12-bits wide and, therefore, provides 4096 bits per frame. Hence, the maximum number of 256 elements per dual- phase frame applies only when the WDLEN is 16 bits. However, any combination of element numbers and element size (defined by the FRLEN and WDLEN bits, respectively) is valid as long as their product is less than or equal to 4096 bits. This limitation does not apply for dual-phase with external frame sync.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 24-10. Dual-Phase Frame Example

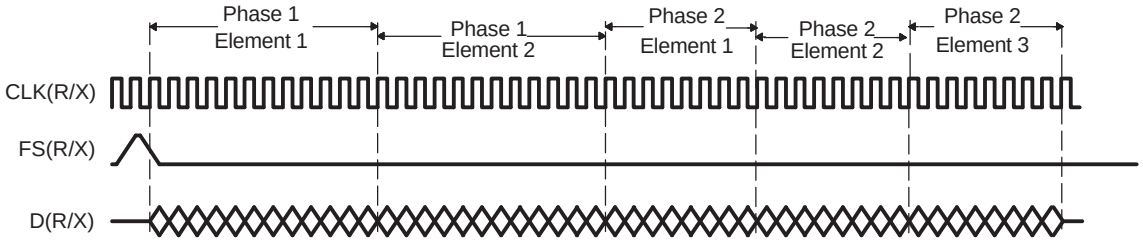


Table 24-7. RCR/XCR Fields Controlling Elements per Frame and Bits per Element

Serial Port	Frame Phase	RCR/XCR Field Control	
		Elements per Frame	Bits per Element
Receive	1	RFRLEN1	RWDLEN1
Receive	2	RFRLEN2	RWDLEN2
Transmit	1	XFRLEN1	XWDLEN1
Transmit	2	XFRLEN2	XWDLEN2

24.2.5.5.2 Frame Length: RFRLEN1/2 and XFRLEN1/2

Frame length specifies the maximum number of serial elements or logical time slots or channels that are available for transfer per frame synchronization signal. In multichannel selection mode, the frame length value is independent of, and perhaps different from, the actual number of channels that the device is programmed to receive or transmit per frame via the MCR, RCERE*n*, and XCERE*n* registers. See Section 24.2.9 for details on multichannel selection mode operation. The 7-bit (R/X)FRLEN1/2 bits in (R/X)CR support up to 128 elements per phase in a frame, as shown in Table 24-8. (R/X)PHASE = 0 selects a single-phase data frame, and (R/X)PHASE = 1 selects a dual-phase frame for the data stream. For a single-phase frame, the value of (R/X)FRLEN2 does not matter. Program the frame length fields with (*w* minus 1), where *w* represents the number of elements per frame. For Figure 24-10, (R/X)FRLEN1 = 1 or 000 0001b and (R/X)FRLEN2 = 2 or 000 0010b.

Table 24-8. Receive/Transmit Frame Length Configuration

(R/X)PHASE	(R/X)FRLEN1	(R/X)FRLEN2	Frame Length
0	0 ≤ n ≤ 127	x	Single-phase frame; (n + 1) elements per frame
1	0 ≤ n ≤ 127	0 ≤ m ≤ 127	Dual-phase frame; (n + 1) plus (m + 1) elements per frame

24.2.5.5.3 Element Length: RWDLEN1/2 and XWDLEN1/2

The (R/X)WDLEN1/2 fields in the receive/transmit control register (RCR and XCR) determine the element length in bits per element for the receiver and the transmitter for each phase of the frame, as indicated in Table 24-7. Table 24-9 shows how the value of these fields selects particular element lengths in bits. For the example in Figure 24-10, (R/X)WDLEN1 = 001b and (R/X)WDLEN2 = 000b. If (R/X)PHASE = 0, indicating a single-phase frame, then (R/X)WDLEN2 is not used by the McBSP and its value does not matter.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 24-9. Receive/Transmit Element Length Configuration

(R/X)WDLEN1/2	Element Length (Bits)
000	8
001	12
010	16
011	20
100	24
101	32
110	Reserved
111	Reserved

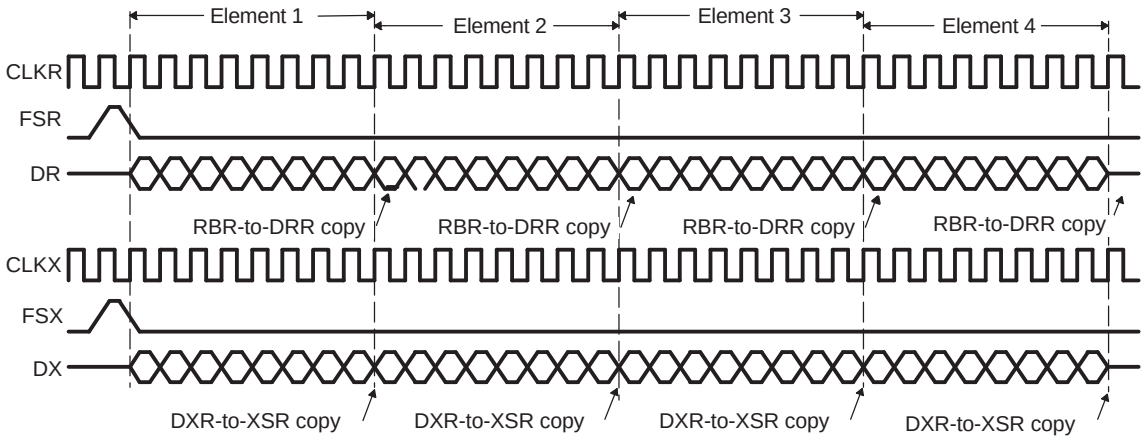
24.2.5.5.4 Data Packing Using Frame Length and Element Length

The frame length and element length can be manipulated to effectively pack data. For example, consider a situation in which four 8-bit elements are transferred in a single-phase frame, as shown in Figure 24-11. In this case:

- (R/X)PHASE = 0, indicating a single-phase frame
- (R/X)FRLEN1 = 000 0011b, indicating a 4-element frame
- (R/X)WDLEN1 = 000b, indicating 8-bit elements

In this example, four 8-bit data elements are transferred to and from the McBSP by the CPU or the EDMA controller. Four reads of DRR and four writes of DXR are necessary for each frame.

Figure 24-11. Single-Phase Frame of Four 8-Bit Elements



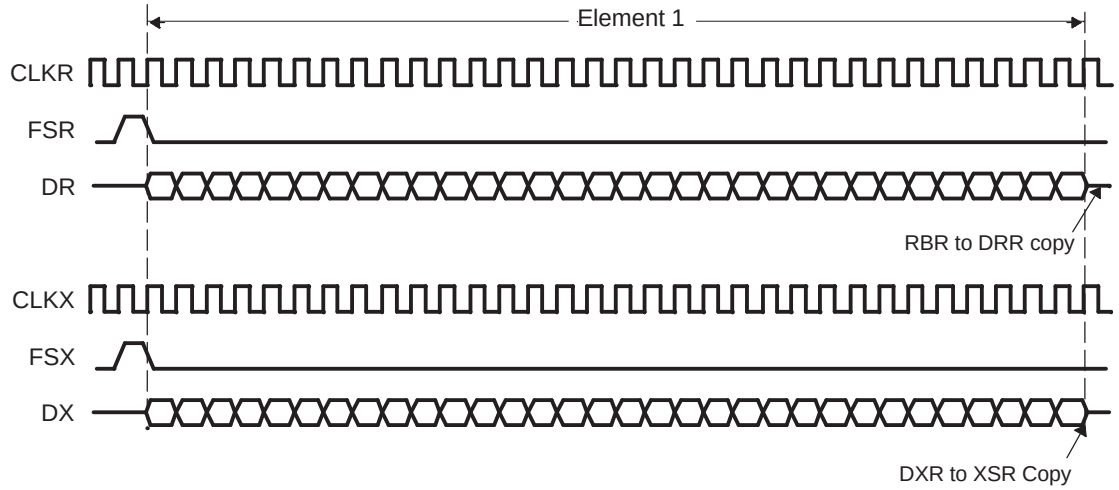
The example in Figure 24-11 can also be viewed as a data stream of a single-phase frame of one 32-bit data element, as shown in Figure 24-12. In this case:

- (R/X)PHASE = 0, indicating a single phase frame
- (R/X)FRLEN1 = 0, indicating a 1-element frame
- (R/X)WDLEN1 = 101b, indicating 32-bit elements

In this example, one 32-bit data element is transferred to and from the McBSP by the CPU or the EDMA controller. Thus, one read of DRR and one write of DXR is necessary for each frame. As a result, the number of transfers is one-fourth that of the previous example (Figure 24-11). This manipulation reduces the percentage of bus time required for serial port data movement.

Copyright © 2013–2016, Texas Instruments Incorporated

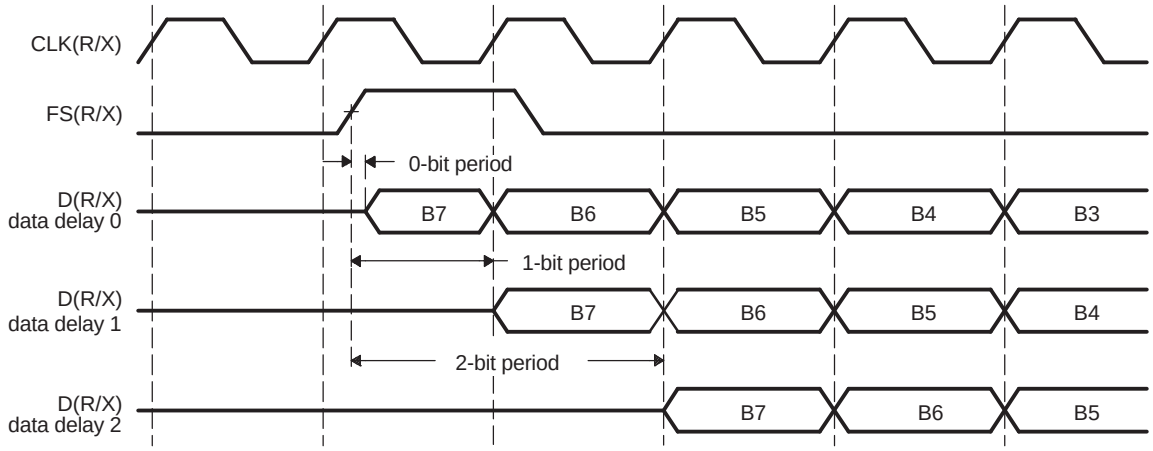
Figure 24-12. Single-Phase Frame of One 32-Bit Element



24.2.5.5.5 Data Delay: RDATDLY and XDATDLY

The start of a frame is defined by the first clock cycle in which frame synchronization is 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 (in RCR) and XDATDLY (in XCR) specify the data delay for reception and transmission, respectively. The range of programmable data delay is zero to two bit clocks ((R/X)DATDLY = 00b to10b), as shown in Figure 24-13. Typically, a 1-bit delay is selected because data often follows a 1-cycle active frame sync pulse.

Figure 24-13. Data Delay

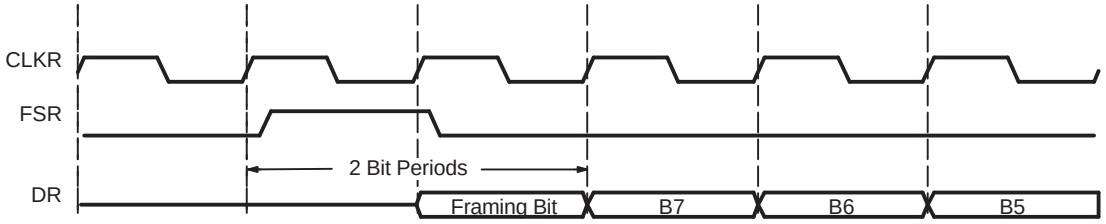


Normally, a frame sync pulse is detected or sampled with respect to an edge of serial clock CLK(R/X). Thus, on a subsequent cycle (depending on data delay value), data can be received or transmitted. However, in the case of a 0-bit data delay, the data must be ready for reception and/or transmission on the same serial clock cycle. For reception, this problem is solved by receive data being sampled on the first falling edge of CLKR when an active (high) FSR is detected. However, data transmission must begin on the rising edge of CLKX that generated the frame synchronization. Therefore, the first data bit is assumed to be in the XSR and DX. The transmitter then asynchronously detects the frame synchronization, FSX goes active, and it immediately starts driving the first bit to be transmitted on the DX pin.

Copyright © 2013–2016, Texas Instruments Incorporated

Another common operation uses a data delay of 2. This configuration allows the serial port to interface to different types of T1 framing devices in which the data stream is preceded by a framing bit. During the reception of such a stream with a data delay of two bits, the 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 24-14. In transmission, by delaying the first transfer bit, the serial port essentially inserts a blank period (high-impedance period) in place of the framing bit. Here, it is expected that the framing device inserts its own framing bit or that the framing bit is generated by another device. Alternatively, you may pull up or pull down DX to achieve the desired value.

Figure 24-14. 2-Bit Data Delay Used to Discard Framing Bit



24.2.5.5.6 Receive Data Justification and Sign Extension: RJUST

The RJUST bit in the serial port control register (SPCR) selects whether data in RBR is right- or left-justified (with respect to the MSB) in the DRR. If right justification is selected, RJUST further selects whether the data is sign-extended or zero-filled. Table 24-10 and Table 24-11 summarize the effect that various values of RJUST have on example receive data.

Table 24-10. Effect of RJUST Bit Values With 12-Bit Example Data ABCh

RJUST Bit in SPCR	Justification	Extension	Value in DRR
00	Right	Zero-fill MSBs	0000 0ABCh
01	Right	Sign-extend MSBs	FFFF FABCh
10	Left	Zero-fill LSBs	ABC0 0000h
11	Reserved	Reserved	Reserved

Table 24-11. Effect of RJUST Bit Values With 20-Bit Example Data ABCDEh

RJUST Bit in SPCR	Justification	Extension	Value in DRR
00	Right	Zero-fill MSBs	000A BCDEh
01	Right	Sign-extend MSBs	FFFA BCDEh
10	Left	Zero-fill LSBs	ABCD E000h
11	Reserved	Reserved	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

24.2.5.5.7 32-Bit Bit Reversal: RWDREVRS, XWDREVRS

Normally all transfers are sent and received with the MSB first; however, you can reverse the receive/transmit bit ordering of a 32-bit element (LSB first) using the 32-bit reversal feature of the McBSP by setting all of the following:

- (R/X)WDREVRS = 1 in the receive/transmit control register (RCR/XCR).
- (R/X)COMPAND = 01b in RCR/XCR.
- (R/X)WDLEN(1/2) = 101b in RCR/XCR to indicate 32-bit elements.

When you set the register fields as above, the bit ordering of the 32-bit element is reversed before being received by or sent from the serial port. If the (R/W)WDREVRS and (R/X)COMPAND fields are set as above, but the element size is not set to 32-bit, operation is undefined.

24.2.6 McBSP Buffer FIFO (BFIFO)

The McBSP Buffer FIFO (BFIFO) provides additional data buffering for the McBSP. The time it takes the host CPU or DMA controller to respond to DMA requests from the McBSP may vary; the additional buffering provided by the BFIFO allows greater tolerance to such variations. For convenience, the BFIFO is treated here as a block between the McBSP and the host/DMA controller (see Figure 24-1). Details on configuration of the BFIFO are provided in Section 24.2.7.6.

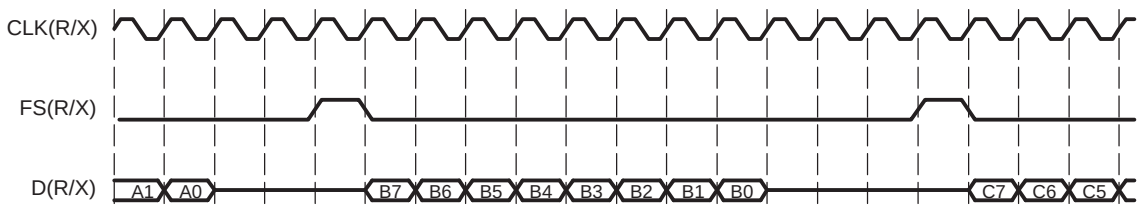
24.2.7 McBSP Standard Operation

During a serial transfer, there are typically periods of serial port inactivity between packets or transfers. The receive and transmit frame synchronization pulse occurs for every serial transfer. When the McBSP is not in the reset state and has been configured for the desired operation, a serial transfer can be initiated by programming (R/X)PHASE = 0 for a single-phase frame with the required number of elements programmed in (R/X)FRLEN1. The number of elements can range from 1 to 128 ((R/X)FRLEN1 = 00h to 7Fh). The required serial element length is set in the (R/X)WDLEN1 field in the (R/X)CR. If a dual-phase frame is required for the transfer, RPHASE = 1 and each (R/X)FRLEN1/2 can be set to any value between 0h and 7Fh.

Figure 24-15 shows a single-phase data frame of one 8-bit element. Since the transfer is configured for a 1-bit data delay, the data on the DX and DR pins are available one bit clock after FS(R/X) goes active. This figure, as well as all others in this section, use the following assumptions:

- (R/X)PHASE = 0, specifying a single-phase frame
- (R/X)FRLEN1 = 0b, specifying one element per frame
- (R/X)WDLEN1 = 000b, specifying eight bits per element
- (R/X)FRLEN2 = (R/X)WDLEN2 = Value is ignored
- CLK(R/X)P = 0, specifying that the receive data is clocked on the falling edge and that transmit data is clocked on the rising edge
- FS(R/X)P = 0, specifying that active (high) frame sync signals are used
- (R/X)DATDLY = 01b, specifying a 1-bit data delay

Figure 24-15. McBSP Standard Operation

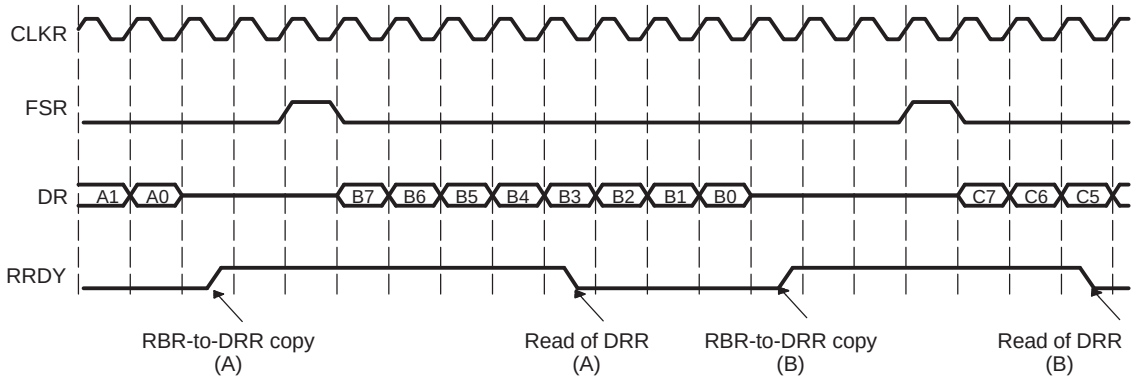


Copyright © 2013–2016, Texas Instruments Incorporated

24.2.7.1 Receive Operation

Figure 24-16 shows serial reception. Once the receive frame synchronization signal (FSR) transitions to its active state, it is detected on the first falling edge of the receivers CLKR. The data on the DR pin is then shifted into the receive shift register (RSR) after the appropriate data delay as set by the RDATDLY bit in the receive control register (RCR). The contents of RSR is copied to RBR at the end of every element on the rising edge of the clock, provided RBR is not full with the previous data. Then, an RBR-to-DRR copy activates the RRDY status bit in the serial port control register (SPCR) to 1 on the following falling edge of CLKR. This indicates that the receive data register (DRR) is ready with the data to be read by the CPU or the EDMA controller. RRDY is deactivated when the DRR is read by the CPU or the EDMA controller. See also Section 24.2.13.1.2 and Section 24.2.14.1.

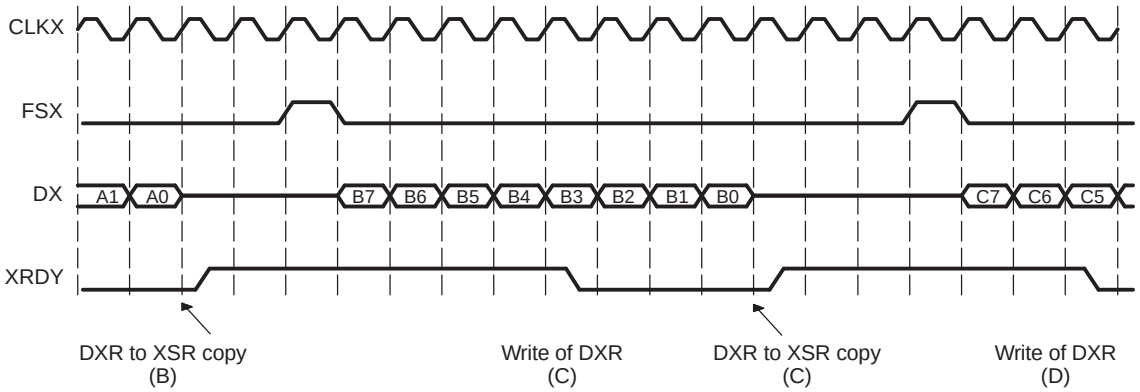
Figure 24-16. Receive Operation



24.2.7.2 Transmit Operation

Once transmit frame synchronization occurs, the value in the transmit shift register (XSR) is shifted out and driven on the DX pin after the appropriate data delay as set by the XDATDLY bit in the transmit control register (XCR). The XRDY bit in the serial port control register (SPCR) is activated after every DXR-to-XSR copy on the following falling edge of CLKX, indicating that the data transmit register (DXR) can be written with the next data to be transmitted. XRDY is deactivated when the DXR is written by the CPU or the EDMA controller. Figure 24-17 illustrates serial transmission. See Section 24.2.7.5.4 for information on transmit operation when the transmitter is pulled out of reset (XRST = 1). See also Section 24.2.13.1.3 and Section 24.2.14.2.

Figure 24-17. Transmit Operation



Copyright © 2013–2016, Texas Instruments Incorporated

24.2.7.3 Maximum Frame Frequency

The frame frequency is determined by the following equation, which calculates the period between frame synchronization signals:

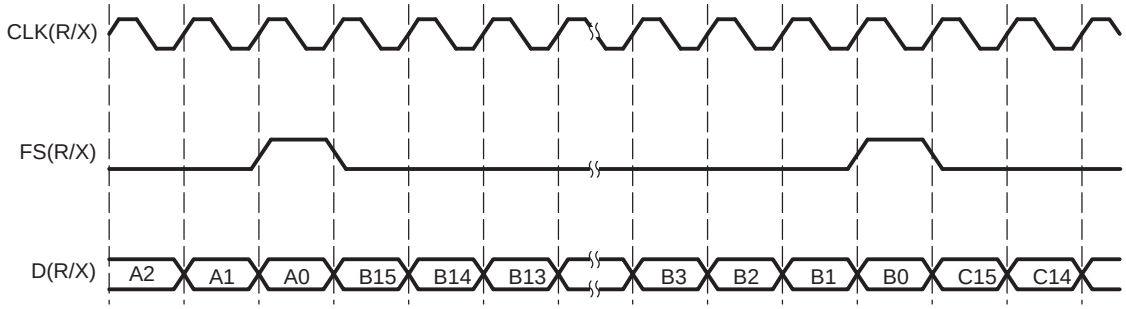
$$\text{Frame frequency} = \frac{\text{Bit-clock frequency}}{\text{Number of bit clocks between frame sync signals}}$$

The frame frequency may be increased by decreasing the time between frame synchronization signals in bit clocks (which is limited only by the number of bits per frame). As the frame transmit frequency is increased, the inactivity period between the data frames for adjacent transfers decreases to 0. The minimum time between frame synchronization pulses is the number of bits transferred per frame. This time also defines the maximum frame frequency, which is calculated by the following equation:

$$\text{Maximum frame frequency} = \frac{\text{Bit-clock frequency}}{\text{Number of bits per frame}}$$

Figure 24-18 shows the McBSP operating at maximum frame frequency. The data bits in consecutive frames are transmitted continuously with no inactivity between bits. If there is a 1-bit data delay, as shown, the frame synchronization pulse overlaps the last bit transmitted in the previous frame.

Figure 24-18. Maximum Frame Frequency for Transmit and Receive



NOTE: For (R/X)DATDLY = 0, the first bit of data transmitted is asynchronous to CLKX, as shown in Figure 24-13.

Maximum frame frequency may not be possible when the word length is 8-bits, depending on the clock divide value CLKGDV. The CPU or EDMA may not be able to service serial port requests that occur as frequently as every 8-bit clocks. This situation can be resolved by allowing additional space between words or choosing a slower bit clock (larger CLKGDV value).

Copyright © 2013–2016, Texas Instruments Incorporated

24.2.7.4 Frame Synchronization Ignore

The McBSP can be configured to ignore transmit and receive frame synchronization pulses. The (R/X)FIG bit in (R/X)CR can be cleared to 0 to recognize frame sync pulses, or it can be set to 1 to ignore frame sync pulses. In this way, you can use (R/X)FIG either to pack data, if operating at maximum frame frequency, or to ignore unexpected frame sync pulses.

24.2.7.4.1 Frame Sync Ignore and Unexpected Frame Sync Pulses

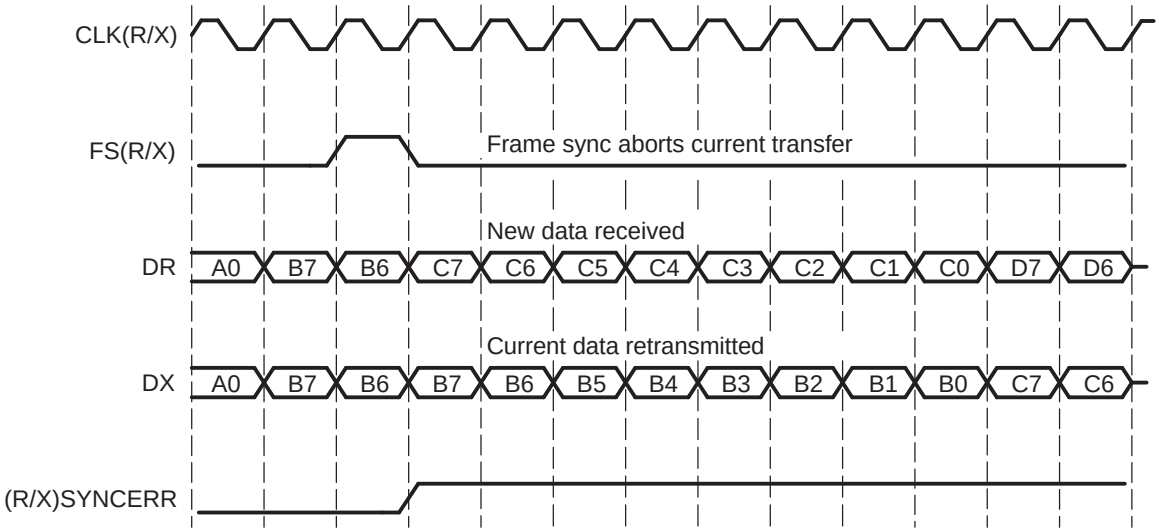
RFIG and XFIG are used to ignore unexpected internal or external frame sync pulses. Any frame sync pulse is considered unexpected if it occurs one or more bit clocks earlier than the programmed data delay from the end of the previous frame specified by ((R/X)DATDLY). Setting the frame ignore bits to 1 causes the serial port to ignore these unexpected frame sync signals.

In reception, if not ignored (RFIG = 0), an unexpected FSR pulse discards the contents of RSR in favor of the incoming data. Therefore, if RFIG = 0, an unexpected frame synchronization pulse aborts the current data transfer, sets RSYNCERR in SPCR to 1, and begins the reception of a new data element. When RFIG = 1, the unexpected frame sync pulses are ignored.

In transmission, if not ignored (XFIG = 0), an unexpected FSX pulse aborts the ongoing transmission, sets the XSYNCERR bit in SPCR to 1, and reinitiates transmission of the current element that was aborted. When XFIG = 1, unexpected frame sync signals are ignored.

Figure 24-19 shows that element B is interrupted by an unexpected frame sync pulse when (R/X)FIG = 0. The reception of B is aborted (B is lost), and a new data element (C) is received after the appropriate data delay. This condition causes a receive synchronization error and thus sets the RSYNCERR bit. However, for transmission, the transmission of B is aborted and the same data (B) is retransmitted after the appropriate data delay. This condition is a transmit synchronization error and thus sets the XSYNCERR bit. Synchronization errors are discussed in Section 24.2.7.5.2 and Section 24.2.7.5.5.

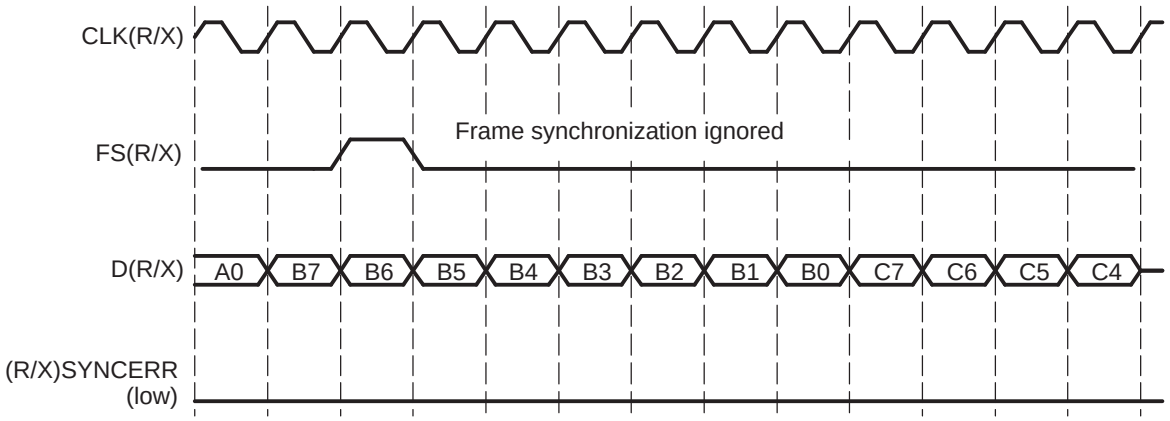
Figure 24-19. Unexpected Frame Synchronization With (R/X)FIG = 0



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 24-20 shows McBSP operation when unexpected internal or external frame synchronization signals are ignored by setting (R/X)FIG = 1. Here, the transfer of element B is not affected by an unexpected frame synchronization.

Figure 24-20. Unexpected Frame Synchronization With (R/X)FIG = 1

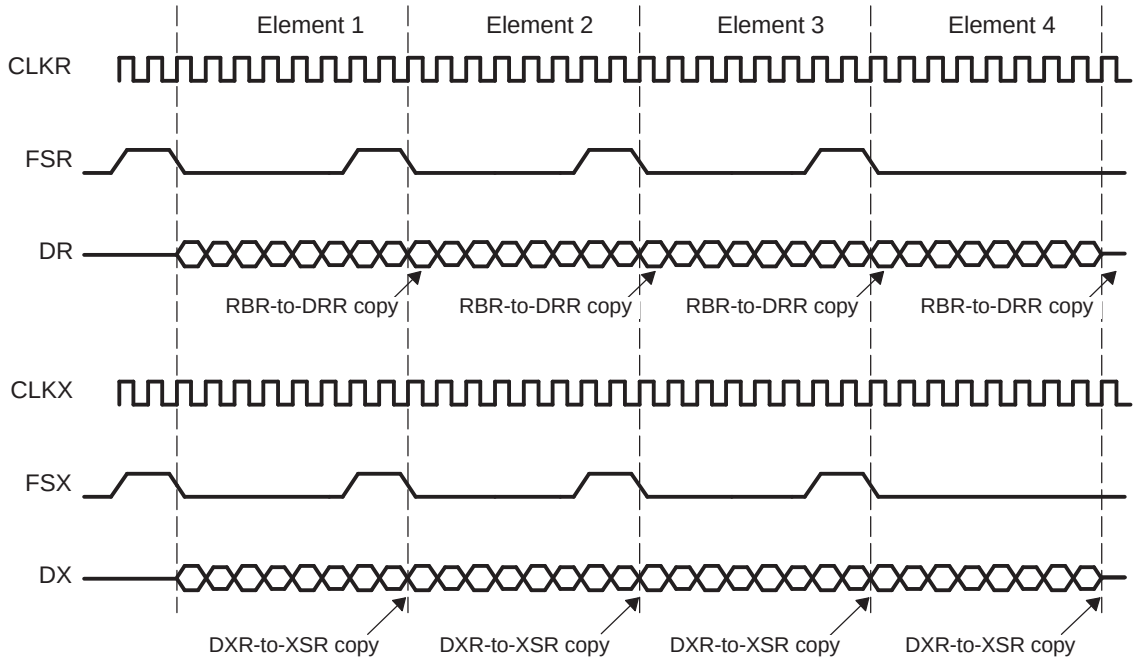


24.2.7.4.2 Data Packing using Frame Sync Ignore Bits

Section 24.2.5.5.4 describes one method of changing the element length and frame length to simulate 32-bit serial element transfers, thus requiring much less bus bandwidth than four 8-bit transfers require. This example works when there are multiple elements per frame.

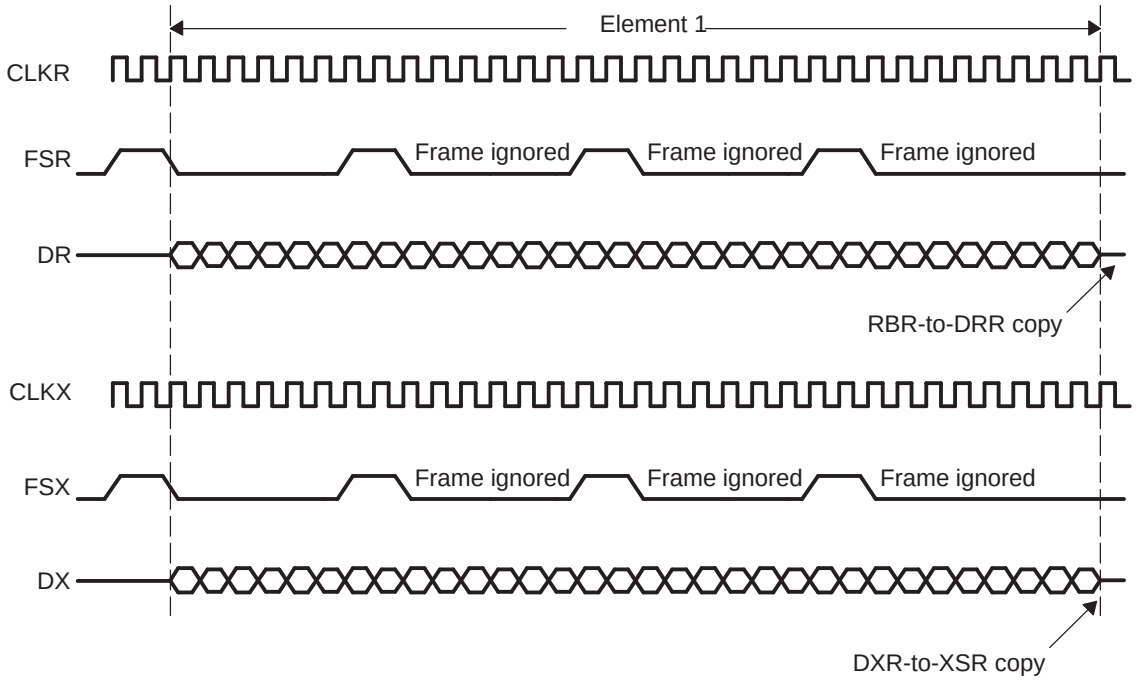
Now consider the case of the McBSP operating at maximum packet frequency, as shown in Figure 24-21. Here, each frame has only a single 8-bit element. This stream takes one read transfer and one write transfer for each 8-bit element. Figure 24-22 shows the McBSP configured to treat this stream as a continuous stream of 32-bit elements. In this example, (R/X)FIG is set to 1 to ignore unexpected subsequent frames. Only one read transfer and one write transfer is needed every 32 bits. This configuration effectively reduces the required bus bandwidth to one-fourth of the bandwidth needed to transfer four 8-bit blocks.

Figure 24-21. Maximum Frame Frequency Operation With 8-Bit Data



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 24-22. Data Packing at Maximum Frame Frequency With (R/X)FIG = 1



24.2.7.5 Serial Port Exception Conditions

There are five serial port events that can constitute a system error:

- Receive overrun (RFULL = 1 in SPCR)
- Unexpected receive frame synchronization (RSYNCERR = 1 in SPCR)
- Transmit data overwrite
- Transmit empty (XEMPTY = 0 in SPCR)
- Unexpected transmit frame synchronization (XSYNCERR = 1 in SPCR)

24.2.7.5.1 Receive Overrun: RFULL

RFULL = 1 in the serial port control register (SPCR) indicates that the receiver has experienced overrun and is in an error condition. RFULL is set when the following conditions are met:

- DRR has not been read since the last RBR-to-DRR transfer.
- RBR is full and an RBR-to-DRR copy has not occurred.
- RSR is full and an RSR-to-RBR transfer has not occurred.

The data arriving on DR is continuously shifted into RSR (Figure 24-23). Once a complete element is shifted into RSR, an RSR-to-RBR transfer can occur only if an RBR-to-DRR copy is complete. Therefore, if DRR has not been read by the CPU or the EDMA controller since the last RBR-to-DRR transfer (RRDY = 1), an RBR-to-DRR copy does not take place until RRDY = 0. This prevents an RSR-to-RBR copy. New data arriving on the DR pin is shifted into RSR, and the previous contents of RSR are lost. After the receiver starts running from reset, a minimum of three elements must be received before RFULL can be set, because there was no last RBR-to-DRR transfer before the first element.

This data loss can be avoided if DRR is read no later than two and a half CLKR cycles before the end of the third element (data C) in RSR, as shown in Figure 24-24.

Copyright © 2013–2016, Texas Instruments Incorporated

Either of the following events clears the RFULL bit to 0 and allows subsequent transfers to be read properly:

- Reading DRR
- Resetting the receiver (RRST = 0) or the device

Another frame synchronization is required to restart the receiver.

Figure 24-23 shows the receive overrun condition. Because element A is not read before the reception of element B is complete, B is not transferred to DRR yet. Another element, C, arrives and fills RSR. DRR is finally read, but not earlier than two and one half cycles before the end of element C. New data D overwrites the previous element C in RSR. If RFULL is still set after the DRR is read, the next element can overwrite D if DRR is not read in time.

Figure 24-24 shows the case in which RFULL is set but the overrun condition is averted by reading the contents of DRR at least two and a half cycles before the next element, C, is completely shifted into RSR. This ensures that a RBR-to-DRR copy of data B occurs before the next element is transferred from RSR to RBR.

Figure 24-23. Serial Port Receive Overrun

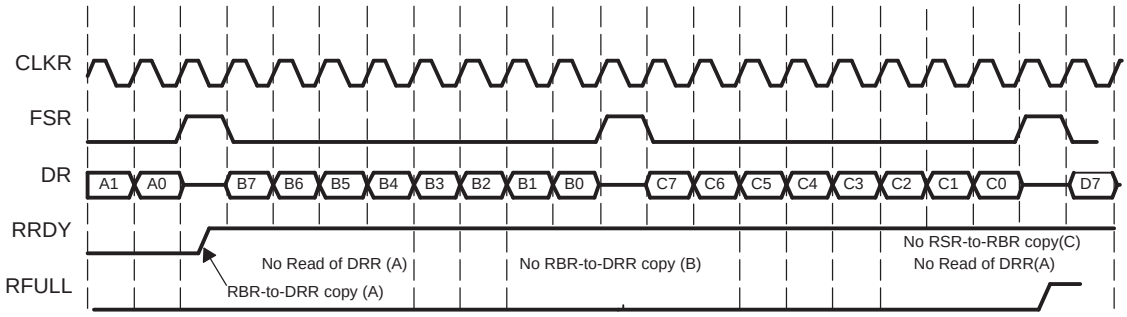
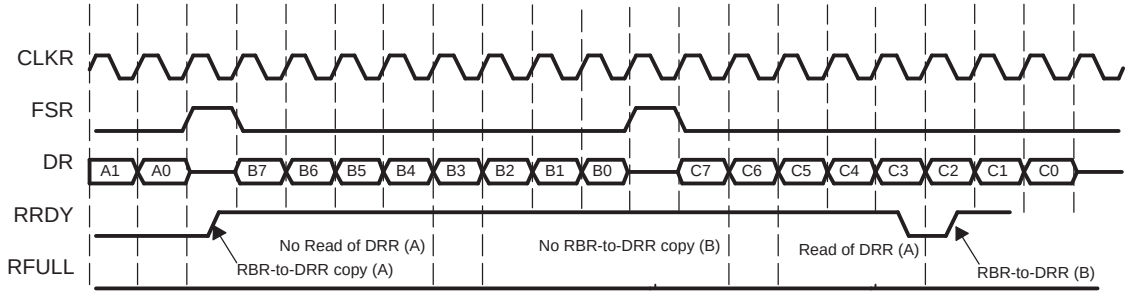


Figure 24-24. Serial Port Receive Overrun Avoided



Copyright © 2013–2016, Texas Instruments Incorporated

24.2.7.5.2 Unexpected Receive Frame Synchronization: RSYNCERR

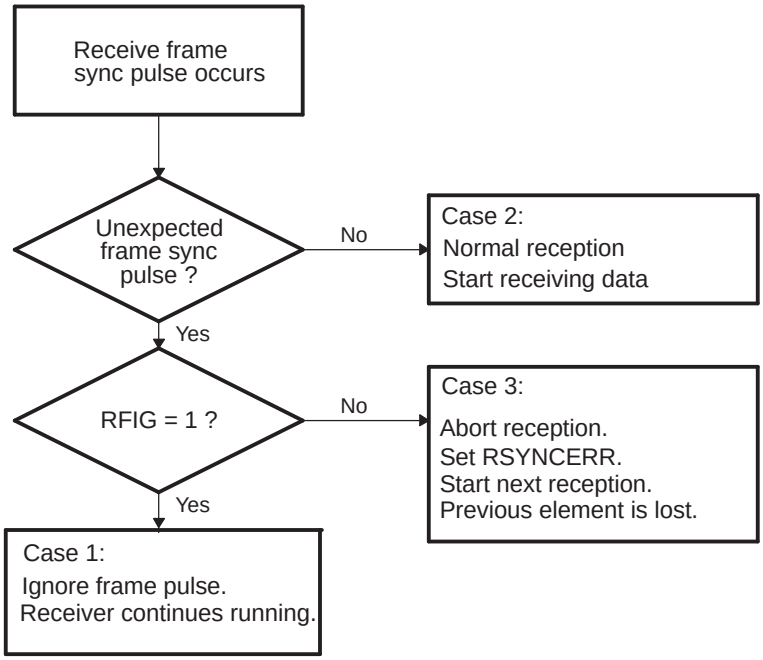
Figure 24-25 shows the decision tree that the receiver uses to handle all incoming frame synchronization pulses. The diagram assumes that the receiver has been activated (RRST = 1). Unexpected frame sync pulses can originate from an external source or from the internal sample rate generator. An unexpected frame sync pulse is defined as a sync pulse which occurs RDATDLY bit clocks earlier than the last transmitted bit of the previous frame. Any one of three cases can occur:

- Case 1: Unexpected FSR pulses with RFIG = 1. This case is discussed in Section 24.2.7.4.1 and shown in Figure 24-20. Here, receive frame sync pulses are ignored and the reception continues.
- Case 2: Normal serial port reception. There are three reasons for a receive not to be in progress:
  - This FSR is the first after RRST = 1.
  - This FSR is the first after DRR has been read clearing an RFULL condition.
  - The serial port is in the inter-packet intervals. The programmed data delay (RDATDLY) for reception may start during these inter-packet intervals for the first bit of the next element to be received. Thus, at maximum frame frequency, frame synchronization can still be received RDATDLY bit clocks before the first bit of the associated element.

    For this case, reception continues normally, because these are not unexpected frame sync pulses.
- Case 3: Unexpected receive frame synchronization with RFIG = 0 (unexpected frame not ignored). This case was shown in Figure 24-19 for maximum packet frequency. Figure 24-26 shows this case during normal operation of the serial port with time intervals between packets. Unexpected frame sync pulses are detected when they occur the value in RDATDLY bit clocks before the last bit of the previous element is received on DR. In both cases, RSYNCERR in SPCR is set. RSYNCERR can be cleared only by receiver reset or by writing a 0 to this bit in SPCR. If RINTM = 11b in SPCR, RSYNCERR drives the receive interrupt (RINT) to the CPU.

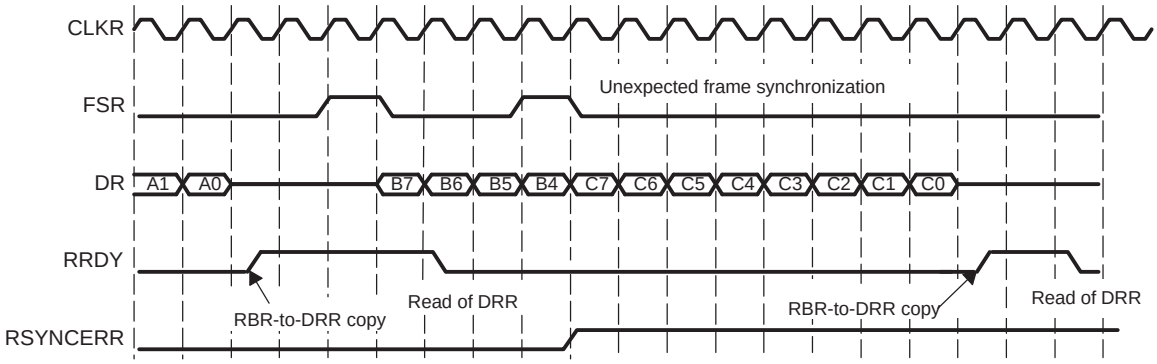
NOTE: Note that the RSYNCERR bit in SPCR is a read/write bit, so writing a 1 to it sets the error condition. Typically, writing a 0 is expected.

Figure 24-25. Decision Tree Response to Receive Frame Synchronization Pulse



Copyright © 2013–2016, Texas Instruments Incorporated

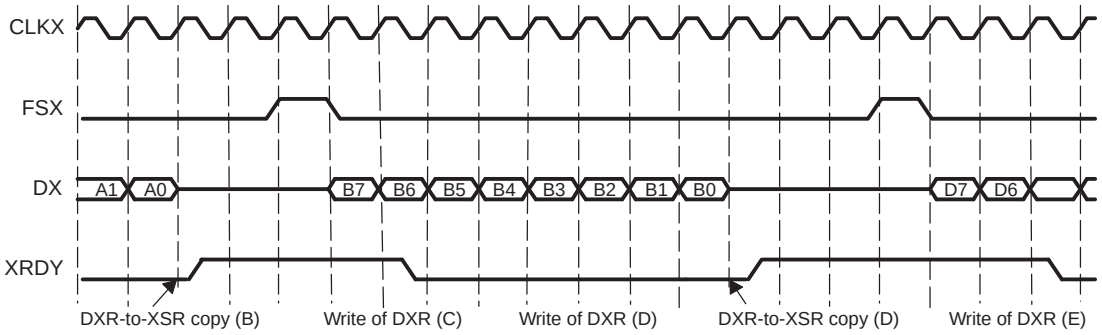
Figure 24-26. Unexpected Receive Frame Synchronization Pulse



24.2.7.5.3 Transmit With Data Overwrite

Figure 24-27 shows what happens if the data in DXR is overwritten before it is transmitted. Suppose you load the DXR with data C. A subsequent write to the DXR overwrites C with D before C is copied to the XSR. Thus, C is never transmitted on DX. The CPU can avoid overwriting data by polling XRDY before writing to DXR or by waiting for a programmed XINT to be triggered by XRDY (XINTM = 00b). The EDMA controller can avoid overwriting by synchronizing data writes with XEVT. See also Section 24.2.13.1.3.

Figure 24-27. Transmit With Data Overwrite



Copyright © 2013–2016, Texas Instruments Incorporated

24.2.7.5.4 Transmit Empty: XEMPTY

XEMPTY indicates whether the transmitter has experienced underflow. Either of the following conditions causes XEMPTY to become active (XEMPTY = 0):

- During transmission, DXR has not been loaded since the last DXR-to-XSR copy, and all bits of the data element in XSR have been shifted out on DX.
- The transmitter is reset (XRST = 0 or the device is reset), and then restarted.

During an underflow condition, the transmitter continues to transmit the old data in DXR for every new frame sync signal FSX (generated by an external device, or by the internal sample rate generator) until a new element is loaded into DXR by the CPU or the EDMA controller. XEMPTY is deactivated (XEMPTY = 1) when this new element in DXR is transferred to XSR. In the case when the FSX is generated by a DXR-to-XSR copy (FSXM = 1 in PCR and FSGM = 0 in SRGR), the McBSP does not generate any new frame sync until new data is written to the DXR and a DXR-to-XSR copy occurs.

When the transmitter is taken out of reset (XRST = 1), it is in a transmit ready (XRDY = 1) and transmit empty (XEMPTY = 0) condition. If DXR is loaded by the CPU or the EDMA controller before FSX goes active, a valid DXR-to-XSR transfer occurs. This allows for the first element of the first frame to be valid even before the transmit frame sync pulse is generated or detected. Alternatively, if a transmit frame sync is detected before DXR is loaded, 0s are output on DX.

Figure 24-28 shows a transmit underflow condition. After B is transmitted, B is retransmitted on DX if you fail to reload the DXR before the subsequent frame synchronization. Figure 24-29 shows the case of writing to DXR just before a transmit underflow condition that would otherwise occur. After B is transmitted, C is written to DXR before the next transmit frame sync pulse occurs.

Figure 24-28. Transmit Empty

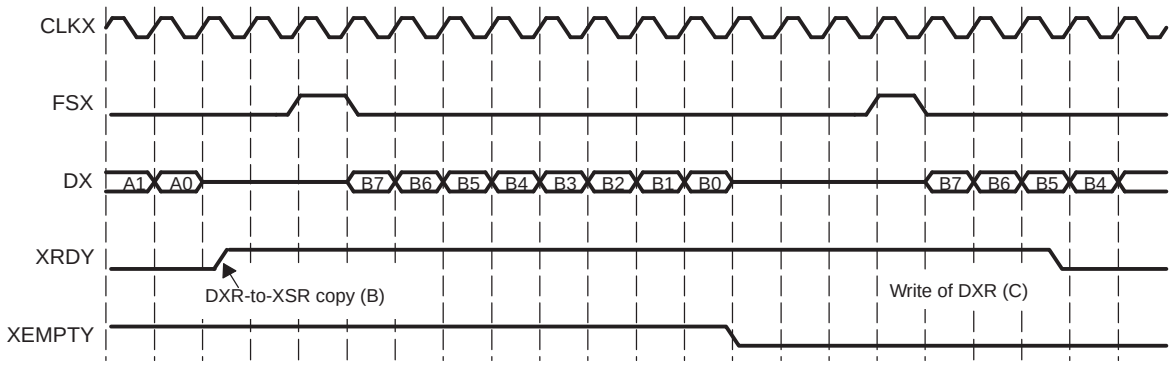
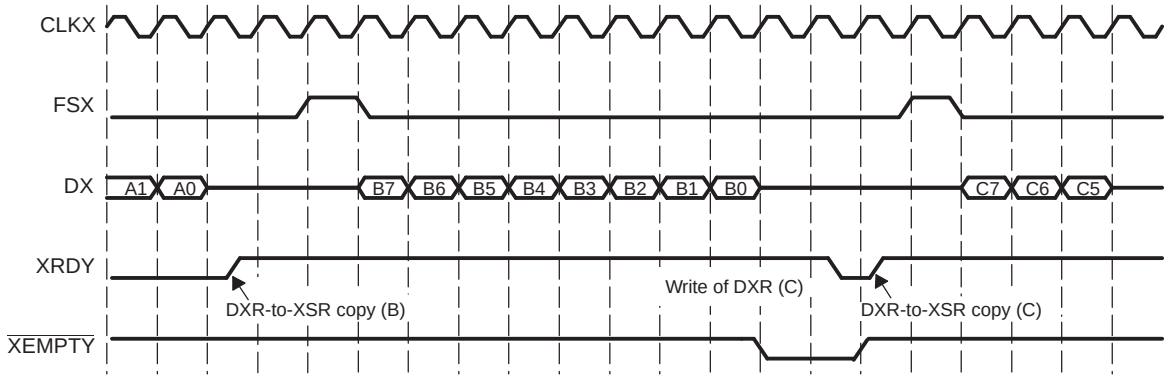


Figure 24-29. Transmit Empty Avoided



Copyright © 2013–2016, Texas Instruments Incorporated

24.2.7.5.5 Unexpected Transmit Frame Synchronization: XSYNCERR

A transmit frame sync error (XSYNCERR) may occur the first time the transmitter is enabled (XRST = 1) after a device reset. To avoid this, after enabling the transmitter for the first time, the following procedure must be followed:

1. Wait for two CLKG cycles. The unexpected frame sync error (XSYNCERR), if any, occurs within this time period.
2. Disable the transmitter (XRST = 0). This clears any XSYNCERR.
3. Re-enable the transmitter (XRST = 1).

See also Section 24.2.12 for details on initialization procedure.

Figure 24-30 shows the decision tree that the transmitter uses to handle all incoming frame synchronization signals. The diagram assumes that the transmitter has been started (XRST = 1). An unexpected transmit frame sync pulse is defined as a sync pulse that occurs XDATDLY bit clocks earlier than the last transmitted bit of the previous frame. Any one of three cases can occur:

- Case 1: Unexpected FSX pulses with XFIG = 1. This case is discussed in Section 24.2.7.4.1 and shown in Figure 24-20. In this case, unexpected FSX pulses are ignored, and the transmission continues.
- Case 2: FSX pulses with normal serial port transmission. This situation is discussed in Section 24.2.7.2. There are two possible reasons for a transmit not to be in progress:
  - This FSX pulse is the first one to occur after XRST = 1.
  - The serial port is in the interpacket intervals. The programmed data delay (XDATDLY) may start during these interpacket intervals before the first bit of the next element is transmitted. Thus, if operating at maximum packet frequency, frame synchronization can still be received XDATDLY bit clocks before the first bit of the associated element.
- Case 3: Unexpected transmit frame synchronization with XFIG = 0. The case was shown in Figure 24-19 for frame synchronization with XFIG = 0 at maximum packet frequency. Figure 24-31 shows the case for normal operation of the serial port with interpacket intervals. In both cases, XSYNCERR in SPCR is set. XSYNCERR can be cleared only by transmitter reset or by writing a 0 to this bit in SPCR. If XINTM = 11b in SPCR, XSYNCERR drives the receive interrupt (XINT) to the CPU.

NOTE: The XSYNCERR bit in SPCR is a read/write bit, so writing a 1 to it sets the error condition. Typically, writing a 0 is expected.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 24-30. Decision Tree Response to Transmit Frame Synchronization Pulse

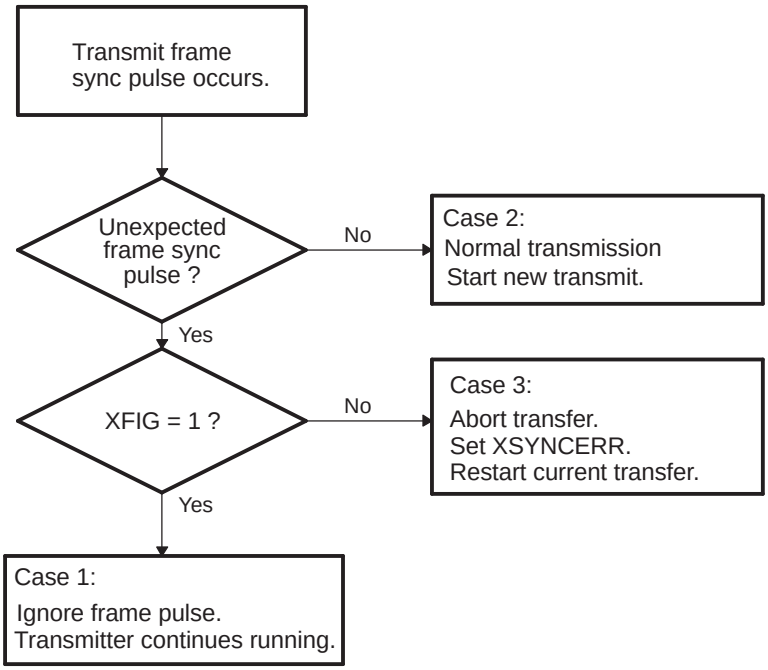
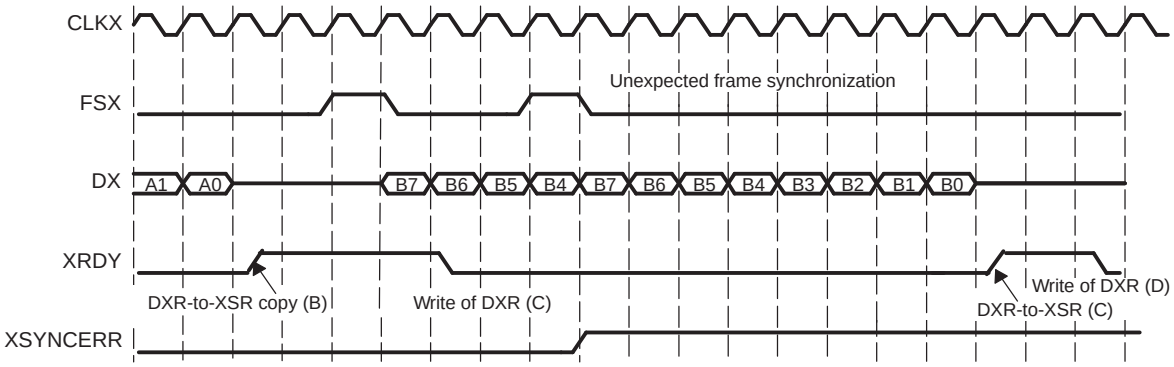


Figure 24-31. Unexpected Transmit Frame Synchronization Pulse



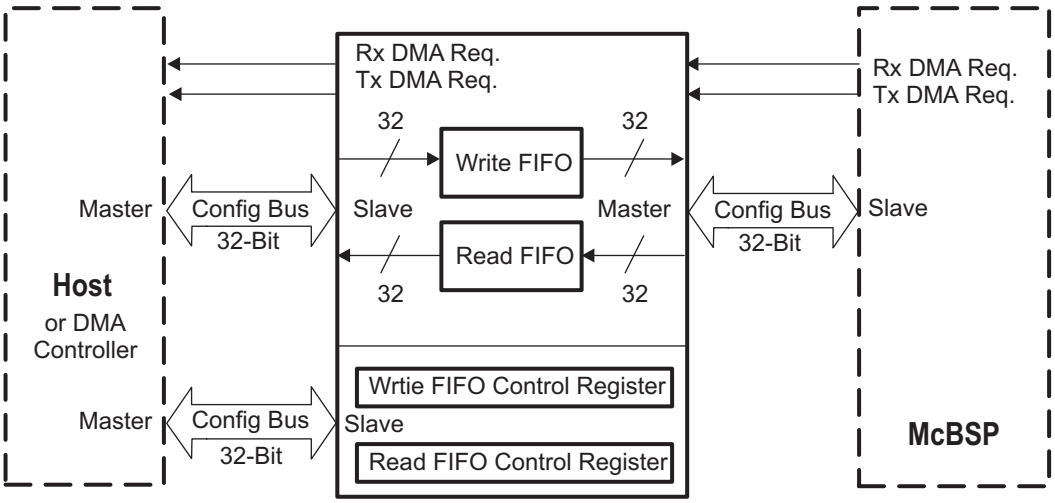
Copyright © 2013–2016, Texas Instruments Incorporated

24.2.7.6 McBSP Buffer FIFO (BFIFO)

The BFIFO contains two FIFOs: one Read FIFO (RFIFO) and one Write FIFO (WFIFO). To ensure backward compatibility with existing software, both the Read and Write FIFOs are disabled by default. See Figure 24-32 for a high-level block diagram of the BFIFO.

The BFIFO may be enabled/disabled and configured via the Write FIFO control register (WFIFOCTL) and the Read FIFO control register (RFIFOCTL). Note that if the Read or Write FIFO is to be enabled, it must be enabled prior to initializing the receive/transmit section of the McBSP (see Section 24.2.12 for details).

Figure 24-32. McBSP Buffer FIFO (BFIFO) Block Diagram



Note that the McBSP Buffer FIFO (BFIFO) has a different memory-map (see your device-specific data manual) than the McBSP memory-mapped registers (MMRs); hence, the BFIFO is accessible by way of a different Configuration Bus.

24.2.7.6.1 BFIFO Data Transmission

When the Write FIFO is disabled, transmit DMA requests pass through directly from the McBSP to the host/DMA controller. Whether the WFIFO is enabled or disabled, the McBSP generates transmit DMA requests as needed; the BFIFO is "invisible" to the McBSP.

When the Write FIFO is enabled, transmit DMA requests from the McBSP are sent to the BFIFO, which in turn generates transmit DMA requests to the host/DMA controller.

If the Write FIFO is enabled, upon a transmit DMA request from the McBSP, the WFIFO writes *WNUMDMA* 32-bit words to the McBSP, if and when there are at least *WNUMDMA* words in the Write FIFO. If there are not, the WFIFO waits until this condition has been satisfied; at this point, it writes *WNUMDMA* words to the McBSP.

If the host CPU writes to the Write FIFO, independent of a transmit DMA request, the WFIFO will accept host writes until full. After this point, excess data will be discarded.

Note that when the WFIFO is first enabled, it will immediately issue a transmit DMA request to the host. This is because it begins in an empty state, and is therefore ready to accept data.

Copyright © 2013–2016, Texas Instruments Incorporated

24.2.7.6.1.1 Transmit DMA Event Pacer

The BFIFO may be configured to delay making a transmit DMA request to the host until the Write FIFO has enough space for a specified number of words. In this situation, the number of transmit DMA requests to the host or DMA controller is reduced.

If the Write FIFO has space to accept *WNUMEVT* 32-bit words, it generates a transmit DMA request to the host and then waits for a response. Once *WNUMEVT* words have been written to the WFIFO, the WFIFO checks again to see if there is space for *WNUMEVT* 32-bit words. If there is space, it generates another transmit DMA request to the host, and so on. In this fashion, the Write FIFO will attempt to stay filled.

Note that if transmit DMA event pacing is desired, the WNUMEVT bits in WFIFOCTL should be set to a non-zero integer multiple of the value in the WNUMDMA bits. If transmit DMA event pacing is not desired, then the value in the WNUMEVT bits should be set equal to the value in the WNUMDMA bits.

24.2.7.6.2 BFIFO Data Reception

When the Read FIFO is disabled, receive DMA requests pass through directly from McBSP to the host/DMA controller. Whether the RFIFO is enabled or disabled, the McBSP generates receive DMA requests as needed; the BFIFO is "invisible" to the McBSP.

When the Read FIFO is enabled, receive DMA requests from the McBSP are sent to the BFIFO, which in turn generates receive DMA requests to the host/DMA controller.

If the Read FIFO is enabled and the McBSP makes a receive DMA request, the RFIFO reads *RNUMDMA* 32-bit words from the McBSP, if and when the RFIFO has space for *RNUMDMA* words. If the RFIFO does not have space, the RFIFO waits until this condition has been satisfied; at this point, it reads *RNUMDMA* words from the McBSP.

If the host CPU reads the Read FIFO, independent of a receive DMA request, and the RFIFO at that time contains less than *RNUMEVT* words, those words will be read correctly, emptying the RFIFO.

24.2.7.6.2.1 Receive DMA Event Pacer

The BFIFO may be configured to delay making a receive DMA request to the host until the Read FIFO contains a specified number of words. In this situation, the number of receive DMA requests to the host or DMA controller is reduced.

If the Read FIFO contains at least *RNUMEVT* 32-bit words, it generates a receive DMA request to the host and then waits for a response. Once *RNUMEVT* 32-bit words have been read from the RFIFO, the RFIFO checks again to see if it contains at least another *RNUMEVT* words. If it does, it generates another receive DMA request to the host, and so on. In this fashion, the Read FIFO will attempt to stay empty.

Note that if receive DMA event pacing is desired, the RNUMEVT bits in RFIFOCTL should be set to a non-zero integer multiple of the value in RNUMDMA bits. If receive DMA event pacing is not desired, then the value in the RNUMEVT bits should be set equal to the value in the RNUMDMA bits.

24.2.7.6.3 Arbitration Between Transmit and Receive DMA Requests

If both the WFIFO and the RFIFO are enabled and a transmit DMA request and receive DMA request occur simultaneously, priority is given to the transmit DMA request. Once a transfer is in progress, it is allowed to complete.

If only the WFIFO is enabled and a transmit DMA request and receive DMA request occur simultaneously, priority is given to the transmit DMA request. Once a transfer is in progress, it is allowed to complete.

If only the RFIFO is enabled and a transmit DMA request and receive DMA request occur simultaneously, priority is given to the receive DMA request. Once a transfer is in progress, it is allowed to complete.

Copyright © 2013–2016, Texas Instruments Incorporated

24.2.8 μ-Law/A-Law Companding Hardware Operation

Companding (compressing and expanding) hardware allows compression and expansion of data in either μ-law or A-law format. The specification for μ-law and A-law log PCM is part of the CCITT G.711 recommendation. The companding standard employed in the United States and Japan is μ-law and allows 14 bits of dynamic range. The European companding standard is A-law and allows 13 bits of dynamic range. Any values outside these ranges are set to the most positive or most negative value. Thus, for companding to work best here, the data transferred to and from the McBSP via the CPU or the EDMA controller must be at least 16 bits wide.

The μ-law and A-law formats encode data into 8-bit code elements. Companded data is always 8 bits wide, so the appropriate (R/X)WDLEN1/2 must be cleared to 0, indicating an 8-bit serial data stream. If companding is enabled and either phase of the frame does not have an 8-bit element length, companding continues as if the element length is eight bits.

When companding is used, transmit data is encoded according to the specified companding law, and receive data is decoded to 2s-complement format. Companding is enabled and the desired format is selected by appropriately setting (R/X)COMPAND in the (R/X)CR. Compression occurs during the process of copying data from DXR to XSR and expansion occurs from RBR to DRR, as shown in Figure 24-33.

For transmit data to be compressed, it should be 16-bit, left-justified data, such as LAW16, as shown in Figure 24-34. The value can be either 13 or 14 bits wide, depending on the companding law. This 16-bit data is aligned in DXR, as shown in Figure 24-35.

For reception, the 8-bit compressed data in RBR is expanded to a left-justified 16-bit data, LAW16. This can be further justified to 32-bit data by programming the RJUST bits in the serial port control register (SPCR), as shown in Table 24-12.

Figure 24-33. Companding Flow

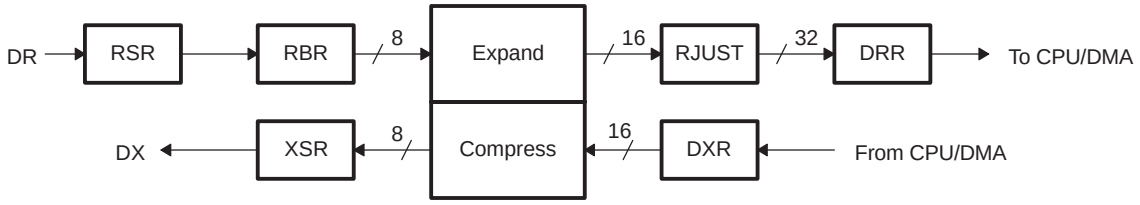


Figure 24-34. Companding Data Formats

LAW16	15	2	1	0
μ-Law	Value		0	0

LAW16	15	3	2	1	0
A-Law	Value		0	0	0

Figure 24-35. Transmit Data Companding Format in DXR

31	16	15	0
Don't care		LAW16	

Copyright © 2013–2016, Texas Instruments Incorporated

Table 24-12. Justification of Expanded Data in DRR

RJUST Bit in SPCR	DRR Bits			
	31	16	15	0
00	0			LAW16
01	sign			LAW16
10	LAW16			0
11	Reserved			

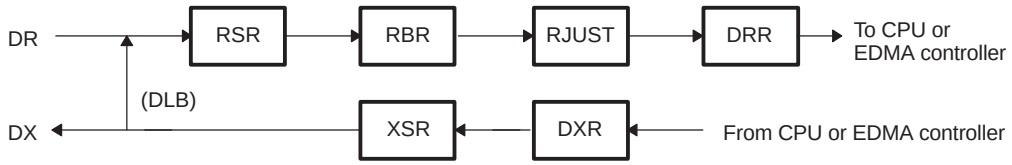
24.2.8.1 Companding Internal Data

If the McBSP is unused, the companding hardware can compand internal data. This hardware can be used to:

- Convert linear data to the appropriate μ-law or A-law format.
- Convert μ-law or A-law data to the linear format.
- Observe the companding 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 24-36 shows the method by which the McBSP can compand internal data. The data path is indicated by the (DLB) arrow. The McBSP is enabled in digital loopback (DLB) mode with companding appropriately enabled by the RCOMPAND and XCOMPAND bits. 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 the EDMA controller to these conversions, respectively. The time for this companding depends on the serial bit rate selected.

Figure 24-36. Companding of Internal Data



24.2.8.2 Bit Ordering

Normally, all transfers on the McBSP are sent and received with the MSB first. However, certain 8-bit data protocols (that do not use companded data) require the LSB to be transferred first. By setting the (R/X)COMPAND = 01b in (R/X)CR, the bit ordering of 8-bit elements is reversed (LSB first) before being sent to the serial port. Like the companding feature, this feature is enabled only if the appropriate (R/X)WDLEN1/2 bit is cleared to 0, indicating that 8-bit elements are to be serially transferred. A 32-bit bit reversal feature is also available, as shown in Section 24.2.5.5.7

Copyright © 2013–2016, Texas Instruments Incorporated

24.2.9 Multichannel Selection Modes

This section defines and provides the functions and all related information concerning the multichannel selection modes.

24.2.9.1 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:

Block 0: Channels 0 through 15
Block 1: Channels 16 through 31
Block 2: Channels 32 through 47
Block 3: Channels 48 through 63
Block 4: Channels 64 through 79
Block 5: Channels 80 through 95
Block 6: Channels 96 through 111
Block 7: Channels 112 through 127

The blocks are assigned to partitions according to the selected partition mode. In the 2-partition mode, 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, blocks 0 through 7 are automatically assigned to partitions, A through H, respectively.

The number of partitions for reception and the number of partitions for transmission are independent. For example, it is possible to use 2 receive partitions (A and B) and 8 transmit partitions (A through H).

24.2.9.2 Multichannel Selection

When 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. The McBSP has one receive multichannel selection mode and three transmit multichannel selection modes.

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.

24.2.9.3 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 (RFRLEN1/XFRLEN1) that includes the highest-numbered channel that is 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.

NOTE: The frame-sync pulse can be generated internally by the sample rate generator or it can be supplied externally by another source. In a multichannel mode configuration with external frame-sync generation, the McBSP transmitter will ignore the first frame-sync pulse after it is taken out of reset. The transmitter will transmit only on the second frame-sync pulse. The receiver will shift in data on the first frame-sync pulse, regardless of whether it is generated internally or externally.

Copyright © 2013–2016, Texas Instruments Incorporated

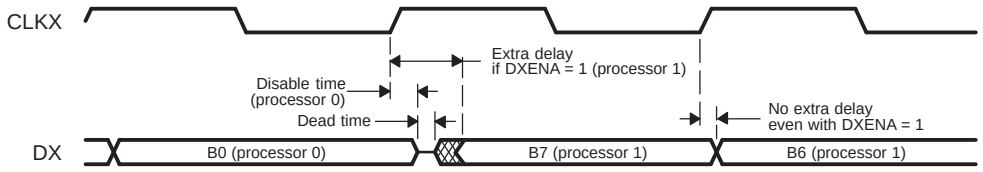
24.2.9.4 DX Enabler: DXENA

The DXENA bit in the serial port control register (SPCR) controls the high-impedance enable on the DX pin. When DXENA = 1, the McBSP enables extra delay for the DX pin turn-on time. This feature is useful for McBSP multichannel operations, such as in a time-division multiplexed (TDM) system. The McBSP supports up to 128 channels in a multichannel operation. These channels can be driven by different devices in a TDM data communication line, such as the T1/E1 line. In any multichannel operation where multiple devices transmit over the same DX line, you need to ensure that no two devices transmit data simultaneously, which results in bus contention. Enough dead time should exist between the transmission of the first data bit of the current device and the transmission of the last data bit of the previous device. In other words, the last data bit of the previous device needs to be disabled to a high-impedance state before the next device begins transmitting data to the same data line, as shown in Figure 24-37.

When two McBSPs are used to transmit data over the same TDM line, bus contention occurs if DXENA = 0. The first McBSP turns off the transmission of the last data bit (changes DX from valid to a high-impedance state) after a disable time specified in the data manual. As shown in Figure 24-37, this disable time is measured from the CLKX active clock edge. The next McBSP turns on its DX pin (changes from a high-impedance state to valid) after a delay time. Again, this delay time is measured from the CLKX active clock edge. Bus contention occurs because the dead time between the two devices is not enough. You need to apply alternative software or hardware methods to ensure proper multichannel operation in this case.

If you set DXENA = 1 in the second McBSP, the second McBSP turns on its DX pin after some extra delay time. This ensures that the previous MMcBSP on the same DX line is disabled before the second McBSP starts driving out data. The DX enabler controls only the high-impedance enable on the DX pin, not the data itself. Data is shifted out to the DX pin at the same time as in the case when DXENA = 0. The only difference is that with DXENA = 1, the DX pin is masked to a high-impedance state for some extra CPU cycles before the data is seen on the TDM data line. Therefore, only the first bit of data is delayed. Refer to the specific device datasheet for the exact amount of delay.

Figure 24-37. DX Timing for Multichannel Operation



24.2.9.5 Using Two Partitions

For multichannel selection operation in the receiver and/or the transmitter, you can use two partitions or eight partitions. 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-sync 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-sync pulse occurs, the next frame is transferred, beginning with the channels in partition A.

Copyright © 2013–2016, Texas Instruments Incorporated

24.2.9.5.1 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 (see Table 24-13), 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 (see Table 24-14). You can dynamically change which blocks of channels are assigned to the partitions, see Section 24.2.9.5.2.

For reception:

- Assign an even-numbered channel block (0, 2, 4, or 6) to receive partition A by writing to the RPABLK bit in the multichannel control register (MCR). In the receive multichannel selection mode, the channels in this partition are controlled by the enhanced receive channel enable register partition A/B (RCERE0).
- Assign an odd-numbered block (1, 3, 5, or 7) to receive partition B with the RPBBLK bit in MCR. In the receive multichannel selection mode, the channels in this partition are controlled by the enhanced receive channel enable register partition A/B (RCERE0).

For transmission:

- Assign an even-numbered channel block (0, 2, 4, or 6) to transmit partition A by writing to the XPABLK bit in the multichannel control register (MCR). In one of the transmit multichannel selection modes, the channels in this partition are controlled by the enhanced transmit channel enable register partition A/B (XCERE0).
- Assign an odd-numbered block (1, 3, 5, or 7) to transmit partition B with the XPBBLK bit in MCR. In one of the transmit multichannel selection modes, the channels in this partition are controlled by the enhanced transmit channel enable register partition A/B (XCERE0).

Table 24-13. Receive Channel Assignment and Control When Two Receive Partitions are Used

Receive Partition	Assigned Block of Receive Channels	RPABLK Bit in MCR	RPBBLK Bit in MCR	RCERE0 Bits
A	Block 0: channels 0 through 15	0	–	RCE0-RCE15
A	Block 2: channels 32 through 47	1h	–	RCE0-RCE15
A	Block 4: channels 64 through 79	2h	–	RCE0-RCE15
A	Block 6: channels 96 through 111	3h	–	RCE0-RCE15
B	Block 1: channels 16 through 31	–	0	RCE16-RCE31
B	Block 3: channels 48 through 63	–	1h	RCE16-RCE31
B	Block 5: channels 80 through 95	–	2h	RCE16-RCE31
B	Block 7: channels 112 through 127	–	3h	RCE16-RCE31

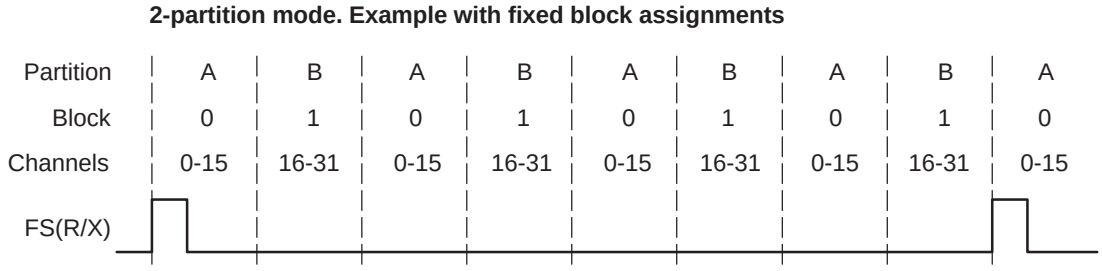
Table 24-14. Transmit Channel Assignment and Control When Two Transmit Partitions are Used

Transmit Partition	Assigned Block of Transmit Channels	XPABLK Bit in MCR	XPBBLK Bit in MCR	XCERE0 Bits
A	Block 0: channels 0 through 15	0	–	XCE0-XCE15
A	Block 2: channels 32 through 47	1h	–	XCE0-XCE15
A	Block 4: channels 64 through 79	2h	–	XCE0-XCE15
A	Block 6: channels 96 through 111	3h	–	XCE0-XCE15
B	Block 1: channels 16 through 31	–	0	XCE16-XCE31
B	Block 3: channels 48 through 63	–	1h	XCE16-XCE31
B	Block 5: channels 80 through 95	–	2h	XCE16-XCE31
B	Block 7: channels 112 through 127	–	3h	XCE16-XCE31

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 24-38 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-sync 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 24-38. Alternating Between the Channels of Partition A and the Channels of Partition B



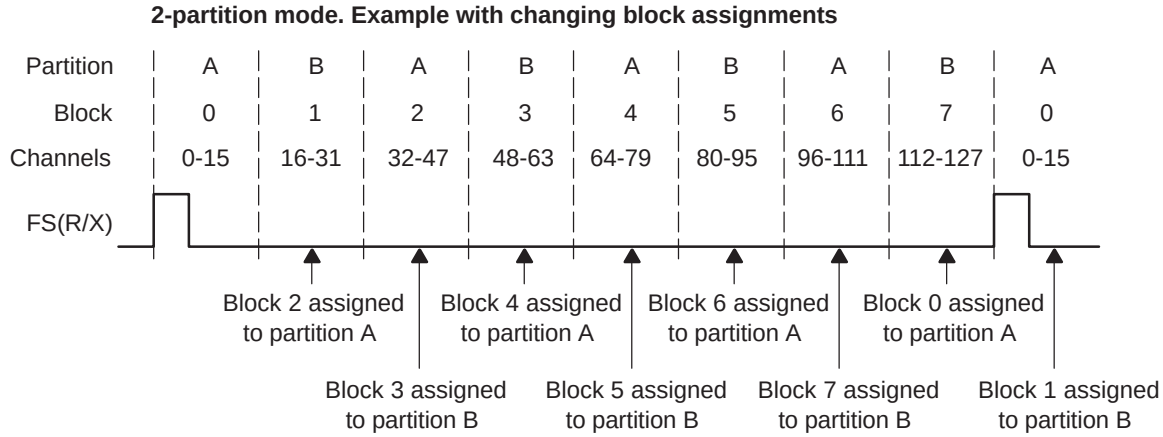
24.2.9.5.2 Reassigning Blocks During Reception/Transmission

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, it's the 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 cannot modify (R/X)PABLK to assign different channels to partition A, and you cannot modify (R/X)CERE0 to change the channel configuration for partition A. Several features of the McBSP helps to 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.

Figure 24-39 shows an example of reassigning channels throughout a data transfer. In response to a frame-sync 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 24-39. Reassigning Channel Blocks Throughout a McBSP Data Transfer



Copyright © 2013–2016, Texas Instruments Incorporated

24.2.9.6 Using Eight Partitions

For multichannel selection operation in the receiver and/or the transmitter, you can use eight partitions or two partitions. If you choose the 8-partition mode (RMCME = 1 for reception, XMCME = 1 for transmission), McBSP partitions are activated in the following order: A, B, C, D, E, F, G, H. In response to a frame-sync 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-sync 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 24-15 and Table 24-16. These assignments cannot be changed. Table 24-15 and Table 24-16 also show the registers used to control the channels in the partitions.

Table 24-15. Receive Channel Assignment and Control When Eight Receive Partitions are Used

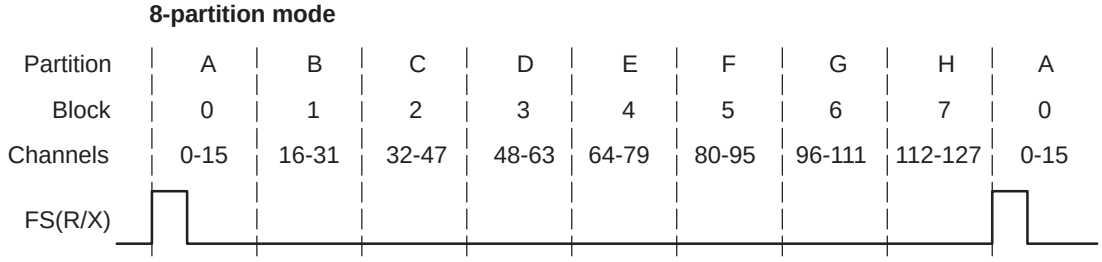
Receive Partition	Assigned Block of Receive Channels	Register Used For Channel Control
A	Block 0: channels 0 through 15	RCERE0
B	Block 1: channels 16 through 31	RCERE0
C	Block 2: channels 32 through 47	RCERE1
D	Block 3: channels 48 through 63	RCERE1
E	Block 4: channels 64 through 79	RCERE2
F	Block 5: channels 80 through 95	RCERE2
G	Block 6: channels 96 through 111	RCERE3
H	Block 7: channels 112 through 127	RCERE3

Table 24-16. 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	XCERE0
B	Block 1: channels 16 through 31	XCERE0
C	Block 2: channels 32 through 47	XCERE1
D	Block 3: channels 48 through 63	XCERE1
E	Block 4: channels 64 through 79	XCERE2
F	Block 5: channels 80 through 95	XCERE2
G	Block 6: channels 96 through 111	XCERE3
H	Block 7: channels 112 through 127	XCERE3

Figure 24-40 shows an example of the McBSP using the 8-partition mode. In response to a frame-sync 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 24-40. McBSP Data Transfer in the 8-Partition Mode



Copyright © 2013–2016, Texas Instruments Incorporated

24.2.9.7 Receive Multichannel Selection Mode

The RMCM bit in the multichannel control register (MCR) 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 enhanced receive channel enable register (RCERE*n*). The way channels are assigned to the RCERE*n* depends on the number of receive channel partitions (2 or 8), as defined by the RMCME bit in MCR.
- If a receive channel is disabled, any bits received in that channel are passed only as far as the receive buffer register (RBR). The receiver does not copy the content of the RBR to the DRR, 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 = 0), 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.

24.2.9.8 Transmit Multichannel Selection Mode

The XMCM bit in the multichannel control register (MCR) determines whether all channels or only selected channels are enabled and unmasked for transmission. The McBSP has three transmit multichannel selection modes (XMCM = 1, XMCM = 2h, and XMCM = 3h), which are described in Table 24-17.

Table 24-17. Selecting a Transmit Multichannel Selection Mode With the XMCM Bits

XMCM Bit in MCR	Transmit Multichannel Selection Mode
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 enhanced transmit channel enable register (XCERE*n*). If enabled, a channel in this mode is also unmasked.
	The XMCME bit in MCR determines whether 32 channels or 128 channels are selectable in XCERE*n*.
2h	All channels are enabled, but they are masked unless they are selected in the appropriate enhanced transmit channel enable register (XCERE*n*).
	The XMCME bit in MCR determines whether 32 channels or 128 channels are selectable in XCERE*n*.
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 enhanced receive channel enable register (RCERE*n*). Once enabled, they are masked unless they are also selected in the appropriate enhanced transmit channel enable register (XCERE*n*).
	The XMCME bit in MCR determines whether 32 channels or 128 channels are selectable in RCERE*n* and XCERE*n*.

As an example of how the McBSP behaves in a transmit multichannel selection mode, suppose that XMCM = 1 (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. TheMcBSP:

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.

Copyright © 2013–2016, Texas Instruments Incorporated

24.2.9.8.1 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 (DXR) to the transmit shift register (XSR).
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 in SPCR is not set. Therefore, no DMA synchronization event (XEVT) is generated, and if the transmit interrupt mode depends on XRDY (XINTM = 0 in SPCR), no interrupt is generated. The XEMPTY bit in SPCR is not affected.
Unmasked channel	A channel that is not masked. Data in the XSR is shifted out on the DX pin.

24.2.9.8.2 Activity on McBSP Pins for Different Values of XMCM

Figure 24-41 shows the activity on the McBSP pins for the various XMCM values. In all cases, the transmit frame is configured as follows:

- In transmit control register (XCR):
  - XPHASE = 0: Single-phase frame (required for multichannel selection modes)
  - XFRLEN1 = 3h: 4 words per frame
  - XWDLEN1 = 0: 8 bits per word
- In multichannel control register (MCR):
  - XMCME = 0: 2-partition mode (only partitions A and B used)

In the case where XMCM = 3h, 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 Figure 24-41, 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.

24.2.9.9 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 the RINTM bit in the serial port control register (SPCR) is set to 1. In any of the transmit multichannel selection modes, a transmit interrupt (XINT) request is generated at the end of each block transfer if the XINTM bit in SPCR is set to 1. When RINTM/XINTM = 1, no interrupt is generated unless a multichannel selection mode is on.

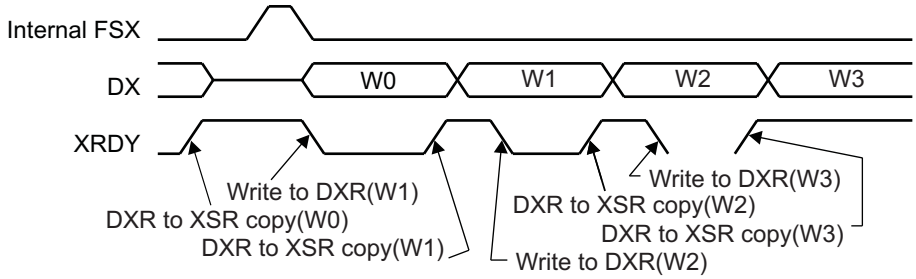
These interrupt pulses are active high and last for two McBSP internal input clock cycles.

This type of interrupt is especially helpful if you are using the two-partition mode and you want to know when you can assign a different block of channels to partition A or B.

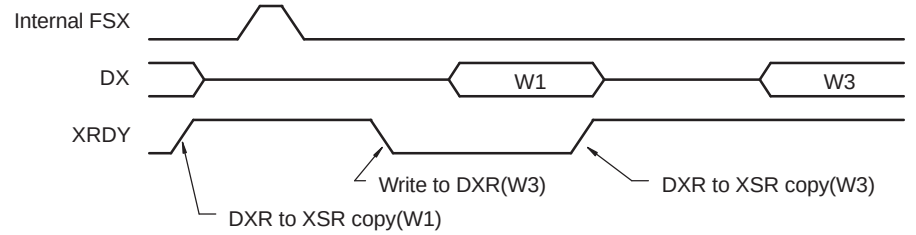
Copyright © 2013–2016, Texas Instruments Incorporated

Figure 24-41. Activity on McBSP Pins for the Possible Values of XMCM

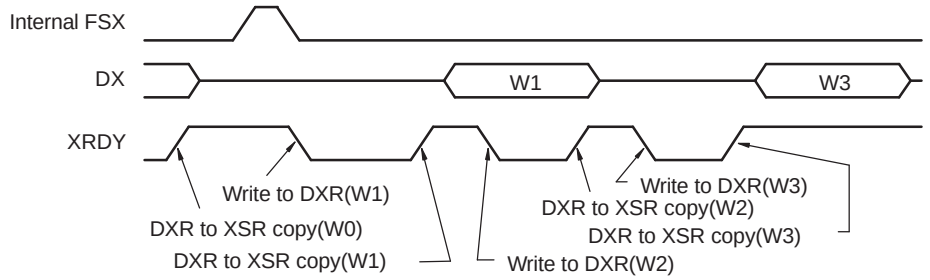
(a) XMCM = 0: All channels enabled and unmasked



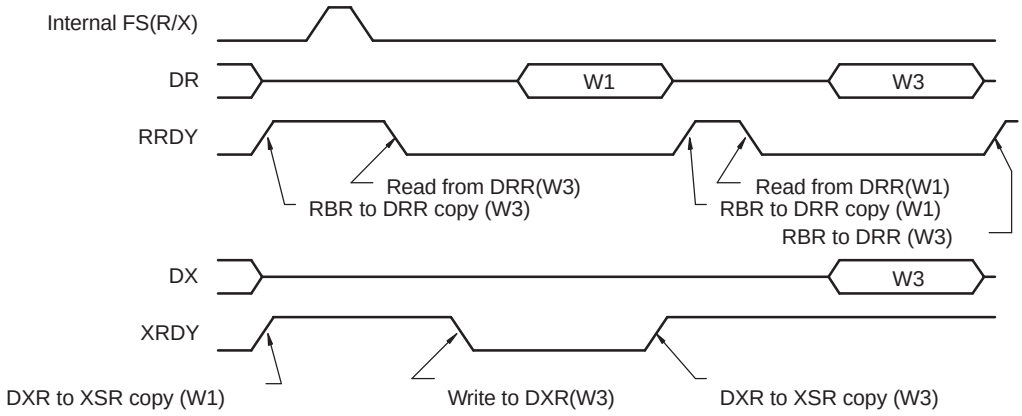
(b) XMCM = 1h, XPABLK = 0, XCERE0 = 0000 000Ah: Only channels 1 and 3 enabled and unmasked



(c) XMCM = 2h, XPABLK = 0, XCERE0 = 0000 000Ah: All channels enabled, only 1 and 3 unmasked



(d) XMCM = 3h, RPABLK = 0, XPABLK = x, RCERE0 = 0000 0008h, XCERE0 = 0000 000Ah: Receive channels: 1 and 3 enabled; transmit channels: 1 and 3 enabled, but only 3 unmasked



Copyright © 2013–2016, Texas Instruments Incorporated

24.2.10 SPI Operation Using the Clock Stop Mode

The McBSP on this device does not support the SPI protocol.

24.2.11 Resetting the Serial Port: RRST, XRST, GRST, and RESET

Device reset or McBSP reset: When the McBSP is reset by device reset or McBSP reset, the state machine is reset to its initial state. All counters and status bits are reset. This includes the receive status bits RFULL, RRDY, and RSYNCERR, and the transmit status bits XEMPTY, XRDY, and XSYNCERR in the serial port control register (SPCR).

The serial port can be reset in the following two ways:

- Device reset (RESET pin is low) places the receiver, the transmitter, and the sample rate generator in reset. When the device reset is removed (RESET = 1), FRST = GRST = RRST = XRST = 0 in SPCR, keeping the entire serial port in the reset state.
- The serial port transmitter and receiver can be independently reset by the XRST and RRST bits in SPCR. The sample rate generator is reset by the GRST bit in SPCR.

Table 24-18 shows the state of the McBSP pins when the serial port is reset by these methods.

Table 24-18. Reset State of McBSP Pins

Pin	Direction	Device Reset ($\overline{\text{RESET}}$ = 0)	McBSP Reset
			Receiver Reset (RRST = 0 and GRST = 1)
DR	I	Input	Input
CLKR	I/O/Z	Input	Known state if input; CLKR if output
FSR	I/O/Z	Input	Known state if input; FSRP(inactive state) if output
CLKS	I	Input	Input
			Transmitter Reset (XRST = 0 and GRST = 1)
DX	O/Z	High impedance	High impedance
CLKX	I/O/Z	Input	Known state if input; CLKX if output
FSX	I/O/Z	Input	Known state if input; FSXP(inactive state) if output
CLKS	I	Input	Input

24.2.11.1 Software Reset Considerations

McBSP reset: When the receiver and transmitter reset bits, RRST and XRST in SPCR, are written with 0, the respective portions of the McBSP are reset and activity in the corresponding section stops. All input-only pins, such as DR, and all other pins that are configured as inputs are in a known state. FS(R/X) is driven to its inactive state (same as its polarity bit, FS(R/X)P) if it is an output. If CLK(R/X) are programmed as outputs, they are driven by CLKG, provided that GRST = 1. The DX pin is in the high-impedance state when the transmitter is reset. During normal operation, the sample rate generator can be reset by writing a 0 to GRST. The sample rate generator should only be reset when not being used by the transmitter or the receiver. In this case, the internal sample rate generator clock CLKG, and its frame sync signal (FSG) is driven inactive (low). When the sample rate generator is not in the reset state (GRST = 1), FSR and FSX 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 FRST = 1 and frame sync is driven by FSG.

Sample-rate generator reset: As mentioned previously, the sample rate generator is reset when the device is reset or when its reset bit, GRST in SPCR, is written with 0.

Emulator software reset: In the event of an emulator software reset initiated from the CPU, the McBSP register values are reset to their default values.

Copyright © 2013–2016, Texas Instruments Incorporated

24.2.11.2 Hardware Reset Considerations

When the McBSP is reset due to device reset, the entire serial port (including the transmitter, receiver, and the sample rate generator) is reset. All input-only pins and 3-state pins should be in a known state. The output-only pin, DX, is in the high-impedance state. When the device is pulled out of reset, the serial port remains in the reset condition (RRST = XRST = FRST = GRST = 0 in SPCR).

24.2.12 McBSP Initialization Procedure

The McBSP initialization procedure varies depending on the specific system setup. Section 24.2.12.1 provides a general initialization sequence.

The transmitter and the receiver of the McBSP can operate independently from each other. Therefore, they can be placed in or taken out of reset individually by modifying only the desired bit in the registers without disrupting the other portion. The steps in the following sections discuss the initialization procedure for taking both the transmitter and the receiver out of reset. To initialize only one portion, configure only the portion desired.

The McBSP internal sample rate generator and internal frame sync generator are shared between the transmitter and the receiver. Table 24-19 and Table 24-20 describe their usage base upon the clock and frame sync configurations of the receiver and transmitter, respectively.

Table 24-19. Receiver Clock and Frame Configurations

CLKR Source	FSR Source	Comment on Configuration
Internal	Internal	The McBSP internal sample rate generator and internal frame sync generator are used by the receiver.
External	Internal	The McBSP internal sample rate generator and internal frame sync generator are used by the receiver.
Internal	External	The McBSP internal sample rate generator is used but the internal frame sync generator is not used by the receiver.
External	External	The McBSP internal sample rate generator and internal frame sync generator are not used by the receiver.

Table 24-20. Transmitter Clock and Frame Configurations

CLKX Source	FSX Source	Comment on Configuration
Internal	Internal	The McBSP internal sample rate generator is used by the transmitter. The transmitter can generate frame sync FSX in one of two ways. First, it can generate FSX by using the internal frame sync generator (FSGM = 1). Alternatively, it can generate FSX upon each DXR-to-XSR copy (FSGM = 0). In this case, the internal frame sync generator can be kept in reset (FRST = 0) if it is not used by the receiver. You can follow the general initialization sequence in Section 24.2.12.1.
External	Internal	The McBSP internal sample rate generator and internal frame sync generator are not used by the transmitter. This configuration is only valid with FSGM = 0 where the McBSP transmitter generates FSX upon each DXR-to-XSR copy. You can follow the general initialization sequence in Section 24.2.12.1.
Internal	External	The McBSP internal sample rate generator is used by the transmitter but the internal frame sync generator is not.
External	External	The McBSP internal sample rate generator and internal frame sync generator are not used by the transmitter.

Copyright © 2013–2016, Texas Instruments Incorporated

24.2.12.1 General Initialization Procedure

This section provides the general initialization procedure.

1. With the McBSP still in reset (Power and Sleep Controller (PSC) in the default state):
   (a) Program the PSC registers in the System Module to put the McBSP in the enable state (see the *Power and Sleep Controller (PSC)* chapter.
   (b) Perform the necessary device pin multiplexing setup (see your device-specific data manual).
2. Ensure that no portion of the McBSP is using the internal sample rate generator signal CLKG and the internal frame sync generator signal FSG (GRST = FRST = 0 in SPCR). The respective portion of the McBSP needs to be in reset (XRST = 0 and/or RRST = 0 in SPCR).
3. Program the control registers as required. Ensure the internal sample rate generator and the internal frame sync generator are still in reset (GRST = FRST = 0). Also ensure the respective portion of the McBSP is still in reset in this step (XRST = 0 and/or RRST = 0).
4. Wait for proper internal synchronization. If the external device provides the bit clock, wait for two CLKR or CLKX cycles. If the McBSP generates the bit clock as a clock master, wait for two CLKSRG cycles. In this case, the clock source to the sample rate generator (CLKSRG) is selected by the CLKSM bit in SRGR and the SCLKME bit in PCR.
5. Skip this step if the bit clock is provided by the external device. This step only applies if the McBSP is the bit clock master and the internal sample rate generator is used.
   (a) Start the sample rate generator by setting the GRST bit to 1. Wait two CLKG bit clocks for synchronization. CLKG is the output of the sample rate generator.
   (b) On the next rising edge of CLKSRG, CLKG transitions to 1 and starts clocking with a frequency equal to 1/(CLKGDV + 1) of the sample rate generator source clock CLKSRG.
6. Skip this step if the transmitter is not used. If the transmitter is used, a transmit sync error (XSYNCERR) may occur when it is enabled for the first time after device reset. The purpose of this step is to clear any potential XSYNCERR that occurs on the transmitter at this time:
   (a) Set the XRST bit to 1 to enable the transmitter.
   (b) Wait for any unexpected frame sync error to occur. If the external device provides the bit clock, wait for two CLKR or CLKX cycles. If the McBSP generates the bit clock as a clock master, wait for two CLKG cycles. The unexpected frame sync error (XSYNCERR), if any, occurs within this time period.
   (c) Disable the transmitter (XRST = 0). This clears any outstanding XSYNCERR.
7. Setup data acquisition as required:
   (a) If the EDMA is used to service the McBSP, setup data acquisition as desired and start the EDMA in this step, before the McBSP is taken out of reset.
   (b) If CPU interrupt is used to service the McBSP, enable the transmit and/or receive interrupt as required.
   (c) If CPU polling is used to service the McBSP, no action is required in this step.
8. Set the XRST bit and/or the RRST bit to 1 to enable the corresponding section of the McBSP The McBSP is now ready to transmit and/or receive.
   (a) If the EDMA is used to service the McBSP, it services the McBSP automatically upon receiving the XEVT and/or REVT.
   (b) If CPU interrupt is used to service the McBSP, the interrupt service routine is automatically entered upon receiving the XINT and/or RINT.
   (c) If CPU polling is used to service the McBSP, it can do so now by polling the XRDY and/or RRDY bit.
9. If the internal frame sync generator is used (FSGM = 1), proceed to the additional steps to turn on the internal frame sync generator. Initialization is complete if any one of the following is true:
   (a) The external device generates frame sync FSX and/or FSR. The McBSP is now ready to transmit and/or receive upon receiving external frame sync.
   (b) The McBSP generates transmit frame sync FSX upon each DXR-to-XSR copy. The internal frame sync generator is not used (FSGM = 0).

Copyright © 2013–2016, Texas Instruments Incorporated

The following additional steps to turn on the internal frame sync generator apply only if FSGM = 1:

10. Skip this step if the transmitter is not used. If the transmitter is used, ensure that DXR is serviced before you start the internal frame sync generator. You can do so by checking XEMPTY = 1 (XSR is not empty) in SPCR.
11. Set the FRST bit to 1 to start the internal frame sync generator. The internal frame sync signal FSG is generated on a CLKG active edge after 7 to 8 CLKG clocks have elapsed.

24.2.12.2 Special Case: External Device is the Transmit Frame Master

Care must be taken if the transmitter expects a frame sync from an external device. After the transmitter comes out of reset (XRST = 1), it waits for a frame sync from the external device. If the first frame sync arrives very shortly after the transmitter is enabled, the CPU or EDMA controller may not have a chance to service the data transmit register (DXR). In this case, the transmitter shifts out the default data in the transmit shift register (XSR) instead of the desired value, which has not yet arrived in DXR. This causes problems in some applications, as the first data element in the frame is invalid. The data stream appears element-shifted (the first data word may appear in the second channel instead of the first).

To ensure proper operation when the external device is the frame master, you must assure that DXR is already serviced with the first word when a frame sync occurs. To do so, you can keep the transmitter in reset until the first frame sync is detected. Software is setup such that it will only take the transmitter out of reset (XRST = 1) promptly after detecting the first frame sync. This assures that the transmitter does not begin data transfers at the data pin during the first frame sync period. This also provides almost an entire frame period for the CPU to service DXR with the first word before the second frame sync occurs. The transmitter only begins data transfers upon receiving the second frame sync. At this point, DXR is already serviced with the first word.

24.2.12.2.1 How to Detect First Frame Sync

Although the McBSP is capable of generating an interrupt to the CPU upon the detection of frame synchronization (XINTM = 2h and/or RINTM = 2h in the serial port control register (SPCR)), the McBSP requires the associated portion (receiver/transmitter) of the McBSP to be out of reset in order for the interrupt to be generated. Therefore, instead of directly using the McBSP interrupt to detect the first frame sync, you can use the GPIO peripheral. This can be achieved by connecting the frame sync signal to a GPIO pin. Software can either poll the GPIO pin to detect the first frame sync or program the GPIO peripheral to generate an interrupt to the CPU upon detecting the first frame sync edge. For more information on the GPIO peripheral, see the *General-Purpose Input/Output (GPIO)* chapter.

The following are some recommended GPIO pin(s) on the device that you can use to detect the first McBSP external frame sync:

- **GPIO pin located near the McBSP pins.** Connect the external frame sync to both the McBSP FSX/FSR pin(s) and the dedicated GPIO pin.
- **GPIO pin multiplexed with the McBSP FSX signal.** Note that on the device, the GPIO pins (of the GPIO peripheral) are multiplexed with the McBSP pins. Software can program the device's pin multiplexing register (PINMUX) to default these pins to the GPIO function, and only switch them to the McBSP function upon detecting the first frame sync. This method is only recommended if the external device is both the frame sync and clock master; that is, the external device drives both the FSX and CLKX signals. This method is not recommended if the McBSP is the clock master (driving CLKX and/or CLKR), as the “on-the-fly” pin multiplexed switching can cause a glitch on the CLKX/CLKR pin. For more details on pin multiplexing, see the device-specific data manual.

24.2.12.2.2 Initialization Procedure When External Device is Frame Sync Master

The initialization procedure assumes the following:

- Using a GPIO pin multiplexed with the McBSP FSX signal. If a dedicated GPIO pin is used instead, skip step 1 and step 8b.
- Software polls the GPIO pin to detect the first frame sync. If the GPIO interrupt is used instead to detect the first frame sync, step 8 can be performed within an interrupt service routine (ISR).

Copyright © 2013–2016, Texas Instruments Incorporated

1. The GPIO and McBSP signals are multiplexed together on the device. Start by programming the pin multiplexing register (PINMUX) to select the GPIO function on the GPIO/McBSP multliplexed pins. Program the GPIO peripheral so that these pins function as GPIO inputs.
2. Ensure that no portion of the McBSP is using the internal sample rate generator signal CLKG and the internal frame sync generator signal FSG (GRST = FRST = 0 in SPCR). The respective portion of the McBSP needs to be in reset (XRST = 0 and/or RRST = 0 in SPCR).
3. Program the sample rate generator register (SRGR) and other control registers as required. Ensure the internal sample rate generator and the internal frame sync generator are still in reset (GRST = FRST = 0 in SPCR). Also ensure the respective portion of the McBSP is still in reset in this step (XRST = 0 and/or RRST = 0 in SPCR).
4. Wait for proper McBSP internal synchronization:
   (a) If the external device provides the bit clock, wait for two CLKR or CLKX cycles. Skip step 5.
   (b) If the McBSP generates the bit clock as a clock master, wait for two CLKSRG cycles. In this case, the clock source to the sample rate generator (CLKSRG) is selected by the CLKSM bit in SRGR.
5. Skip this step if the bit clock is provided by the external device. This step only applies if the McBSP is the bit clock master and the internal sample rate generator is used.
   (a) Start the sample rate generator by setting the GRST bit in SPCR to 1. Wait two CLKG bit clocks for synchronization. CLKG is the output of the sample rate generator.
   (b) On the next rising edge of CLKSRG, CLKG transitions to 1 and starts clocking with a frequency equal to 1/(CLKGDV + 1) of the sample rate generator source clock CLKSRG.
6. A transmit sync error (XSYNCERR) may occur when it is enabled for the first time after device reset. The purpose of this step is to clear any potential XSYNCERR that occurs on the transmitter at this time:
   (a) Set the XRST bit in SPCR to 1 to enable the transmitter.
   (b) Wait for any unexpected frame sync error to occur. If the external device provides the bit clock, wait for two CLKR or CLKX cycles. If the McBSP generates the bit clock as a clock master, wait for two CLKG cycles. The unexpected frame sync error (XSYNCERR), if any, occurs within this time period.
   (c) Disable the transmitter (XRST = 0). This clears any outstanding XSYNCERR.
7. Setup data acquisition as required:
   (a) If the EDMA controller is used to service the McBSP, setup data acquisition as desired and start the EDMA controller in this step, before the McBSP is taken out of reset.
   (b) If the CPU interrupt is used to service the McBSP, no action is required in this step.
   (c) If CPU polling is used to service the McBSP, no action is required in this step.
8. Poll the GPIO pin (through reading the appropriate registers in the GPIO peripheral) to detect the first transmit frame sync from the external device. Upon detection of the first frame sync, perform the following in this order:
   (a) Set the XRST bit and/or the RRST bit to 1 to enable the respective portion of the McBSP. The McBSP is now ready to transmit and/or receive.
   (b) Program PINMUX to switch the GPIO/McBSP multiplexed pins to the McBSP function.
9. Service the McBSP:
   (a) If CPU polling is used to service the McBSP in normal operations, it can do so upon exit from the ISR.
   (b) If the CPU interrupt is used to service the McBSP in normal operations, upon XRDY interrupt service routine is entered. The ISR should be setup to verify that XRDY = 1 and service the McBSP accordingly.
   (c) If the EDMA controller is used to service the McBSP in normal operations, it services the McBSP automatically upon receiving the XEVT and/or REVT.
10. Upon detection of the second frame sync, DXR is already serviced and the transmitter is ready to transmit the valid data. The receiver is also serviced properly by the CPU.

Copyright © 2013–2016, Texas Instruments Incorporated

24.2.13 Interrupt Support

The McBSP can send both receive and transmit interrupts.

24.2.13.1 Interrupt Events and Requests

The RRDY and XRDY bits in the serial port control register (SPCR) indicate the ready state of the McBSP receiver and transmitter, respectively. Writes and reads from the serial port can be synchronized by any of the following methods:

- Polling RRDY and XRDY bits in SPCR
- Using the events sent to the EDMA controller (REVT and XEVT)
- Using the interrupts to the CPU (RINT and XINT) that the events generate

Reading DRR and writing to DXR affects RRDY and XRDY, respectively.

24.2.13.1.1 Interrupt Events: RINT and XINT

The receive interrupt (RINT) and transmit interrupt (XINT) signals inform the CPU of changes to the serial port status. Three options exist for configuring these interrupts. These options are set by the receive/transmit interrupt mode bits (RINTM and XINTM) in SPCR. The possible values of the mode, and the configurations they represent, are:

- (R/X)INTM = 00b: Interrupt on every serial element by tracking the (R/X)RDY bits in SPCR.
- (R/X)INTM = 01b: Interrupt at the end of a subframe (16 elements or less) within a frame. See Section 24.2.9.9 for more details.
- (R/X)INTM = 10b: Interrupt on detection of frame synchronization pulses. The associated portion (receiver/transmitter) of the McBSP must be out of reset.
- (R/X)INTM = 11b: Interrupt on frame synchronization error. Note that if any of the other interrupt modes are selected, (R/X)SYNCERR may be read when servicing the interrupts to detect this condition. See Section 24.2.7.5.2 and Section 24.2.7.5.5 for more details on synchronization error.

24.2.13.1.2 Receive Ready Status: RINT and RRDY

RRDY = 1 indicates that the RBR contents have been copied to DRR and that the data can now be read by either the CPU or the EDMA controller. Once that data has been read by either the CPU, RRDY is cleared to 0. Also, at device reset or serial port receiver reset (RRST = 0), the RRDY bit is cleared to 0 to indicate that no data has been received and loaded into DRR. RRDY directly drives the McBSP receive interrupt (RINT) to the CPU if RINTM = 00b (default value) in SPCR.

24.2.13.1.3 Transmit Ready Status: XINT and XRDY

XRDY = 1 indicates that the DXR contents have been copied to XSR and that DXR is ready to be loaded with a new data word. When the transmitter transitions from reset to non-reset (XRST transitions from 0 to 1), XRDY also transitions from 0 to 1 indicating that DXR is ready for new data. Once new data is loaded by the CPU, the XRDY bit is cleared to 0. However, once this data is copied from DXR to XSR, the XRDY bit transitions again from 0 to 1. The CPU can write to DXR although XSR has not yet been shifted out on DX. XRDY directly drives the McBSP transmit interrupt (XINT) to the CPU if XINTM = 00b (default value) in SPCR.

NOTE: If the polling method is used to service the transmitter, the CPU should wait for one McBSP bit clock (CLKX) before polling again to write the next element in DXR. This is because XRDY transitions occur based on bit clock and not CPU clock. The CPU clock is much faster and can cause false XRDY status, leading to data errors due to overwrites.

Copyright © 2013–2016, Texas Instruments Incorporated

24.2.13.2 Interrupt Multiplexing

The RINT and XINT interrupts generated by the McBSP peripheral to the CPU are multiplexed with other interrupt sources. Refer to the device-specific data manual to determine how pin multiplexing affects the McBSP.

24.2.14 EDMA Event Support

24.2.14.1 Receive Ready Status: REVT and RRDY

RRDY = 1 in the serial port control register (SPCR) indicates that the RBR contents have been copied to DRR and that the data can now be read by the EDMA controller. Once that data has been read by the EDMA controller, RRDY is cleared to 0. Also, at device reset or serial port receiver reset (RRST = 0 in SPCR), the RRDY bit is cleared to 0 to indicate that no data has been received and loaded into DRR. RRDY directly drives the McBSP receive event to the EDMA controller (via REVT).

For detailed information on using the EDMA to read or write to the McBSP, see the *Enhanced Direct Memory Access (EDMA3) Controller* chapter.

24.2.14.2 Transmit Ready Status: XEVT and XRDY

XRDY = 1 in the serial port control register (SPCR) indicates that the DXR contents have been copied to XSR and that DXR is ready to be loaded with a new data word. When the transmitter transitions from reset to non-reset (XRST transitions from 0 to 1 in SPCR), XRDY also transitions from 0 to 1 indicating that DXR is ready for new data. Once new data is loaded by the EDMA controller, the XRDY bit is cleared to 0. However, once this data is copied from DXR to XSR, the XRDY bit transitions again from 0 to 1. The EDMA controller can write to DXR although XSR has not yet been shifted out on DX. XRDY directly drives the transmit synchronization event to the EDMA controller (via XEVT).

For detailed information on using the EDMA to read or write to the McBSP, see the *Enhanced Direct Memory Access (EDMA3) Controller* chapter.

NOTE: If the polling method is used to service the transmitter, the CPU should wait for one McBSP bit clock (CLKX) before polling again to write the next element in DXR. This is because XRDY transitions occur based on bit clock and not CPU clock. The CPU clock is much faster and can cause false XRDY status, leading to data errors due to overwrites.

Copyright © 2013–2016, Texas Instruments Incorporated

24.2.15 Power Management

The McBSP can be placed in reduced power modes to conserve power during periods of low activity. The power management of the peripheral is controlled by the processor Power and Sleep Controller (PSC). The PSC acts as a master controller for power management for all of the peripherals on the device.

In order for the McBSP to be placed in power-down mode by the PSC, ensure that the XRDY and RRDY flags in the serial port control register (SPCR) are cleared by performing the following steps:

1. Place the McBSP in reset by clearing the XRST, RRST, FRST, and GRST bits to 0 in SPCR.

   If EDMA is being used to service the transmitter and/or the receiver, disable the associated EDMA channels.

   For detailed information on using the EDMA to read or write to the McBSP, see the *Enhanced Direct Memory Access (EDMA3) Controller* chapter.
2. Switch the McBSP clocks and frames to internal clock source:

   (a) Set the CLKSM and FSGM bits to 1 in the sample rate generator register (SRGR).

   (b) Set the CLKXM, CLKRM, FSXM, and FSRM bits to 1 in the pin control register (PCR).

   (c) Clear the SCLKME bit to 0 in PCR.
3. Bring the McBSP out of reset by setting the XRST, RRST, and GRST bits to 1 in SPCR.
4. Wait for two CLKSRG cycles for proper internal synchronization.
5. Write a dummy data value to the data transmit register (DXR) in order to clear the first XRDY flag.
6. Wait for at least one McBSP bit clock, since once the first dummy data value is internally copied from DXR to XSR, the XRDY flag transitions again from 0 to 1.
7. Write a second dummy data value to DXR in order to clear the second XRDY flag.
8. Check the RRDY flag in SPCR and if set to 1, read the data receive register (DRR) and discard the data to clear the RRDY flag.
9. Place the McBSP in power-down mode by issuing the proper PSC commands. For detailed information on power management procedures using the PSC, see the *Power and Sleep Controller (PSC)* chapter.

NOTE: After waking up the McBSP from a power-down mode using the proper PSC commands, remember to reconfigure the SPCR, SRGR, and PCR registers to the clock and frame combination that they were in before entering the power-down sequence and discard the two dummy data values that were used to clear the XRDY flags. If EDMA is used, re-enable the corresponding EDMA channels.

24.2.16 Emulation Considerations

The FREE and SOFT bits are special emulation bits in the serial port control register (SPCR) that determine the state of the McBSP when an emulation suspend event occurs in the emulator. An emulation suspend event corresponds to any type of emulator access to the CPU, such as a hardware or software breakpoint, a probe point, or a printf instruction.

Table 24-21 shows the effects of the FREE and SOFT bits on the response of the McBSP to emulation suspend events.

Table 24-21. McBSP Emulation Modes Selectable With the FREE and SOFT Bits of SPCR

FREE Bit in SPCR	SOFT Bit in SPCR	McBSP Emulation Mode
0	0	Immediate stop mode (reset condition). The transmitter and receiver stop immediately in response to an emulation suspend event.
0	1	Soft stop mode. When an emulation suspend event 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 an emulation suspend event occurs.

Copyright © 2013–2016, Texas Instruments Incorporated

24.3 Registers

Table 24-22 lists the memory-mapped registers for the McBSP. See the device-specific data manual for the memory address of these registers. All other register offset addresses not listed in Table 24-22 should be considered as reserved locations and the register contents should not be modified.

The McBSP control registers are accessible by the CPU. You should halt the McBSP before making changes to the serial port control register (SPCR), receive control register (RCR), transmit control register (XCR), and pin control register (PCR). Changes made to these registers without halting the McBSP could result in an undefined state.

Table 24-22. McBSP Registers

Offset	Acronym	Register Name	Section
-	RBR[1]	Receive buffer register	—
-	RSR[1]	Receive shift register	—
-	XSR[1]	Transmit shift register	—
0h	DRR[2] [3]	Data receive register	Section 24.3.1
4h	DXR[3]	Data transmit register	Section 24.3.2
8h	SPCR	Serial port control register	Section 24.3.3
Ch	RCR	Receive control register	Section 24.3.4
10h	XCR	Transmit control register	Section 24.3.5
14h	SRGR	Sample rate generator register	Section 24.3.6
18h	MCR	Multichannel control register	Section 24.3.7
1Ch	RCERE0	Enhanced receive channel enable register partition A/B	Section 24.3.8
20h	XCERE0	Enhanced transmit channel enable register partition A/B	Section 24.3.9
24h	PCR	Pin control register	Section 24.3.10
28h	RCERE1	Enhanced receive channel enable register partition C/D	Section 24.3.8
2Ch	XCERE1	Enhanced transmit channel enable register partition C/D	Section 24.3.9
30h	RCERE2	Enhanced receive channel enable register partition E/F	Section 24.3.8
34h	XCERE2	Enhanced transmit channel enable register partition E/F	Section 24.3.9
38h	RCERE3	Enhanced receive channel enable register partition G/H	Section 24.3.8
3Ch	XCERE3	Enhanced transmit channel enable register partition G/H	Section 24.3.9
0h	BFIFOREV[4]	BFIFO Revision Identification Register	Section 24.3.11
10h	WFIFOCTL[4]	Write FIFO Control Register	Section 24.3.12
14h	WFIFOSTS[4]	Write FIFO Status Register	Section 24.3.13
18h	RFIFOCTL[4]	Read FIFO Control Register	Section 24.3.14
1Ch	RFIFOSTS[4]	Read FIFO Status Register	Section 24.3.15

(1) The RBR, RSR, and XSR are not directly accessible via the CPUs or the EDMA controller.
(2) The CPUs and EDMA controller can only read this register; they cannot write to it.
(3) The DRR and DXR are accessible via the CPUs or the EDMA controller.
(4) The McBSP Buffer FIFO (BFIFO) has a different memory-map (see your device-specific data manual) than the McBSP memory-mapped registers (MMRs); hence, the BFIFO is accessible by way of a different Configuration Bus.

Copyright © 2013–2016, Texas Instruments Incorporated

24.3.1 Data Receive Register (DRR)

The data receive register (DRR) contains the value to be written to the data bus. The DRR is shown in Figure 24-42 and described in Table 24-23.

See the device-specific data manual for the memory address of these registers. Both the CPUs and the EDMA can access DRR in all the memory-mapped locations. An access to *any* EDMA bus location is equivalent to an access to DRR of the corresponding McBSP.

Figure 24-42. Data Receive Register (DRR)

31	16
DR	
R-0	

15	0
DR	
R-0	

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

Table 24-23. Data Receive Register (DRR) Field Descriptions

Bit	Field	Value	Description
31-0	DR	0-FFFF FFFFh	Data receive register value to be written to the data bus.

24.3.2 Data Transmit Register (DXR)

The data transmit register (DXR) contains the value to be loaded into the data transmit shift register (XSR). The DXR is shown in Figure 24-43 and described in Table 24-24.

See the device-specific data manual for the memory address of these registers. DXR is accessible via the peripheral bus and via the EDMA bus. Both the CPUs and the EDMA can access DXR in all the memory-mapped locations.

Figure 24-43. Data Transmit Register (DXR)

31	16
DX	
R/W-0	

15	0
DX	
R/W-0	

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

Table 24-24. Data Transmit Register (DXR) Field Descriptions

Bit	Field	Value	Description
31-0	DX	0-FFFF FFFFh	Data transmit register value to be loaded into the data transmit shift register (XSR).

Copyright © 2013–2016, Texas Instruments Incorporated

24.3.3 Serial Port Control Register (SPCR)

The serial port is configured via the serial port control register (SPCR) and the pin control register (PCR). The SPCR contains McBSP status control bits. The SPCR is shown in Figure 24-44 and described in Table 24-25.

Figure 24-44. Serial Port Control Register (SPCR)

31						26	25	24
Reserved							FREE	SOFT
R-0							R/W-0	R/W-0
23	22	21			20	19	18	17	16
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
15	14		13	12		11	10		8
DLB	RJUST			CLKSTP			Reserved		
R/W-0	R/W-0			R-0			R-0		
7	6	5			4	3	2	1	0
DXENA	Reserved	RINTM				RSYNCERR	RFULL	RRDY	RRST
R/W-0	R-0	R/W-0				R/W-0	R-0	R-0	R/W-0

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

Table 24-25. Serial Port Control Register (SPCR) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.
25	FREE		Free-running enable mode bit. This bit is used in conjunction with SOFT bit to determine state of serial port clock during emulation halt.
		0	Free-running mode is disabled. During emulation halt, SOFT bit determines operation of McBSP.
		1	Free-running mode is enabled. During emulation halt, serial clocks continue to run.
24	SOFT		Soft bit enable mode bit. This bit is used in conjunction with FREE bit to determine state of serial port clock during emulation halt. This bit has no effect if FREE = 1.
		0	Soft mode is disabled. Serial port clock stops immediately during emulation halt, thus aborting any transmissions.
		1	Soft mode is enabled. During emulation halt, serial port clock stops after completion of current transmission.
23	FRST		Frame-sync generator reset bit.
		0	Frame-synchronization logic is reset. Frame-sync signal (FSG) is not generated by the sample-rate generator.
		1	Frame-sync signal (FSG) is generated after (FPER + 1) number of CLKG clocks; that is, all frame counters are loaded with their programmed values.
22	GRST		Sample-rate generator reset bit.
		0	Sample-rate generator is reset.
		1	Sample-rate generator is taken out of reset. CLKG is driven as per programmed value in sample-rate generator register (SRGR).
21-20	XINTM	0-3h	Transmit interrupt (XINT) mode bit.
		0	XINT is driven by XRDY (end-of-word).
		1h	Reserved
		2h	XINT is generated by a new frame synchronization.
		3h	XINT is generated by XSYNCERR.
19	XSYNCERR		Transmit synchronization error bit. Writing a 1 to XSYNCERR sets the error condition when the transmitter is enabled (XRST = 1). Thus, it is used mainly for testing purposes or if this operation is desired.
		0	No synchronization error is detected.
		1	Synchronization error is detected.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 24-25. Serial Port Control Register (SPCR) Field Descriptions (continued)

Bit	Field	Value	Description
18	XEMPTY		Transmit shift register empty bit.
		0	XSR is empty.
		1	XSR is not empty.
17	XRDY		Transmitter ready bit.
		0	Transmitter is not ready.
		1	Transmitter is ready for new data in DXR.
16	XRST		Transmitter reset bit resets or enables the transmitter.
		0	Serial port transmitter is disabled and in reset state.
		1	Serial port transmitter is enabled.
15	DLB		Digital loop back mode enable bit.
		0	Digital loop back mode is disabled.
		1	Digital loop back mode is enabled.
14-13	RJUST	0-3h	Receive sign-extension and justification mode bit.
		0	Right-justify and zero-fill MSBs in DRR.
		1h	Right-justify and sign-extend MSBs in DRR.
		2h	Left-justify and zero-fill LSBs in DRR.
		3h	Reserved
12-11	CLKSTP	0-3h	Clock stop mode bit.
		0	Clock stop mode is disabled. Normal clocking for non-SPI mode.
		1h-3h	Reserved
10-8	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.
7	DXENA		DX enabler bit. See Section 24.2.9.4 for details on the DX enabler bit.
		0	DX enabler is off.
		1	DX enabler is on.
6	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.
5-4	RINTM	0-3h	Receive interrupt (RINT) mode bit.
		0	RINT is driven by RRDY (end-of-word).
		1h	Reserved
		2h	RINT is generated by a new frame synchronization.
		3h	RINT is generated by RSYNCERR.
3	RSYNCERR		Receive synchronization error bit. Writing a 1 to RSYNCERR sets the error condition when the receiver is enabled (RRST = 1). Thus, it is used mainly for testing purposes or if this operation is desired.
		0	No synchronization error is detected.
		1	Synchronization error is detected.
2	RFULL		Receive shift register full bit.
		0	RBR is not in overrun condition.
		1	DRR is not read, RBR is full, and RSR is also full with new word.
1	RRDY		Receiver ready bit.
		0	Receiver is not ready.
		1	Receiver is ready with data to be read from DRR.
0	RRST		Receiver reset bit resets or enables the receiver.
		0	The serial port receiver is disabled and in reset state.
		1	The serial port receiver is enabled.

Copyright © 2013–2016, Texas Instruments Incorporated

24.3.4 Receive Control Register (RCR)

The receive control register (RCR) configures parameters of the receive operations. The RCR is shown in Figure 24-45 and described in Table 24-26.

Figure 24-45. Receive Control Register (RCR)

31	30 24	23 21	20 19	18	17 16
RPHASE	RFRLEN2	RWDLEN2	RCOMPAND	RFIG	RDATDLY
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15	14 8	7 5	4	3 0
Reserved	RFRLEN1	RWDLEN1	RWDREVRS	Reserved
R-0	R/W-0	R/W-0	R/W-0	R-0

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

Table 24-26. Receive Control Register (RCR) Field Descriptions

Bit	Field	Value	Description
31	RPHASE		Receive phases bit.
		0	Single-phase frame
		1	Dual-phase frame
30-24	RFRLEN2	0-7Fh	Specifies the receive frame length (number of words) in phase 2.
		0	1 word in phase 2
		1h	2 words in phase 2
		2h	3 words in phase 2
		...	...
		7Fh	128 words in phase 2
23-21	RWDLEN2	0-7h	Specifies the receive word length (number of bits) in phase 2.
		0	Receive word length is 8 bits.
		1h	Receive word length is 12 bits.
		2h	Receive word length is 16 bits.
		3h	Receive word length is 20 bits.
		4h	Receive word length is 24 bits.
		5h	Receive word length is 32 bits.
		6h-7h	Reserved
20-19	RCOMPAND	0-3h	Receive companding mode bit. Modes other than 00 are only enabled when RWDLEN1/2 bit is 000 (indicating 8-bit data).
		0	No companding, data transfer starts with MSB first.
		1h	No companding, 8-bit data transfer starts with LSB first.
		2h	Compand using μ-law for receive data.
		3h	Compand using A-law for receive data.
18	RFIG		Receive frame ignore bit.
		0	Receive frame-synchronization pulses after the first pulse restarts the transfer.
		1	Receive frame-synchronization pulses after the first pulse are ignored.
17-16	RDATDLY	0-3h	Receive data delay bit.
		0	0-bit data delay
		1h	1-bit data delay
		2h	2-bit data delay
		3h	Reserved
15	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 24-26. Receive Control Register (RCR) Field Descriptions (continued)

Bit	Field	Value	Description
14-8	RFRLEN1	0-7Fh	Specifies the receive frame length (number of words) in phase 1.
		0	1 word in phase 1
		1h	2 words in phase 1
		2h	3 words in phase 1
		...	...
		7Fh	128 words in phase 1
7-5	RWDLEN1	0-7h	Specifies the receive word length (number of bits) in phase 1.
		0	Receive word length is 8 bits.
		1h	Receive word length is 12 bits.
		2h	Receive word length is 16 bits.
		3h	Receive word length is 20 bits.
		4h	Receive word length is 24 bits.
		5h	Receive word length is 32 bits.
		6h-7h	Reserved
4	RWDREVRS		Receive 32-bit bit reversal enable bit.
		0	32-bit bit reversal is disabled.
		1	32-bit bit reversal is enabled. 32-bit data is received LSB first. RWDLEN1/2 bit should be set to 5h (32-bit operation); RCOMPAND bit should be set to 1h (transfer starts with LSB first); otherwise, operation is undefined.
3-0	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.

Copyright © 2013–2016, Texas Instruments Incorporated

24.3.5 Transmit Control Register (XCR)

The transmit control register (XCR) configures parameters of the transmit operations. The XCR is shown in Figure 24-46 and described in Table 24-27.

Figure 24-46. Transmit Control Register (XCR)

31	30 24	23 21	20 19	18	17 16
XPHASE	XFRLEN2	XWDLEN2	XCOMPAND	XFIG	XDATDLY
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15	14 8	7 5	4	3 0
Reserved	XFRLEN1	XWDLEN1	XWDREVRS	Reserved
R-0	R/W-0	R/W-0	R/W-0	R-0

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

Table 24-27. Transmit Control Register (XCR) Field Descriptions

Bit	Field	Value	Description
31	XPHASE		Transmit phases bit.
		0	Single-phase frame
		1	Dual-phase frame
30-24	XFRLEN2	0-7Fh	Specifies the transmit frame length (number of words) in phase 2.
		0	1 word in phase 2
		1h	2 words in phase 2
		2h	3 words in phase 2
		...	...
		7Fh	128 words in phase 2
23-21	XWDLEN2	0-7h	Specifies the transmit word length (number of bits) in phase 2.
		0	Transmit word length is 8 bits.
		1h	Transmit word length is 12 bits.
		2h	Transmit word length is 16 bits.
		3h	Transmit word length is 20 bits.
		4h	Transmit word length is 24 bits.
		5h	Transmit word length is 32 bits.
		6h-7h	Reserved
20-19	XCOMPAND	0-3h	Transmit companding mode bit. Modes other than 00 are only enabled when XWDLEN1/2 bit is 000 (indicating 8-bit data).
		0	No companding, data transfer starts with MSB first.
		1h	No companding, 8-bit data transfer starts with LSB first.
		2h	Compand using μ-law for transmit data.
		3h	Compand using A-law for transmit data.
18	XFIG		Transmit frame ignore bit.
		0	Transmit frame-synchronization pulses after the first pulse restarts the transfer.
		1	Transmit frame-synchronization pulses after the first pulse are ignored.
17-16	XDATDLY	0-3h	Transmit data delay bit.
		0	0-bit data delay
		1h	1-bit data delay
		2h	2-bit data delay
		3h	Reserved
15	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 24-27. Transmit Control Register (XCR) Field Descriptions (continued)

Bit	Field	Value	Description
14-8	XFRLEN1	0-7Fh	Specifies the transmit frame length (number of words) in phase 1.
		0	1 word in phase 1
		1h	2 words in phase 1
		2h	3 words in phase 1
		...	...
		7Fh	128 words in phase 1
7-5	XWDLEN1	0-7h	Specifies the transmit word length (number of bits) in phase 1.
		0	Transmit word length is 8 bits.
		1h	Transmit word length is 12 bits.
		2h	Transmit word length is 16 bits.
		3h	Transmit word length is 20 bits.
		4h	Transmit word length is 24 bits.
		5h	Transmit word length is 32 bits.
		6h-7h	Reserved
4	XWDREVRS		Transmit 32-bit bit reversal feature enable bit.
		0	32-bit bit reversal is disabled.
		1	32-bit bit reversal is enabled. 32-bit data is transmitted LSB first. XWDLEN1/2 bit should be set to 5h (32-bit operation); XCOMPAND bit should be set to 1h (transfer starts with LSB first); otherwise, operation is undefined.
3-0	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.

Copyright © 2013–2016, Texas Instruments Incorporated

24.3.6 Sample Rate Generator Register (SRGR)

The sample rate generator register (SRGR) controls the operation of various features of the sample rate generator. The SRGR is shown in Figure 24-47 and described in Table 24-28.

Figure 24-47. Sample Rate Generator Register (SRGR)

31	30	29	28	27	16
GSYNC	CLKSP	CLKSM	FSGM	FPER	
R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	

15	8	7	0
FWID		CLKGDV	
R/W-0		R/W-1	

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

Table 24-28. Sample Rate Generator Register (SRGR) Field Descriptions

Bit	Field	Value	Description
31	GSYNC		Sample-rate generator clock synchronization bit is only used when the external clock (CLKS) drives the sample-rate generator clock (CLKSM = 0).
		0	The sample-rate generator clock (CLKG) is free running.
		1	The sample-rate generator clock (CLKG) is running; however, CLKG is resynchronized and frame-sync signal (FSG) is generated only after detecting the receive frame-synchronization signal (FSR). Also, frame period (FPER) is a don't care because the period is dictated by the external frame-sync pulse.
30	CLKSP		CLKS polarity clock edge select bit is only used when the external clock (CLKS) drives the sample-rate generator clock (CLKSM = 0).
		0	Rising edge of CLKS generates CLKG and FSG.
		1	Falling edge of CLKS generates CLKG and FSG.
29	CLKSM		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 in the pin control register (PCR) to determine the source for the input clock. A CPU reset selects the McBSP internal input clock as the input clock and forces the CLKG frequency to 1/2 the McBSP internal input clock frequency.
		0	The input clock for the sample rate generator is taken from the CLKS pin or from the CLKR pin, depending on the value of the SCLKME bit in PCR: **CLKSM** / **SCLKME** / **Input Clock for Sample Rate Generator** 0 / 0 / Signal on CLKS pin 0 / 1 / Signal on CLKR pin
		1	The input clock for the sample rate generator is taken from the McBSP internal input clock or from the CLKX pin, depending on the value of the SCLKME bit in PCR: **CLKSM** / **SCLKME** / **Input Clock for Sample Rate Generator** 1 / 0 / McBSP internal input clock 1 / 1 / Signal on CLKX pin
28	FSGM		Sample-rate generator transmit frame-synchronization mode bit is only used when FSXM = 1 in PCR.
		0	Transmit frame-sync signal (FSX) is generated on every DXR-to-XSR copy. When FSGM = 0, FWID bit and FPER bit are ignored.
		1	Transmit frame-sync signal (FSX) is driven by the sample-rate generator frame-sync signal (FSG).
27-16	FPER	0-FFFh	Frame period value plus 1 specifies when the next frame-sync signal becomes active. Range is 1 to 4096 sample-rate generator clock (CLKG) periods.
15-8	FWID	0-FFh	Frame width value plus 1 specifies the width of the frame-sync pulse (FSG) during its active period.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 24-28. Sample Rate Generator Register (SRGR) Field Descriptions (continued)

Bit	Field	Value	Description
7-0	CLKGDV	0-FFh	Sample-rate generator clock (CLKG) divider value is used as the divide-down number to generate the required sample-rate generator clock frequency.

24.3.7 Multichannel Control Register (MCR)

The multichannel control register (MCR) has control and status bits for multichannel selection operation in the receiver (with an R prefix) and the same type of bits for the transmitter (with an X prefix). The MCR is shown in Figure 24-48 and described in Table 24-29. This I/O-mapped register enables you to:

- 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).

Figure 24-48. Multichannel Control Registers (MCR)

31	26	25	24	23	22	21	20	18	17	16
Reserved		XMCME	XPBBLK		XPABLK		XCBLK		XMCM	
R-0		R/W-0	R/W-0		R/W-0		R-0		R/W-0	

15	10	9	8	7	6	5	4	2	1	0
Reserved		RMCME	RPBBLK		RPABLK		RCBLK		Reserved	RMCM
R-0		R/W-0	R/W-0		R/W-0		R-0		R-0	R/W-0

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

Table 24-29. Multichannel Control Register (MCR) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	Reserved.
25	XMCME		Transmit multichannel partition mode bit. XMCME is only applicable if channels can be individually disabled/enabled or masked/unmasked for transmission (XMCM is nonzero). XMCME 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 transmit multichannel selection mode selected with the XMCM bit.
			If XMCM = 1 or 2h: assign 16 channels to partition A with the XPABLK bit and assign 16 channels to partition B with the XPBBLK bit.
			If XMCM = 3h: (for symmetric transmission and reception), assign 16 channels to receive partition A with the RPABLK bit and assign 16 channels to receive partition B with the RPBBLK bit.
		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 bit.
			You control the channels with the appropriate enhanced transmit channel enable register (XCERE*n*):
			XCERE0: Channels 0 through 31
			XCERE1: Channels 32 through 63
			XCERE2: Channels 64 through 95
			XCERE3: Channels 96 through 127

Copyright © 2013–2016, Texas Instruments Incorporated

Table 24-29. Multichannel Control Register (MCR) Field Descriptions (continued)

Bit	Field	Value	Description
24-23	XPBBLK	0-3h	Transmit partition B block bit. 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 the XPBBLK bit to assign one of the odd-numbered blocks (1, 3, 5, or 7) to partition B; 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 bit is regularly updated to indicate which block is active. When XMCM = 3h (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
22-21	XPABLK	0-3h	Transmit partition A block bit. 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 XPBBLK bit description 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
20-18	XCBLK	0-7h	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

Copyright © 2013–2016, Texas Instruments Incorporated

Table 24-29. Multichannel Control Register (MCR) Field Descriptions (continued)

Bit	Field	Value	Description
17-16	XMCM	0-3h	Transmit multichannel selection mode bit. XMCM determines whether all channels or only selected channels are enabled and unmasked for transmission.
		0	Transmit multichannel selection is off. 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 enhanced transmit channel enable register (XCERE*n*). If enabled, a channel in this mode is also unmasked.
			The XMCME bit determines whether 32 channels or 128 channels are selectable in XCERE*n*.
		2h	All channels are enabled, but they are masked unless they are selected in the appropriate enhanced transmit channel enable register (XCERE*n*).
			The XMCME bit determines whether 32 channels or 128 channels are selectable in XCERE*n*.
		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 enhanced receive channel enable register (RCERE*n*). Once enabled, they are masked unless they are also selected in the appropriate enhanced transmit channel enable register (XCERE*n*).
			The XMCME bit determines whether 32 channels or 128 channels are selectable in RCERE*n* and XCERE*n*.
15-10	Reserved	0	Reserved.
9	RMCME		Receive multichannel partition mode bit. RMCME is only applicable if channels can be individually disabled/enabled 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 bit and assign 16 channels to partition B with the RPBBLK bit.
			You control the channels with the enhanced receive channel enable register partition A/B (RCERE0).
		1	You can control up to 128 channels in the receive multichannel selection mode.
			You control the channels with the appropriate enhanced receive channel enable register (RCERE*n*):
			RCERE0: Channels 0 through 31
			RCERE1: Channels 32 through 63
			RCERE2: Channels 64 through 95
			RCERE3: Channels 96 through 127
8-7	RPBBLK	0-3h	Receive partition B block bit.
			RPBBLK is only applicable if channels can be individually disabled/enabled (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 the RPBBLK bit to assign one of the odd-numbered blocks (1, 3, 5, or 7) to partition B; use the RPABLK 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 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 bit is regularly updated to indicate which block is active.
			When XMCM = 3h (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

Copyright © 2013–2016, Texas Instruments Incorporated

Table 24-29. Multichannel Control Register (MCR) Field Descriptions (continued)

Bit	Field	Value	Description
6-5	RPABLK	0-3h	Receive partition A block bit. RPABLK is only applicable if channels can be individually disabled/enabled (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 RPBBLK bit description 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	RCBLK	0-7h	Receive current block indicator. RCBLK indicates which block of 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.
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	Multichannel selection mode. Channels can be individually enabled or disabled. The only channels enabled are those selected in the appropriate enhanced receive channel enable register (RCERE*n*). The way channels are assigned to RCERE*n* depends on the number of receive channel partitions (2 or 8), as defined by the RMCME bit.

Copyright © 2013–2016, Texas Instruments Incorporated

24.3.8 Enhanced Receive Channel Enable Registers (RCERE0-RCERE3)

The enhanced receive channel enable register (RCERE*n*) is shown in Figure 24-49 and described in Table 24-30. The RCERE*n* is used to enable any of 128 elements for receive. RCERE0 is the only register used in normal mode (up to 32 channels can be selected in partitions A and B, RMCME = XMCME = 0 in MCR). RCERE0-RCERE3 are used when in enhanced mode (up to 128 channels can be selected in all partitions, RMCME = XMCME = 1 in MCR).

The receive multichannel partition mode (RMCME) bit in the multichannel control register (MCR) is only applicable if channels can be individually disabled/enabled for reception (RMCM = 1). The RMCME bit determines whether only 32 channels or all 128 channels are to be individually selectable:

- **When RMCME = 0:** Only partitions A and B are used. RCERE0 is used to enable any of the 32 elements for a receive. Of the 32 elements, 16 channels belong to a subframe in partition A and 16 channels belong to a subframe in partition B. The RCE0-RCE15 bits enable elements within the 16-channel elements in partition A and the RCE16-RCE31 bits enable elements within the 16-channel elements in partition B. The 16 channels in partition A are assigned with the RPABLK bit in MCR and the 16 channels in partition B are assigned with the RPBBLK bit in MCR.
- **When RMCME = 1:** All partitions are used. RCERE0 is used to enable any of the 32 elements in channels 0 through 31 for a receive. Of the 32 elements, channels 0 to 15 belong to a subframe in partition A and channels 16 to 31 belong to a subframe in partition B. The RCE0-RCE15 bits enable elements within the 16-channel elements in partition A and the RCE16-RCE31 bits enable elements within the 16-channel elements in partition B.

Section 24.3.8.1 shows the 128 channels in a multichannel data stream and their corresponding enable bits in RCERE*n*.

Figure 24-49. Enhanced Receive Channel Enable Register *n* (RCERE*n*)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RCE31	RCE30	RCE29	RCE28	RCE27	RCE26	RCE25	RCE24	RCE23	RCE22	RCE21	RCE20	RCE19	RCE18	RCE17	RCE16
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	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
RCE15	RCE14	RCE13	RCE12	RCE11	RCE10	RCE9	RCE8	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	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 afer reset

Table 24-30. Enhanced Receive Channel Enable Register *n* (RCERE*n*) Field Descriptions

Bit	Field	Value	Description
31-0	RCE*n*		Receive channel enable bit. For receive multichannel selection mode (RMCM = 1 in MCR).
		0	Disable the channel that is mapped to RCE*n*.
		1	Enable the channel that is mapped to RCE*n*.

Copyright © 2013–2016, Texas Instruments Incorporated

24.3.8.1 RCERE*n* Used in the Receive Multichannel Selection Mode

For multichannel selection operation, the assignment of channels to the enhanced receive channel enable register (RCERE*n*) depends on whether 32 or 128 channels are individually selectable, as defined by the RMCME bit in the multichannel control register (MCR). For each of these two cases, Table 24-31 shows which block of channels is assigned to each RCERE*n* used. For each RCERE*n*, Table 24-31 shows which channel is assigned to each of the bits.

Table 24-31. Use of the Receive Channel Enable Registers

Number of selectable channels	Block Assignments		Channel Assignments	
	RCERE*n*	Block assigned[1]	Bit in RCERE*n*	Channel assigned[1]
32	RCERE0	Channels *n* to (*n* + 15)	RCE0	Channel *n*
(RMCME = 0)		The block of channels (0, 2, 4, or 6) is selected with the RPABLK bit in MCR	RCE1	Channel (*n* + 1)
			...	...
			RCE15	Channel (*n* + 15)
		Channels *m* to (*m* + 15)	RCE16	Channel *m*
		The block of channels (1, 3, 5, or 7) is selected with the RPBBLK bit in MCR	RCE17	Channel (*m* + 1)
			...	...
			RCE31	Channel (*m* + 15)
128	RCERE0	Block 0	RCE0	Channel 0
(RMCME = 1)			...	...
			RCE15	Channel 15
		Block 1	RCE16	Channel 16
			...	...
			RCE31	Channel 31
	RCERE1	Block 2	RCE0	Channel 32
			...	...
			RCE15	Channel 47
		Block 3	RCE16	Channel 48
			...	...
			RCE31	Channel 63
	RCERE2	Block 4	RCE0	Channel 64
			...	...
			RCE15	Channel 79
		Block 5	RCE16	Channel 80
			...	...
			RCE31	Channel 95
	RCERE3	Block 6	RCE0	Channel 96
			...	...
			RCE15	Channel 111
		Block 7	RCE16	Channel 112
			...	...
			RCE31	Channel 127

[1] *n* is any even-numbered block 0, 2, 4, or 6. *m* is any odd-numbered block 1, 3, 5, or 7.

Copyright © 2013–2016, Texas Instruments Incorporated

24.3.9 Enhanced Transmit Channel Enable Registers (XCERE0-XCERE3)

The enhanced transmit channel enable register (XCERE*n*) is shown in Figure 24-50 and described in Table 24-32. The XCERE*n* is used to enable any of 128 elements for transmit. XCERE0 is the only register used in normal mode (up to 32 channels can be selected in partitions A and B, RMCME = XMCME = 0 in MCR). XCERE0-XCERE3 are used when in enhanced mode (up to 128 channels can be selected in all partitions, RMCME = XMCME = 1 in MCR).

The transmit multichannel partition mode (XMCME) bit in the multichannel control register (MCR) is only applicable if channels can be individually disabled/enabled or masked/unmasked for transmission (XMCM is nonzero). The XMCME bit determines whether only 32 channels or all 128 channels are to be individually selectable:

- **When XMCME = 0:** Only partitions A and B are used. XCERE0 is used to enable any of the 32 elements for a transmit. Of the 32 elements, 16 channels belong to a subframe in partition A and 16 channels belong to a subframe in partition B. The XCE0-XCE15 bits enable elements within the 16-channel elements in partition A and the XCE16-XCE31 bits enable elements within the 16-channel elements in partition B. You can control up to 32 channels in the transmit multichannel selection mode selected with the XMCM bit in MCR.
- **When XMCME = 1:** All partitions are used. XCERE0 is used to enable any of the 32 elements in channels 0 through 31 for a transmit. Of the 32 elements, channels 0 to 15 belong to a subframe in partition A and channels 16 to 31 belong to a subframe in partition B. The XCE0-XCE15 bits enable elements within the 16-channel elements in partition A and the XCE16-XCE31 bits enable elements within the 16-channel elements in partition B.

Section 24.3.9.1 shows the 128 channels in a multichannel data stream and their corresponding enable bits in XCERE*n*.

Figure 24-50. Enhanced Transmit Channel Enable Register *n* (XCERE*n*)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
XCE31	XCE30	XCE29	XCE28	XCE27	XCE26	XCE25	XCE24	XCE23	XCE22	XCE21	XCE20	XCE19	XCE18	XCE17	XCE16
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	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
XCE15	XCE14	XCE13	XCE12	XCE11	XCE10	XCE9	XCE8	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	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 afer reset

Table 24-32. Enhanced Transmit Channel Enable Register *n* (XCERE*n*) Field Descriptions

Bit	Field	Value	Description
31-0	XCE*n*		Transmit channel enable bit. The role of this bit depends on which transmit multichannel mode is selected with the XMCM bit in MCR:
			When XMCM = 1h (all channels disabled unless selected):
		0	Disable and mask the channel that is mapped to XCE*n.*
		1	Enable and unmask the channel that is mapped to XCE*n*.
			When XMCM = 2h (all channels enabled but masked unless selected):
		0	Mask the channel that is mapped to XCE*n*.
		1	Unmask the channel that is mapped to XCE*n*.
			When XMCM = 3h (all channels masked unless selected):
		0	Mask the channel that is mapped to XCE*n*. Even if this channel is enabled by the corresponding enhanced receive channel enable bit, this channel's data cannot appear on the DX pin.
		1	Unmask the channel that is mapped to XCE*n*. Even if this channel is also enabled by the corresponding enhanced receive channel enable bit, full transmission can occur.

Copyright © 2013–2016, Texas Instruments Incorporated

24.3.9.1 XCER*En* Used in the Transmit Multichannel Selection Mode

For multichannel selection operation, the assignment of channels to the enhanced transmit channel enable register (XCERE*n*) depends on whether 32 or 128 channels are individually selectable, as defined by the XMCME bit in the multichannel control register (MCR). For each of these two cases, Table 24-33 shows which block of channels is assigned to each XCERE*n* used. For each XCERE*n*, Table 24-33 shows which channel is assigned to each of the bits.

When XMCM = 3h (symmetric transmission and reception), the transmitter uses the enhanced receive channel enable register (RCERE*n*) to enable channels and uses XCERE*n* to unmask channels for transmission.

Table 24-33. Use of the Transmit Channel Enable Registers

	Block Assignments		Channel Assignments	
Number of selectable channels	**XCERE*n***	**Block assigned[(1)]**	**Bit in XCERE*n***	**Channel assigned[(1)]**
32	XCERE0	Channels *n* to (*n* + 15)	XCE0	Channel *n*
(XMCME = 0)		When XMCM = 1h or 2h, the block of channels (0, 2, 4, or 6) is selected with the XPABLK bit in MCR. When XMCM = 3h, the block of channels (0, 2, 4, or 6) is selected with the RPABLK bit in MCR.	XCE1	Channel (*n* + 1)
			...	...
			XCE15	Channel (*n* + 15)
		Channels *m* to (*m* + 15)	XCE16	Channel *m*
		When XMCM = 1h or 2h, the block of channels (1, 3, 5, or 7) is selected with the XPBBLK bit in MCR. When XMCM = 3h, the block of channels (1, 3, 5, or 7) is selected with the RPBBLK bit in MCR.	XCE17	Channel (*m* + 1)
			...	...
			XCE31	Channel (*m* + 15)
128	XCERE0	Block 0	XCE0	Channel 0
(XMCME = 1)			...	...
			XCE15	Channel 15
		Block 1	XCE16	Channel 16
			...	...
			XCE31	Channel 31
	XCERE1	Block 2	XCE0	Channel 32
			...	...
			XCE15	Channel 47
		Block 3	XCE16	Channel 48
			...	...
			XCE31	Channel 63
	XCERE2	Block 4	XCE0	Channel 64
			...	...
			XCE15	Channel 79
		Block 5	XCE16	Channel 80
			...	...
			XCE31	Channel 95

(1) *n* is any even-numbered block 0, 2, 4, or 6. *m* is any odd-numbered block 1, 3, 5, or 7.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 24-33. Use of the Transmit Channel Enable Registers (continued)

Number of selectable channels	Block Assignments		Channel Assignments	
	XCERE*n*	Block assigned[1]	Bit in XCERE*n*	Channel assigned[1]
	XCERE3	Block 6	XCE0	Channel 96
			...	...
			XCE15	Channel 111
		Block 7	XCE16	Channel 112
			...	...
			XCE31	Channel 127

24.3.10 Pin Control Register (PCR)

The serial port is configured via the serial port control register (SPCR) and the pin control register (PCR). The PCR contains McBSP status control bits. The PCR is shown in Figure 24-51 and described in Table 24-34.

Figure 24-51. Pin Control Register (PCR)

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		Reserved[1]	Reserved[1]	FSXM	FSRM	CLKXM	CLKRM
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
SCLKME	Reserved[1]	Reserved	Reserved	FSXP	FSRP	CLKXP	CLKRP
R/W-0	R-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R = Read only; R/W = Rear/Write; -n = value afer reset

[1] If writing to this field, always write the default value of 0 to ensure proper McBSP operation.

Table 24-34. Pin Control Register (PCR) Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.
13-12	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect. If writing to this field, always write the default value of 0 to ensure proper McBSP operation.
11	FSXM		Transmit frame-synchronization mode bit.
		0	Frame-synchronization signal is derived from an external source.
		1	Frame-synchronization signal is determined by FSGM bit in SRGR.
10	FSRM		Receive frame-synchronization mode bit.
		0	Frame-synchronization signal is derived from an external source. FSR is an input pin.
		1	Frame-synchronization signal is generated internally by the sample-rate generator. FSR is an output pin.
9	CLKXM		Transmit clock mode bit. When CLKSTP bit in SPCR is cleared to 0:
		0	CLKX is an input pin and is driven by an external clock.
		1	CLKX is an output pin and is driven by the internal sample-rate generator.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 24-34. Pin Control Register (PCR) Field Descriptions (continued)

Bit	Field	Value	Description
8	CLKRM		Receive clock mode bit.
			Digital loop back mode is disabled (DLB = 0 in SPCR):
		0	CLKR is an input pin and is driven by an external clock.
		1	CLKR is an output pin and is driven by the internal sample-rate generator.
			Digital loop back mode is enabled (DLB = 1 in SPCR):
		0	Receive clock (not the CLKR pin) is driven by transmit clock (CLKX) that is based on CLKXM bit. CLKR pin is in high-impedance state.
		1	CLKR is an output pin and is driven by the transmit clock. The transmit clock is based on 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 frequency = Input clock frequency/(CLKGDV + 1)* SCLKME is used in conjunction with the CLKSM bit in the sample rate generator register (SRGR) to select the input clock. A CPU reset selects the McBSP internal input clock as the input clock and forces the CLKG frequency to 1/2 the McBSP internal input clock frequency.
		0	The input clock for the sample rate generator is taken from the CLKS pin or from the McBSP internal input clock, depending on the value of the CLKSM bit in SRGR: **SCLKME** / **CLKSM** / **Input Clock for Sample Rate Generator** 0 / 0 / Signal on CLKS pin 0 / 1 / McBSP internal input clock
		1	The input clock for the sample rate generator is taken from the CLKR pin or from the CLKX pin, depending on the value of the CLKSM bit in SRGR: **SCLKME** / **CLKSM** / **Input Clock for Sample Rate Generator** 1 / 0 / Signal on CLKR pin 1 / 1 / Signal on CLKX pin
6	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect. If writing to this field, always write the default value of 0 to ensure proper McBSP operation.
5-4	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.
3	FSXP		Transmit frame-synchronization polarity bit.
		0	Transmit frame-synchronization pulse is active high.
		1	Transmit frame-synchronization pulse is active low.
2	FSRP		Receive frame-synchronization polarity bit.
		0	Receive frame-synchronization pulse is active high.
		1	Receive frame-synchronization pulse is active low.
1	CLKXP		Transmit clock polarity bit.
		0	Transmit data driven on rising edge of CLKX.
		1	Transmit data driven on falling edge of CLKX.
0	CLKRP		Receive clock polarity bit.
		0	Receive data sampled on falling edge of CLKR.
		1	Receive data sampled on rising edge of CLKR.

Copyright © 2013–2016, Texas Instruments Incorporated

24.3.11 BFIFO Revision Identification Register (BFIFOREV)

The Buffer FIFO (BFIFO) revision identification register (BFIFOREV) contains revision data for the Buffer FIFO (BFIFO). The BFIFOREV is shown in Figure 24-52 and described in Table 24-35.

Figure 24-52. BFIFO Revision Identification Register (BFIFOREV)

31	0
REV	
R-4431 1100h	

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

Table 24-35. BFIFO Revision Identification Register (BFIFOREV) Field Descriptions

Bit	Field	Value	Description
31-0	REV	4431 1100h	Identifies revision of Buffer FIFO.

Copyright © 2013–2016, Texas Instruments Incorporated

24.3.12 Write FIFO Control Register (WFIFOCTL)

The Write FIFO control register (WFIFOCTL) is shown in Figure 24-53 and described in Table 24-36.

NOTE: The WNUMEVT and WNUMDMA values must be set prior to enabling the Write FIFO. If the Write FIFO is to be enabled, it must be enabled prior to taking the McBSP out of reset.

Figure 24-53. Write FIFO Control Register (WFIFOCTL)

31	17	16
Reserved		WENA
R-0		R/W-0

15	8	7	0
WNUMEVT		WNUMDMA	
R/W-10h		R/W-4h	

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

Table 24-36. Write FIFO Control Register (WFIFOCTL) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16	WENA		Write FIFO enable bit.
		0	Write FIFO is disabled. The WLVL bit in the Write FIFO status register (WFIFOSTS) is reset to 0 and the pointers are initialized, that is, the Write FIFO is "flushed."
		1	Write FIFO is enabled. If the Write FIFO is to be enabled, it must be enabled prior to taking the McBSP out of reset.
15-8	WNUMEVT	0-FFh	Write word count per DMA event (32-bit). When the Write FIFO has space for at least *WNUMEVT* words of data, then an XEVT (transmit DMA event) is generated to the host/DMA controller. This value must be set prior to enabling the Write FIFO.
		0	0 words
		1h	1 word
		2h	2 words
		...	...
		40h	64 words
		41h-FFh	Reserved
7-0	WNUMDMA	0-FFh	Write word count per transfer (32-bit words). Upon a transmit DMA event from the McBSP, *WNUMDMA* words are transferred from the Write FIFO to the McBSP. This value must be set prior to enabling the Write FIFO.
		0	0 words
		1	1 word
		2h-FFh	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

24.3.13 Write FIFO Status Register (WFIFOSTS)

The Write FIFO status register (WFIFOSTS) is shown in Figure 24-54 and described in Table 24-37.

Figure 24-54. Write FIFO Status Register (WFIFOSTS)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		WLVL	
R-0		R-0	

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

Table 24-37. Write FIFO Status Register (WFIFOSTS) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	WLVL	0-FFh	Write level. Number of 32-bit words currently in the Write FIFO.
		0	0 words currently in the Write FIFO.
		1h	1 word currently in the Write FIFO.
		2h	2 words currently in the Write FIFO.
		...	...
		40h	64 words currently in the Write FIFO.
		41h-FFh	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

24.3.14 Read FIFO Control Register (RFIFOCTL)

The Read FIFO control register (RFIFOCTL) is shown in Figure 24-55 and described in Table 24-38.

NOTE: The RNUMEVT and RNUMDMA values must be set prior to enabling the Read FIFO. If the Read FIFO is to be enabled, it must be enabled prior to taking the McBSP out of reset.

Figure 24-55. Read FIFO Control Register (RFIFOCTL)

31	17	16
Reserved		RENA
R-0		R/W-0

15	8	7	0
RNUMEVT		RNUMDMA	
R/W-10h		R/W-4h	

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

Table 24-38. Read FIFO Control Register (RFIFOCTL) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16	RENA		Read FIFO enable bit.
		0	Read FIFO is disabled. The RLVL bit in the Read FIFO status register (RFIFOSTS) is reset to 0 and the pointers are initialized, that is, the Read FIFO is "flushed."
		1	Read FIFO is enabled. If the Read FIFO is to be enabled, it must be enabled prior to taking the McBSP out of reset.
15-8	RNUMEVT	0-FFh	Read word count per DMA event (32-bit). When the Read FIFO contains at least *RNUMEVT* words of data, then an REVT (receive DMA event) is generated to the host/DMA controller. This value must be set prior to enabling the Read FIFO.
		0	0 words
		1h	1 word
		2h	2 words
		...	...
		40h	64 words
		41h-FFh	Reserved
7-0	RNUMDMA	0-FFh	Read word count per transfer (32-bit words). Upon a receive DMA event from the McBSP, the Read FIFO reads *RNUMDMA* words from the McBSP. This value must be set prior to enabling the Read FIFO.
		0	0 words
		1	1 word
		2h-FFh	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

24.3.15 Read FIFO Status Register (RFIFOSTS)

The Read FIFO status register (RFIFOSTS) is shown in Figure 24-56 and described in Table 24-39.

Figure 24-56. Read FIFO Status Register (RFIFOSTS)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		RLVL	
R-0		R-0	

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

Table 24-39. Read FIFO Status Register (RFIFOSTS) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	RLVL	0-FFh	Read level. Number of 32-bit words currently in the Read FIFO.
		0	0 words currently in the Read FIFO.
		1h	1 word currently in the Read FIFO.
		2h	2 words currently in the Read FIFO.
		...	...
		40h	64 words currently in the Read FIFO.
		41h-FFh	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

Multimedia Card (MMC)/Secure Digital (SD) Card Controller

This chapter describes the multimedia card (MMC)/secure digital (SD) card controller.

Topic **Page**

25.1 Introduction **1129**
25.2 Architecture **1130**
25.3 Procedures for Common Operations **1146**
25.4 Registers **1158**

Copyright © 2013–2016, Texas Instruments Incorporated

25.1 Introduction

25.1.1 Purpose of the Peripheral

A number of applications use the multimedia card (MMC)/secure digital (SD) card to provide removable data storage. The MMC/SD card controller provides an interface to external MMC and SD cards. The communication between the MMC/SD card controller and MMC/SD card(s) is performed according to the MMC/SD protocol.

25.1.2 Features

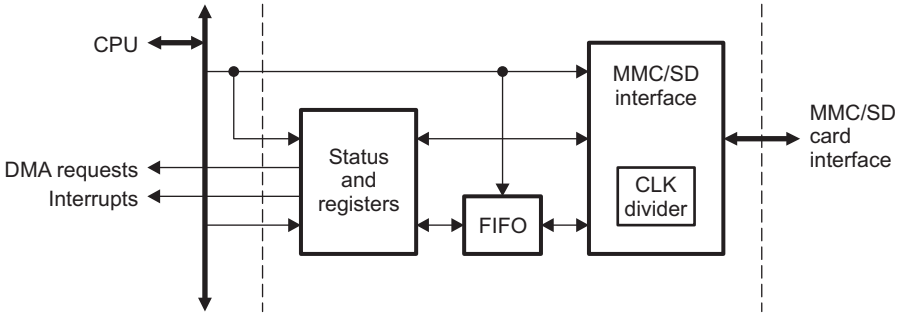
The MMC/SD card controller has the following features:

- Supports interface to multimedia cards (MMC)
- Supports interface to secure digital (SD) memory cards
- Ability to use the MMC/SD protocol and Secure Digital Input Output (SDIO) protocol
- Programmable frequency of the clock that controls the timing of transfers between the MMC/SD controller and memory card
- 512-bit read/write FIFO to lower system overhead
- Signaling to support enhanced direct memory access (EDMA) transfers (slave)
- Maximum clock to MMC varies based on core voltage (see your device-specific data manual)
- Maximum clock to SD varies based on core voltage (see your device-specific data manual)

25.1.3 Functional Block Diagram

The MMC/SD card controller transfers data between the CPU and the EDMA controller on one side and the MMC/SD card on the other side, as shown in Figure 25-1. This means you have a choice of performing data transfers using the CPU or EDMA as a mechanism to move data between the device memory and the FIFO. The CPU and the EDMA controller can read from or write to the data in the card by accessing the registers in the MMC/SD controller.

Figure 25-1. MMC/SD Card Controller Block Diagram



25.1.4 Supported Use Case Statement

The MMC/SD card controller supports the following user cases:

- MMC/SD card identification
- MMC/SD single-block read using CPU
- MMC/SD single-block read using EDMA
- MMC/SD single-block write using CPU
- MMC/SD single-block write using EDMA
- MMC/SD multiple-block read using CPU
- MMC/SD multiple-block read using EDMA

Copyright © 2013–2016, Texas Instruments Incorporated

- MMC/SD multiple-block write using CPU
- MMC/SD multiple-block write using EDMA

25.1.5 Industry Standard(s) Compliance Statement

The MMC/SD card controller supports the following industry standards (with the exception noted below):

- MMC (Multimedia Card) Specification v4.0
- SD (Secure Digital) Physical Layer Specification v1.1
- SDIO (Secure Digital Input Output) Specification v2.0

The information in this chapter assumes that you are familiar with these standards.

The MMC/SD controller does not support the SPI mode of operation.

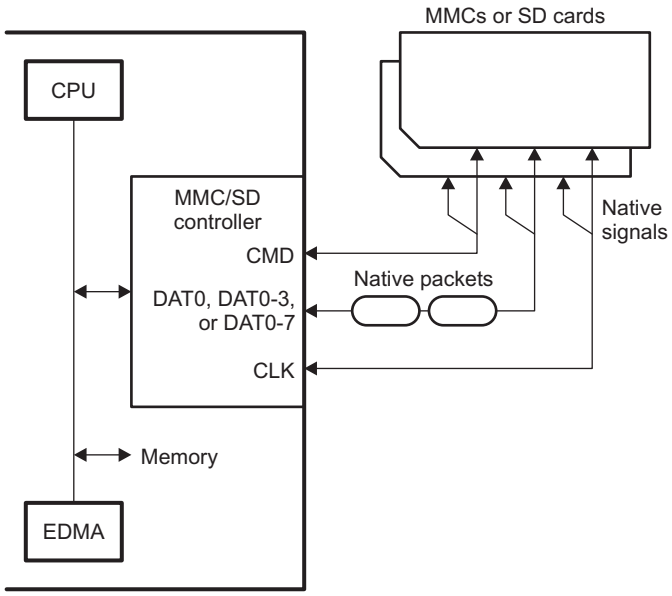
25.2 Architecture

The MMC/SD controller uses the MMC/SD protocol to communicate with the MMC/SD cards. You can configure the MMC/SD controller to work as an MMC or SD controller, based on the type of card the controller is communicating with. Figure 25-2 summarizes the MMC/SD mode interface. Figure 25-3 illustrates how the controller interfaces to the cards in MMC/SD mode.

In the MMC/SD mode, the MMC controller supports one or more MMC/SD cards. Regardless of the number of cards connected, the MMC/SD controller selects one by using identification broadcast on the data line. The following MMC/SD controller pins are used:

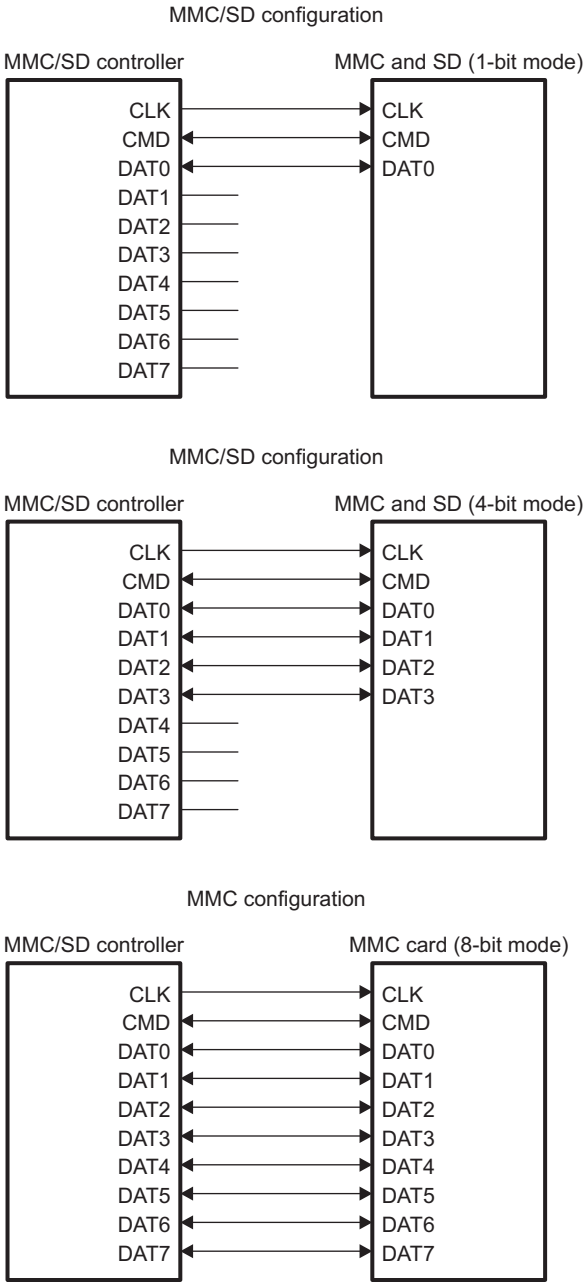
- MMCSD_CMD: This pin is used for two-way communication between the connected card and the MMC/SD controller. The MMC/SD controller transmits commands to the card and the memory card drives responses to the commands on this pin.
- MMCSD_DAT0, MMCSD_DAT0-3, or MMCSD_DAT0-7: MMC cards only use one data line (DAT0), four data lines (DAT0-3), or eight data lines (DAT0-7), and SD cards use one data line (DAT0) or four data lines (DAT0-3). The number of MMCSD_DAT pins (the data bus width) is set by the WIDTH bit in the MMC control register (MMCCTL), see Section 25.4.1).
- MMCSD_CLK: This pin provides the clock to the memory card from the MMC/SD controller.

Figure 25-2. MMC/SD Controller Interface Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 25-3. MMC Configuration and SD Configuration Diagram



25.2.1 Clock Control

There are two clocks, the function clock and the memory clock, in the MMC/SD controller (Figure 25-4).

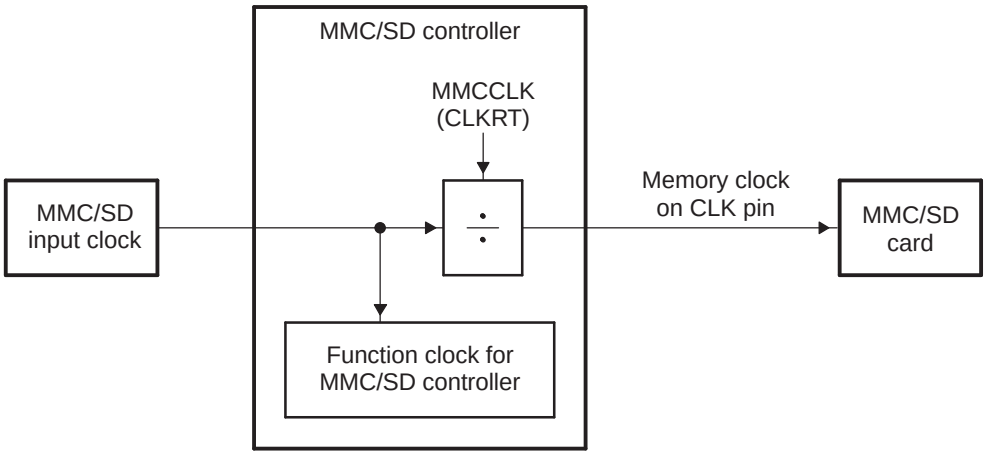
The function clock determines the operational frequency of the MMC/SD controller and is the input clock to the MMC/SD card(s).

Copyright © 2013–2016, Texas Instruments Incorporated

The memory clock appears on the MMCSD_CLK pin of the MMC/SD controller interface. The memory clock controls the timing of communication between the MMC/SD controller and the connected memory card. The memory clock is generated by dividing the function clock in the MMC/SD controller. The divide-down value is set by CLKRT bits in the MMC memory clock control register (MMCCLK) and is determined by the following equation:

memory clock frequency = function clock frequency/(2 × (CLKRT + 1))

Figure 25-4. MMC/SD Controller Clocking Diagram



25.2.2 Signal Descriptions

Table 25-1 shows the MMC/SD controller pins that each mode uses. The MMC/SD protocol uses the clock, command (two-way communication between the MMC controller and memory card), and data (MMCSD_DAT0, MMCSD_DAT0-3, or MMCSD_DAT0-7 for MMC card; MMCSD_DAT0 or MMCSD_DAT0-3 for SD card) pins.

Table 25-1. MMC/SD Controller Pins Used in Each Mode

		Function		
Pin	**Type[1]**	**MMC and SD (1-bit mode) Communications**	**MMC and SD (4-bit mode) Communications**	**MMC (8-bit mode) Communication**
MMCSD_CLK	O	Clock line	Clock line	Clock line
MMCSD_CMD	I/O	Command line	Command line	Command line
MMCSD_DAT0	I/O	Data line 0	Data line 0	Data line 0
MMCSD_DAT1	I/O	(Not used)	Data line 1	Data line 1
MMCSD_DAT2	I/O	(Not used)	Data line 2	Data line 2
MMCSD_DAT3	I/O	(Not used)	Data line 3	Data line 3
MMCSD_DAT4	I/O		(Not used)	Data line 4
MMCSD_DAT5	I/O		(Not used)	Data line 5
MMCSD_DAT6	I/O		(Not used)	Data line 6
MMCSD_DAT7	I/O		(Not used)	Data line 7

(1) I = input to the MMC controller; O = output from the MMC controller.

Copyright © 2013–2016, Texas Instruments Incorporated

25.2.3 Protocol Descriptions

The MMC/SD controller follows the MMC/SD protocol for completing any kind of transaction with the multimedia card and secure digital cards. For more detailed information, refer to the supported MMC and SD specifications in Section 25.1.5.

25.2.3.1 MMC/SD Mode Write Sequence

Figure 25-5 and Table 25-2 show the signal activity when the MMC/SD controller is in the MMC/SD mode and is writing data to a memory card. The same block length must be defined in the MMC/SD controller and in the memory card before initiating a data write. In a successful write protocol sequence, the following steps occur:

- The MMC/SD controller requests the CSD content.
- The card receives the command and sends the content of the CSD register as its response.
- If the desired block length, WRITE_BL_LEN value, is different from the default value determined from the response, the MMC/SD controller sends the block length command.
- The card receives the command and sends responses to the command.
- The MMC/SD controller requests the card to change states from standby to transfer.
- The card receives the command and sends responses to the command.
- The MMC/SD controller sends a write command to the card.
- The card receives the command and sends responses to the command.
- The MMC/SD controller sends a block of data to the card.
- The card sends the CRC status to the MMC/SD controller.
- The card sends a low BUSY bit until all of the data has been programmed into the flash memory inside the card.

Figure 25-5. MMC/SD Mode Write Sequence Timing Diagram

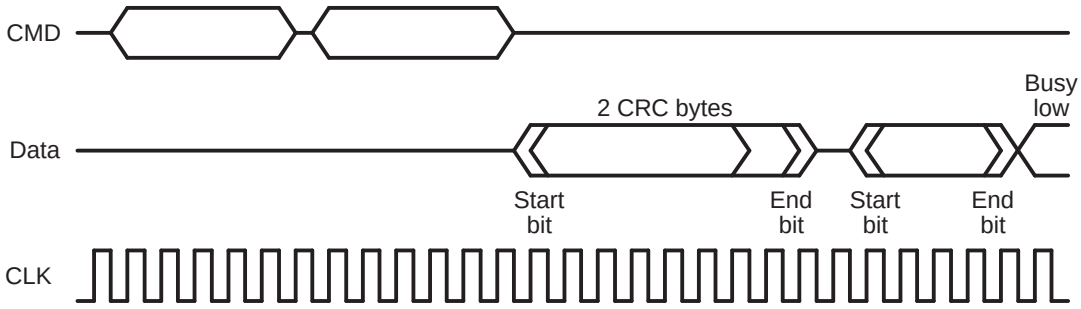


Table 25-2. MMC/SD Mode Write Sequence

Portion of the Sequence	Description
WR CMD	Write command: A 6-byte WRITE_BLOCK command token is sent from the CPU to the card.
CMD RSP	Command response: The card sends a 6-byte response of type R1 to acknowledge the WRITE_BLOCK to the CPU.
DAT BLK	Data block: The CPU writes a block of data to the card. The data content is preceded by one start bit and is followed by two CRC bytes and one end bit.
CRC STAT	CRC status: The card sends a one byte CRC status information, which indicates to the CPU whether the data has been accepted by the card or rejected due to a CRC error. The CRC status information is preceded by one start bit and is followed by one end bit.
BSY	BUSY bit: The CRC status information is followed by a continuous stream of low busy bits until all of the data has been programmed into the flash memory on the card.

Copyright © 2013–2016, Texas Instruments Incorporated

25.2.3.2 MMC/SD Mode Read Sequence

Figure 25-6 and Table 25-3 show the signal activity when the MMC controller is in the MMC/SD mode and is reading data from a memory card. The same block length must be defined in the MMC controller and in the memory card before initiating a data read. In a successful read protocol sequence, the following steps occur:

- The MMC/SD controller requests for the CSD content.
- The card receives the command and sends the content of the CSD register as its response.
- If the desired block length, READ_BL_LEN value, is different from the default value determined from the response, the MMC/SD controller sends the block length command.
- The card receives the command and sends responses to the command.
- The MMC/SD controller requests the card to change state from stand-by to transfer.
- The card receives the command and sends responses to the command.
- The MMC/SD controller sends a read command to the card.
- The card drives responses to the command.
- The card sends a block of data to the CPU.

Figure 25-6. MMC/SD Mode Read Sequence Timing Diagram

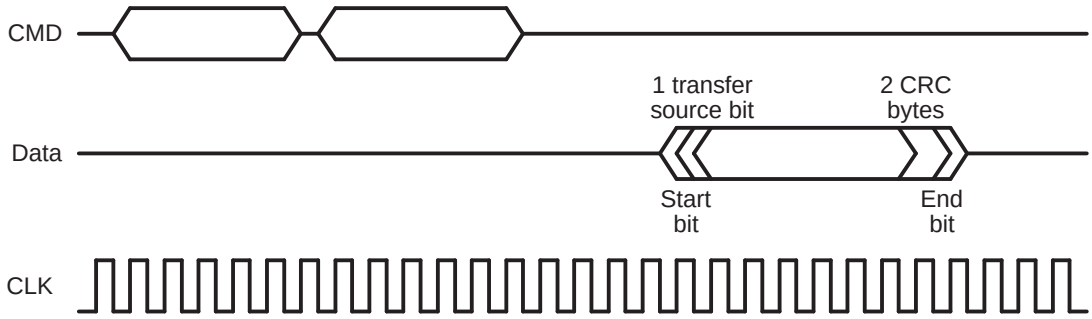


Table 25-3. MMC/SD Mode Read Sequence

Portion of the Sequence	Description
RD CMD	Read command: A 6-byte READ_SINGLE_BLOCK command token is sent from the CPU to the card.
CMD RSP	Command response: The card sends a response of type R1 to acknowledge the READ_SINGLE_BLOCK command to the CPU.
DAT BLK	Data block: The card sends a block of data to the CPU. The data content is preceded by a start bit and is followed by two CRC byte and an end bit.

25.2.4 Data Flow in the Input/Output FIFO

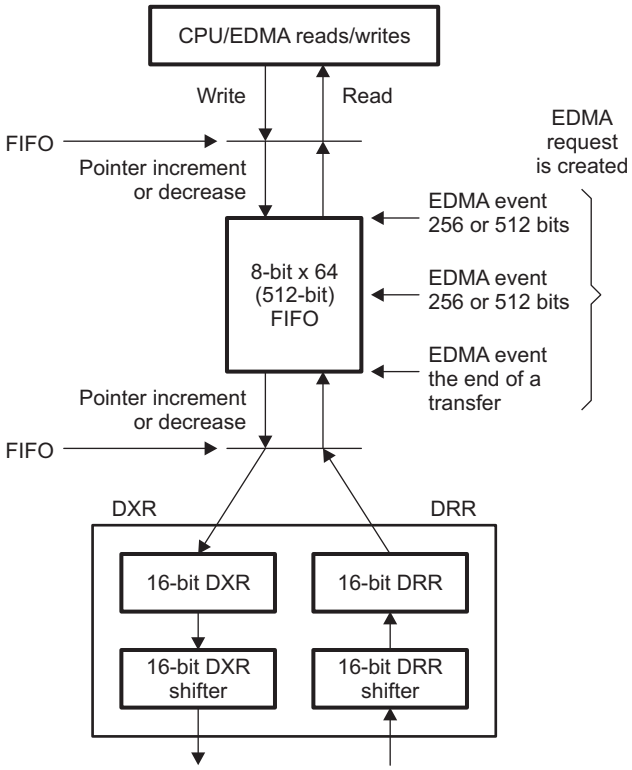
The MMC/SD controller contains a single 512-bit FIFO, organized as 8-bit × 64 entries, that is used for both reading data from the memory card and writing data to the memory card (see Figure 25-7). The conversion from the 32-bit bus to the byte format of the FIFO follows the little-endian convention (details are provided in later sections). The read and write FIFOs act as an interim location to store data transferred from/to the card momentarily via the CPU or EDMA. The FIFO includes logic to generate EDMA events and interrupts based on the amount of data in the FIFO and a programmable number of bytes received/transmitted. Flags are set when the FIFO is full or empty.

Copyright © 2013–2016, Texas Instruments Incorporated

A high-level operational description is as follows:

- Data is written to the FIFO through the MMC data transmit register (MMCDXR). Data is read from the FIFO through the MMC data receive register (MMCDRR). This is true for both the CPU and EDMA driven transactions; however, for the EDMA transaction, the EDMA access to the FIFO is transparent.
- The ACCWD bits in the MMC FIFO control register (MMCFIFOCTL) determines the behavior of the FIFO full (FIFOFUL) and FIFO empty (FIFOEMP) status flags in the MMC status register 1 (MMCST1):
  - If ACCWD = 00b:
    - FIFO full is active when the write pointer + 4 > read pointer
    - FIFO empty is active when the write pointer - 4 < read pointer
  - If ACCWD = 01b:
    - FIFO full is active when the write pointer + 3 > read pointer
    - FIFO empty is active when the write pointer - 3 < read pointer
  - If ACCWD = 10b:
    - FIFO full is active when the write pointer + 2 > read pointer
    - FIFO empty is active when the write pointer - 2 < read pointer
  - If ACCWD = 11b:
    - FIFO full is active when the write pointer + 1 > read pointer
    - FIFO empty is active when the write pointer - 1 < read pointer

Figure 25-7. FIFO Operation Diagram



Transmission of data

Step 1: Set FIFO reset
Step 2: Set FIFO direction
Step 3: EDMA driven transaction
Step 4: CPU driven transaction: Fill the FIFO by writing to MMCDXR (only first time) or every 256 or 512-bits transmitted and DXRDYINT interrupt is generated
Step 5: EDMA send xmit data
Step 6: If DXR ready is active, FIFO -> 16-bit DXR

Reception of data

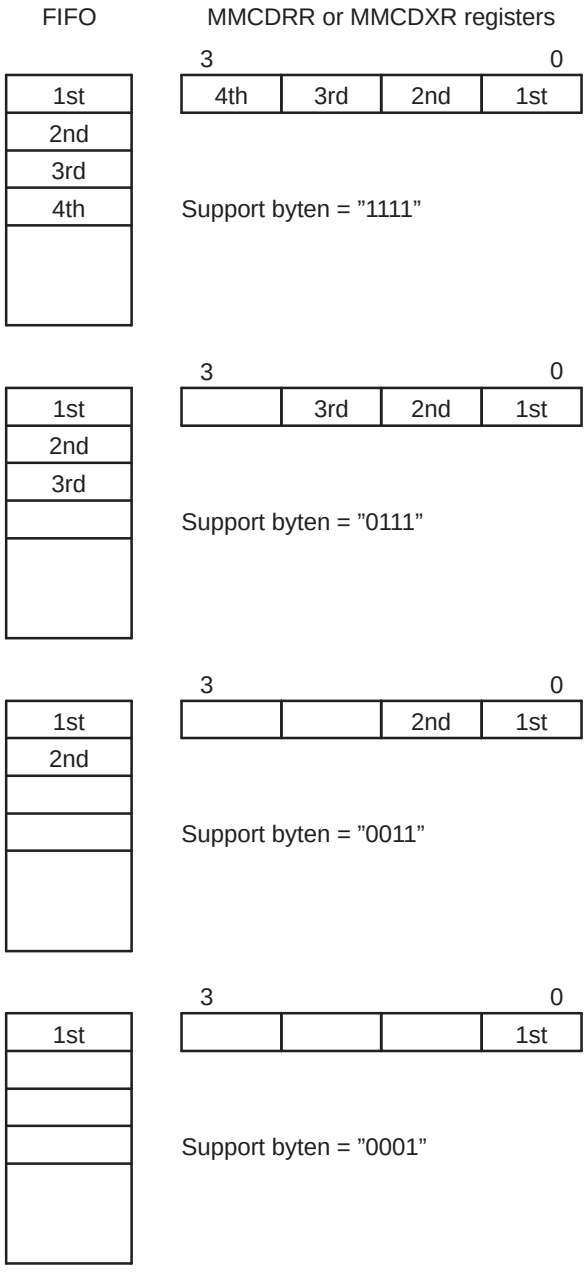
Step 1: Set FIFO reset
Step 2: Set FIFO direction
Step 3: If DRR ready is active, 16-bit DRR -> FIFO
Step 4: EDMA driven transaction
Step 5: DRRDYINT interrupt occur when FIFO every 256 or 512-bits of data received by FIFO
Step 6: EDMA read reception data

Copyright © 2013–2016, Texas Instruments Incorporated

25.2.5 Data Flow in the Data Registers (MMCDRR and MMCDXR)

The CPU or EDMA controller can read 32 bits at a time from the FIFO by reading the MMC data receive register (MMCDRR) and write 32 bits at a time to the FIFO by writing to the MMC data transmit register (MMCDXR). However, since the memory card is an 8-bit device, it transmits or receives one byte at a time. Figure 25-8 shows how the data size is handled by the data registers in little-endian mode.

Figure 25-8. Little-Endian Access to MMCDXR/MMCDRR from the CPU or the EDMA



Copyright © 2013–2016, Texas Instruments Incorporated

25.2.6 FIFO Operation During Card Read Operation

25.2.6.1 EDMA Reads

The FIFO controller manages the activities of reading the data in from the card and issuing EDMA read events. Each time an EDMA read event is issued, an EDMA read request interrupt generates.

Figure 25-9 provides details of the FIFO controllers operation. As data is received from the card, it is read into the FIFO. When the number of bytes of data received is equal to the level set by the FIFOLEV bits in MMCFIFOCTL, an EDMA read event is issued and new EDMA events are disabled until the EDMA is done with the transfer issued by the current event. Data is read from the FIFO by way of MMCDRR. The FIFO controller continues to read in data from the card while checking for the EDMA event to occur or for the FIFO to become full. Once the EDMA event finishes, new EDMA events are enabled. If the FIFO fills up, the FIFO controller stops the MMC/SD controller from reading any more data until the FIFO is no longer full.

An EDMA read event generates when the last data arrives, as determined by the MMC block length register (MMCBLEN) and the MMC number of blocks register (MMCNBLK) settings. This EDMA event flushes all of the data that was read from the card to the FIFO.

Each time an EDMA read event generates, an interrupt (DRRDYINT) generates and the DRRDY bit in the MMC status register 0 (MMCST0) is also set.

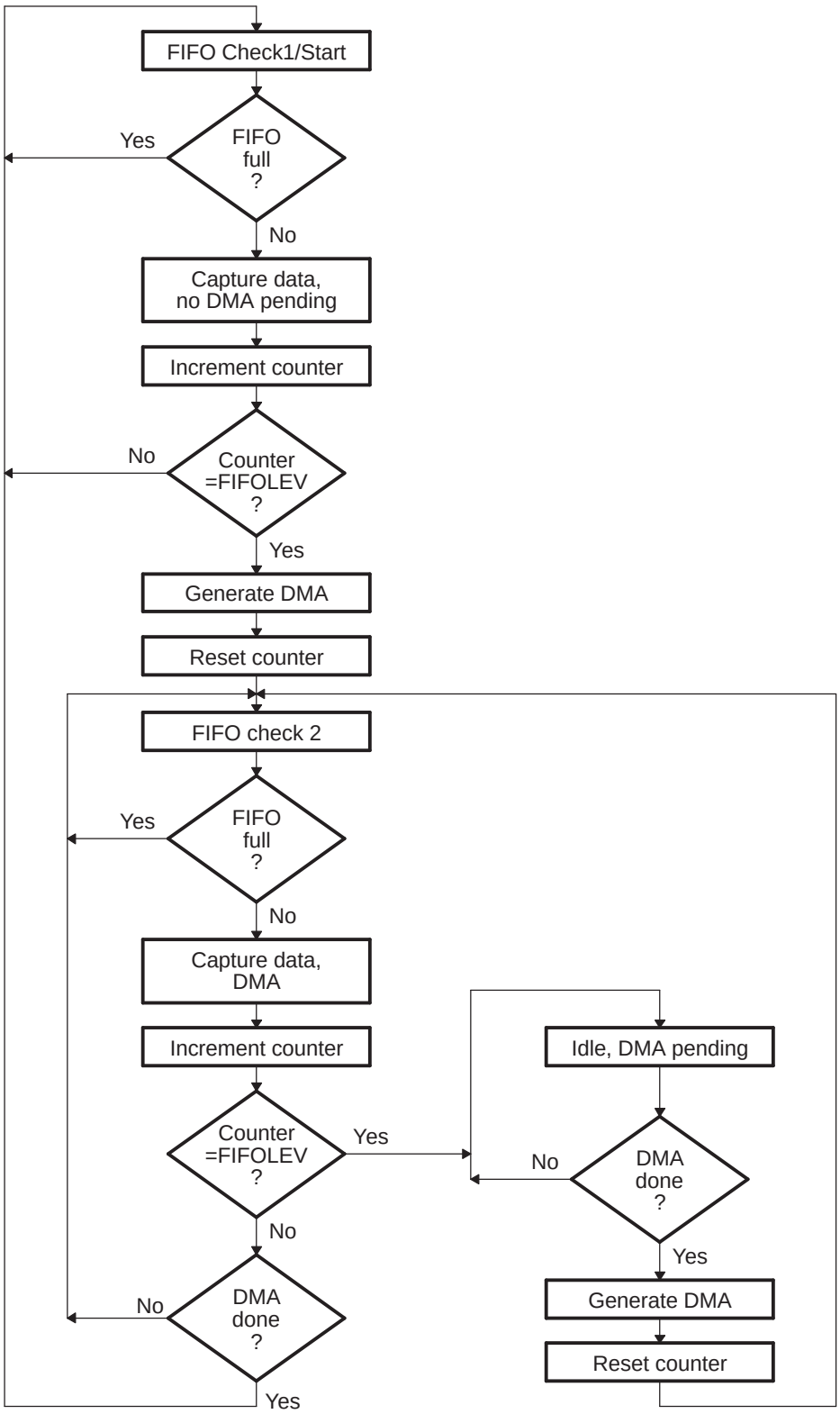
25.2.6.2 CPU Reads

The system CPU can also directly read the card data by reading the MMC data receive register (MMCDRR). The MMC/SD peripheral supports reads that are 1-, 2-, 3-, or 4-bytes wide as, shown in Figure 25-8.

As data is received from the card, it is read into the FIFO. When the number of bytes of data received is equal to the level set by the FIFOLEV bits in MMCFIFOCTL, a DRRDYINT interrupt is issued and the DRRDY bit in the MMC status register 0 (MMCST0) is set. Upon receiving the interrupt, the CPU quickly reads out the bytes received (equal to the level set by the FIFOLEV bits). A DRRDYINT interrupt also generates when the last data arrives as determined by the MMC block length register (MMCBLEN) and the MMC numbers of blocks register (MMCNBLK) settings.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 25-9. FIFO Operation During Card Read Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

25.2.7 FIFO Operation During Card Write Operation

25.2.7.1 EDMA Writes

The FIFO controller manages the activities of accepting data from the CPU or EDMA and passing the data to the MMC/SD controller. The FIFO controller issues EDMA write events as appropriate. Each time an EDMA write event is issued, an EDMA write request interrupt generates. Data is written into the FIFO through MMCDXR. Note that the EDMA access to MMCDXR is transparent.

Figure 25-10 provides details of the FIFO controller's operation. The CPU or EDMA controller writes data into the FIFO. The FIFO passes the data to the MMC/SD controller which manages writing the data to the card. When the number of bytes of data in the FIFO is less than the level set by the FIFOLEV bits in MMCFIFOCTL, an EDMA write event is issued and new EDMA events are disabled. The FIFO controller continues to transfer data to the MMC/SD controller while checking for the EDMA event to finish or for the FIFO to become empty. Once the EDMA event finishes, new EDMA events are enabled. If the FIFO becomes empty, the FIFO controller informs the MMC/SD controller.

Each time an EDMA write event generates, an interrupt (DXRDYINT) generates and the DXRDY bit in the MMC status register 0 (MMCST0) is also set.

25.2.7.2 CPU Writes

The system CPU can also directly write the card data by writing the MMC data transmit register (MMCDXR). The MMC/SD peripheral supports writes that are 1-, 2-, 3-, or 4-bytes wide, as shown in Figure 25-8.

The CPU makes use of the FIFO to transfer data to the card via the MMC/SD controller. The CPU writes the data to be transferred into MMCDXR. As is the case with the EDMA driven transaction, when the number of data in the FIFO is less than the level set by the FIFOLEV bits in MMCFIFOCTL, a DXRDYINT interrupt generates and the DXRDY bit in the MMC status register 0 (MMCST0) is set to signify to the CPU that space is available for new data.

NOTE: When starting the write transaction, the CPU is responsible for getting the FIFO ready to start transferring data by filling up the FIFO with data prior to invoking/posting the write command to the card. Filling up the FIFO is a requirement since no interrupt/event generates at the start of the write transfer.

25.2.8 Reset Considerations

The MMC/SD peripheral has two reset sources: hardware reset and software reset.

25.2.8.1 Software Reset Considerations

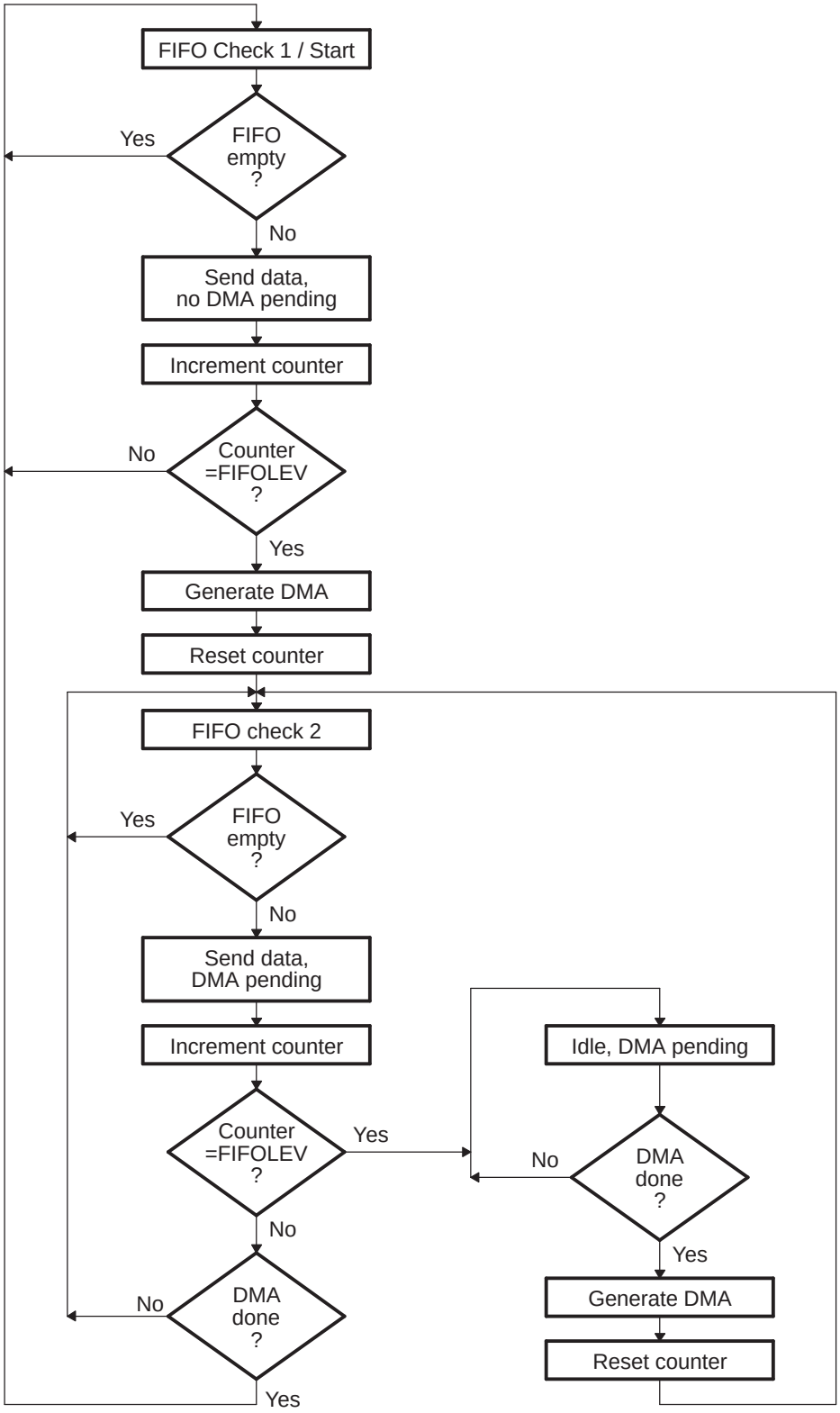
A software reset (such as a reset that the emulator generates) does not cause the MMC/SD controller registers to alter. After a software reset, the MMC/SD controller continues to operate as it was configured prior to the reset.

25.2.8.2 Hardware Reset Considerations

A hardware reset of the processor causes the MMC/SD controller registers to return to their default values after reset.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 25-10. FIFO Operation During Card Write Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

25.2.9 Initialization

25.2.9.1 MMC/SD Controller Initialization

The general procedure for initializing the MMC/SD controller is given in the following steps. Details about the registers or register bit fields to be configured in the MMC/SD mode are in the subsequent subsections.

1. Place the MMC/SD controller in its reset state by setting the CMDRST bit and DATRST bit in the MMC control register (MMCCTL). You can set other bits in MMCCTL after reset.
2. Write the required values to other registers to complete the MMC/SD controller configuration.
3. Clear the CMDRST bit and the DATRST bit in MMCCTL to release the MMC/SD controller from its reset state. It is recommended not to rewrite the values that are written to the other bits of MMCCTL in .
4. Enable the MMCSD_CLK pin so that the memory clock is sent to the memory card by setting the CLKEN bit in the MMC memory clock control register (MMCCLK).

NOTE: The MMC/SD cards require a clock frequency of 400 kHz or less for the card initialization procedure. Make sure that the memory clock confirms this requirement. Once card initialization completes, you can adjust the memory clock up to the lower of the card capabilities or the maximum frequency that is supported.

25.2.9.2 Initializing the MMC Control Register (MMCCTL)

The bits in the MMC control register (MMCCTL) affect the operation of the MMC/SD controller. The subsections that follow help you decide how to initialize each of control register bits.

In the MMC/SD mode, the MMC/SD controller must know how wide the data bus must be for the memory card that is connected. If an MMC card is connected, specify a 1-bit data bus (WIDTH = 0 in MMCCTL); if an SD card is connected, specify a 4-bit data bus (WIDTH = 1 in MMCCTL).

To place the MMC/SD controller in its reset state and disable it, set the CMDRST bit and DATRST bit in MMCCTL. The first step of the MMC/SD controller initialization process is to disable both sets of logic. When initialization is complete, but before you enable the MMCSD_CLK pin, clear the CMDRST bit and DATRST bit in MMCCTL to enable the MMC/SD controller.

25.2.9.3 Initializing the Clock Controller Registers (MMCCLK)

A clock divider in the MMC/SD controller divides-down the function clock to produce the memory clock. Load the divide-down value into the CLKRT bits in the MMC memory clock control register (MMCCLK). The divide-down value is determined by the following equation:

memory clock frequency = function clock frequency/(2 × (CLKRT + 1)), when DIV4 = 0 in MMCCLK

memory clock frequency = function clock frequency/(4 × (CLKRT + 1)), when DIV4 = 1 in MMCCLK

The CLKEN bit in MMCCLK determines whether the memory clock appears on the MMCSD_CLK pin. If you clear the CLKEN to 0, the memory clock is not provided except when required.

25.2.9.4 Initialize the Interrupt Mask Register (MMCIM)

The bits in the MMC interrupt mask register (MMCIM) individually enable or disable the interrupt requests. To enable the associated interrupt request, set the corresponding bit in MMCIM. To disable the associated interrupt request, clear the corresponding bit. Load zeros into the bits that are not used in the MMC/SD mode.

Copyright © 2013–2016, Texas Instruments Incorporated

25.2.9.5 Initialize the Time-Out Registers (MMCTOR and MMCTOD)

Specify the time-out period for responses using the MMC response time-out register (MMCTOR) and the time-out period for reading data using the MMC data read time-out register (MMCTOD).

When the MMC/SD controller sends a command to a memory card, it must often wait for a response. The MMC/SD controller can wait indefinitely or up to 255 memory clock cycles. If you load 0 into MMCTOR, the MMC/SD controller waits indefinitely for a response. If you load a nonzero value into MMCTOR, the MMC/SD controller stops waiting after the specified number of memory clock cycles and then sets a response time-out flag (TOUTRS) in the MMC status register 0 (MMCST0). If you enable the associated interrupt request, the MMC/SD controller also sends an interrupt request to the CPU.

When the MMC/SD controller requests data from a memory card, it can wait indefinitely for that data or it can stop waiting after a programmable number of cycles. If you load 0 into MMCTOD, the MMC/SD controller waits indefinitely. If you load a nonzero value into MMCTOD, the MMC/SD controller waits the specified number of memory clock cycles and then sets a read data time-out flag (TOUTRD) in MMCST0. If you enable the associated interrupt request, the MMC/SD controller also sends an interrupt request to the CPU.

25.2.9.6 Initialize the Data Block Registers (MMCBLEN and MMCNBLK)

Specify the number of bytes in a data block in the MMC block length register (MMCBLEN) and the number of blocks in a multiple-block transfer in the MMC number of blocks register (MMCNBLK).

You must define the size for each block of data transferred between the MMC/SD controller and a memory card in MMCBLEN. The valid size depends on the type of read/write operations. A length of 0 bytes is prohibited.

For multiple-block transfers, you must specify how many blocks of data are to be transferred between the MMC/SD controller and a memory card. You can specify an infinite number of blocks by loading 0 into MMCNBLK. When MMCNBLK = 0, the MMC/SD controller continues to transfer data blocks until the transferring is stopped with a STOP_TRANSMISSION command. To transfer a specific number of blocks, load MMCNBLK with a value from 1 to 65 535.

25.2.9.7 Monitoring Activity in the MMC/SD Mode

This section describes registers and specific register bits that you can use to obtain the status of the MMC/SD controller in the MMC/SD mode. You can determine the status of the MMC/SD controller by reading the bits in the MMC status register 0 (MMCST0) and MMC status register 1 (MMCST1).

25.2.9.7.1 Determining Whether New Data is Available in MMCDRR

The MMC/SD controller sets the DRRDY bit in MMCST0 when the data in the FIFO is greater than the threshold set in the MMC FIFO control register (MMCFIFOCTL). If the interrupt request is enabled (EDRRDY = 1 in MMCIM), the processor is notified of the event by an interrupt. A read of the MMC data receive register (MMCDDR) clears the DRRDY flag.

25.2.9.7.2 Verifying that MMCDXR is Ready to Accept New Data

The MMC/SD controller sets the DXRDY bit in MMCST0 when the amount of data in the FIFO is less than the threshold set in the MMC FIFO control register (MMCFIFOCTL). If the interrupt request is enabled (EDXRDY = 1 in MMCIM), the CPU is notified of the event by an interrupt.

25.2.9.7.3 Checking for CRC Errors

The MMC/SD controller sets the CRCRS, CRCRD, and CRCWR bits in MMCST0 in response to the corresponding CRC errors of command response, data read, and data write. If the interrupt request is enabled (ECRCRS/ECRCRD/ECRCWR = 1 in MMCIM), the CPU is notified of the CRC error by an interrupt.

Copyright © 2013–2016, Texas Instruments Incorporated

25.2.9.7.4 Checking for Time-Out Events

The MMC/SD controller sets the TOUTRS and TOUTRD bits in MMCST0 in response to the corresponding command response or data read time-out event. If the interrupt request is enabled (ETOUTRS/ETOUTRD = 1 in MMCIM), the CPU is notified of the event by an interrupt.

25.2.9.7.5 Determining When a Response/Command is Done

The MMC/SD controller sets the RSPDNE bit in MMCST0 when the response is done; or in the case of commands that do not require a response, when the command is done. If the interrupt request is enabled (ERSPDNE = 1 in MMCIM), the CPU is also notified.

25.2.9.7.6 Determining Whether the Memory Card is Busy

The card sends a busy signal either when waiting for an R1b-type response or when programming the last write data into its flash memory. The MMC/SD controller has two flags to notify you whether the memory card is sending a busy signal. The two flags are complements of each other:

- The BSYDNE flag in MMCST0 is set if the card did not send or is not sending a busy signal when the MMC/SD controller is expecting a busy signal (BSYEXP = 1 in MMCCMD). The interrupt by this bit is enabled by a corresponding interrupt enable bit (EBSYDNE = 1 in MMCIM).
- The BUSY flag in MMCST1 is set when a busy signal is received from the card.

25.2.9.7.7 Determining Whether a Data Transfer is Done

The MMC/SD controller sets the DATDNE bit in MMCST0 when all of the bytes of a data transfer have been transmitted/received. The DATDNE bit is polled to determine when to stop writing to the data transmit register (for a write operation) or when to stop reading from the data receive register (for a read operation). The CPU is also notified of the time-out event by an interrupt if the interrupt request is enabled (EDATDNE = 1 in MMCIM).

25.2.9.7.8 Determining When Last Data has Been Written to Card (SanDisk SD cards)

Some SanDisk brand SD™ cards exhibit a behavior that requires a multiple-block write command to terminate with a STOP (CMD12) command before the data write sequence completes. To enable support of this function, the transfer done interrupt (TRNDNE) is provided. Set the ETRNDNE bit in MMCIM to enable the TRNDNE interrupt. This interrupt is issued when the last byte of data (as defined by MMCNBLK and MMCBLEN) is transferred from the FIFO to the output shift register. The CPU should respond to this interrupt by sending a STOP command to the card. This interrupt differs from DATDNE by timing. DATDNE does not occur until after the CRC and memory programming are complete.

25.2.9.7.9 Checking For a Data Transmit Empty Condition

During transmission, a data value is passed from the MMC data transmit register (MMCDXR) to the data transmit shift register. The data is then passed from the shift register to the memory card one bit at a time. The DXEMP bit in MMCST1 indicates when the shift register is empty.

Typically, the DXEMP bit is not used to control data transfers; rather, it is checked during recovery from an error condition. There is no interrupt associated with the DXEMP bit.

25.2.9.7.10 Checking for a Data Receive Full Condition

During reception, the data receive shift register accepts a data value one bit at a time. The entire value is then passed from the shift register to the MMC data receive register (MMCDRR). The DRFUL bit in MMCST1 indicates that when the shift register is full no new bits can be shifted in from the memory card.

The DRFUL bit is not typically used to control data transfers; rather, it is checked during recovery from an error condition. There is no interrupt associated with the DRFUL bit.

Copyright © 2013–2016, Texas Instruments Incorporated

25.2.9.7.11 Checking the Status of the MMCSD_CLK Pin

Read the CLKSTP bit in MMCST1 to determine whether the memory clock has been stopped on the MMCSD_CLK pin.

25.2.9.7.12 Checking the Remaining Block Count During a Multiple-Block Transfer

During a transfer of multiple data blocks, the MMC number of blocks counter register (MMCNBLC) indicates how many blocks are remaining to be transferred. The MMCNBLC is a read-only register.

25.2.10 Interrupt Support

25.2.10.1 Interrupt Events and Requests

The MMC/SD controller generates the interrupt requests described in Table 25-4. When an interrupt event occurs, its flag bit is set in the MMC status register 0 (MMCST0). If the enable bits corresponding to each flag are set in the MMC interrupt mask register (MMCIM), an interrupt request generates. All such requests are multiplexed to a single MMC/SD interrupt request from the MMC/SD peripheral to the CPU.

The MMC/SD interrupts are part of the maskable CPU interrupts. One CPU interrupt is associated with MMC functions and one CPU interrupt is associated with SD functions (see your device-specific data manual for details). The interrupt service routine (ISR) for the MMC/SD interrupt can determine the event that caused the interrupt by checking the bits in MMCST0. When MMCST0 is read, all register bits automatically clear. During a middle of data transfer, the DXRDY and DRRDY bits are set during every 256-bit or 512-bit transfer, depending on the MMC FIFO control register (MMCFIFOCTL) setting. Performing a write and a read in response to the interrupt generated by the FIFO automatically clears the corresponding interrupt bit/flag.

NOTE: You must be aware that an emulation read of the status register clears the interrupt status flags. To avoid inadvertently clearing the flag, be careful while monitoring MMCST0 via the debugger.

Table 25-4. Description of MMC/SD Interrupt Requests

Interrupt Request	Interrupt Event
TRNDNEINT	**For read operations:** The MMC/SD controller has received the last byte of data (before CRC check).
	For write operations: The MMC/SD controller has transferred the last word of data to the output shift register.
DATEDINT	An edge was detected on the MMCSD_DAT3 pin.
DRRDYINT	MMCDRR is ready to be read (data in FIFO is above threshold).
DXRDYINT	MMCDXR is ready to transmit new data (data in FIFO is less than threshold).
CRCRSINT	A CRC error was detected in a response from the memory card.
CRCRDINT	A CRC error was detected in the data read from the memory card.
CRCWRINT	A CRC error was detected in the data written to the memory card.
TOUTRSINT	A time-out occurred while the MMC controller was waiting for a response to a command.
TOUTRDINT	A time-out occurred while the MMC controller was waiting for the data from the memory card.
RSPDNEINT	**For a command that requires a response:** The MMC controller has received the response without a CRC error.
	For a command that does not require a response: The MMC controller has finished sending the command.
BSYDNEINT	The memory card stops or is no longer sending a busy signal when the MMC controller is expecting a busy signal.
DATDNEINT	**For read operations:** The MMC controller has received data without a CRC error.
	For write operations: The MMC controller has finished sending data.

Copyright © 2013–2016, Texas Instruments Incorporated

25.2.10.2 Interrupt Multiplexing

The interrupts from the MMC/SD peripheral to the CPU are not multiplexed with any other interrupt source.

25.2.11 DMA Event Support

The MMC/SD controller is capable of generating EDMA events for both read and write operations in order to request service from an EDMA controller. Based on the FIFO threshold setting, the EDMA event signals generate every time 256-bit or 512-bit data is transferred from the FIFO.

25.2.12 Power Management

You can put the MMC/SD peripheral in reduced-power modes to conserve power during periods of low activity. The processor power and sleep controller (PSC) controls the power management of the MMC/SD peripheral. The PSC acts as a master controller for power management for all of the peripherals on the device. For detailed information on power management procedures using the PSC, see the *Power and Sleep Controller (PSC)* chapter.

25.2.13 Emulation Considerations

The MMC/SD peripheral is not affected by emulation halt events (such as breakpoints).

Copyright © 2013–2016, Texas Instruments Incorporated

25.3 Procedures for Common Operations

25.3.1 Card Identification Operation

Before the MMC/SD controller starts data transfers to or from memory cards in the MMC/SD native mode, it must first identify how many cards are present on the bus and configure them. For each card that responds to the ALL_SEND_CID broadcast command, the controller reads that card's unique card identification address (CID) and then assigns it a relative address (RCA). This address is much shorter than the CID and the MMC/SD controller uses this address to identify the card in all future commands that involve the card.

Only one card completes the response to ALL_SEND_CID at any one time. The absence of any response to ALL_SEND_CID indicates that all cards have been identified and configured.

NOTE: The following steps assume that the MMC/SD controller is configured to operate in MMC or SD mode, and the memory clock frequency on the MMCSD_CLK pin is set for 400 kHz or less.

The procedure for a card identification operation is issued in open-drain bus mode for both MMC and SD cards.

25.3.1.1 MMC Card Identification Procedure

The MMC card identification procedure is:

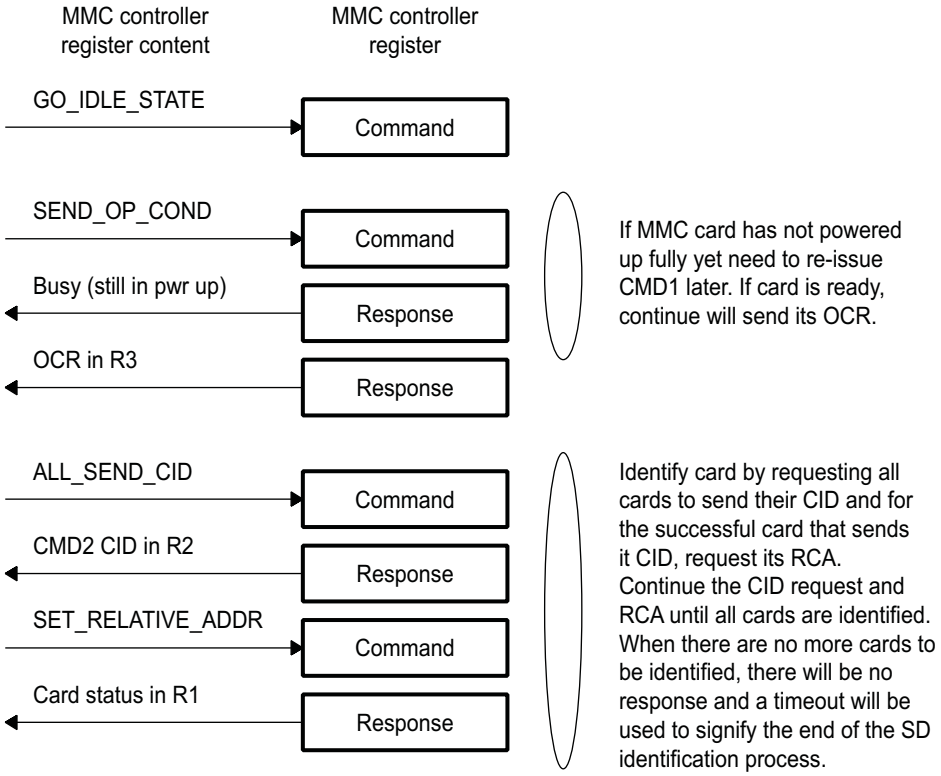
1. Use the MMC command register (MMCCMD) to issue the GO_IDLE_STATE (CMD0) command to the MMC cards. Using MMCCMD to issue the CMD0 command puts all cards (MMC and SD) in the idle state and no response from the cards is expected.
2. Use MMCCMD to issue the SEND_OP_CMD (CMD1) command with the voltage range supported (R3 response, if it is successful; R1b response, if the card is expected to be busy). Using MMCCMD to issue the CMD1 command allows the host to identify and reject cards that do not match the VDD range that the host supports.
3. If the response in Step 2 is R1b (that is, the card is still busy due to power up), then return to Step 2. If the card is not busy, go to Step 4.
4. Use MMCCMD to send the ALL_SEND_CID (CMD2) command (R2 response is expected) to the MMC cards. Using MMCCMD to send the CMD2 command notifies all cards to send their unique card identification (CID) number. There should only be one card that successfully sends its full CID number to the host. The successful card goes into the identification state and does not respond to this command again.
5. Use MMCMD to issue the SET_RELATIVE_ADDR (CMD3) command (R1 response is expected) in order to assign an address that is shorter than the CID number that will be used in the future to address the card in the future data transfer mode.

   NOTE: This command is only addressed to the card that successfully sent its CID number in step 4. This card now goes into standby mode. This card also changes its output drivers from open-drain to push-pull. It stops replying to the CMD2 command, allowing for the identification of other cards.
6. Repeat Step 4 and Step 5 to identify and assign relative addresses to all remaining cards until no card responds to the CMD1 command. No card responding within 5 memory clock cycles indicates that all cards have been identified and the MMC card identification procedure terminates.

The sequence of events in this operation is shown in Figure 25-11.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 25-11. MMC Card Identification Procedure



25.3.1.2 SD Card Identification Procedure

The SD card identification procedure is:

1. Use the MMC command register (MMCCMD) to issue the GO_IDLE_STATE (CMD0) command to the MMC cards. Using MMCMD to issue the CMD0 command puts all cards (MMC and SD) in the idle state and no response from the cards is expected.
2. Use MMCCMD to issue the APP_CMD (CMD55) command (R1 response is expected) to indicate that the command that follows is an application command.
3. Use MMCCMD to send the SD_SEND_OP_COND (ACMD41) command with the voltage range supported (R3 response is expected) to SD cards. Using MMCCMD to send the ACMD41 command allows the host to identify and reject cards that do not match the VDD range that the host supports.
4. Use MMCCMD to send the ALL_SEND_CID (CMD2) command (R2 response is expected) to the MMC cards. Using MMCCMD to send the CMD2 command notifies all cards to send their unique card identification (CID) number. There should only be one card that successfully sends its full CID number to the host. The successful card goes into identification state and does not respond to this command again.
5. Use MMCMD to issue the SEND_RELATIVE_ADDR (CMD3) command (R1 response is expected) in order to ask the card to publish a new relative address for future use to address the card in data transfer mode.

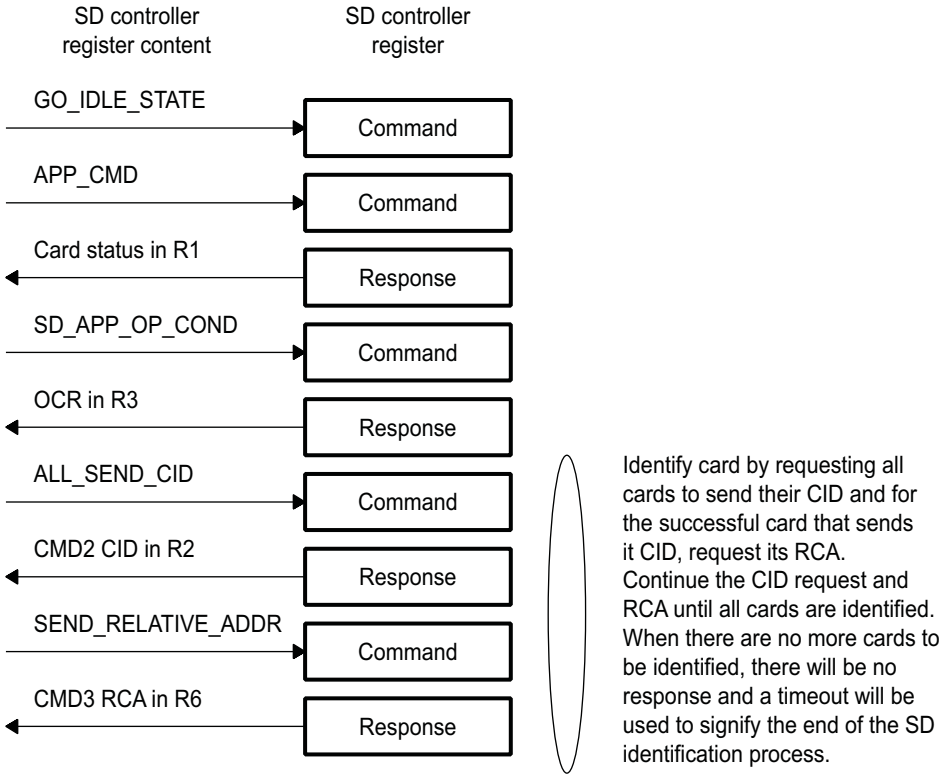
NOTE: This command is only addressed to the card that successfully sent its CID number in step 4. This card now goes into standby mode. This card also changes its output drivers from open-drain to push-pull. It stops replying to the CMD2 command, allowing for the identification of other cards.

Copyright © 2013–2016, Texas Instruments Incorporated

6. Repeat Step 4 and Step 5 to identify and retrieve relative addresses from all remaining SD cards until no card responds to the CMD2 command. No card responding within 5 memory clock cycles indicates that all cards have been identified and the MMC card and the identification procedure terminates.

The sequence of events in this operation is shown in Figure 25-12.

Figure 25-12. SD Card Identification Procedure



Copyright © 2013–2016, Texas Instruments Incorporated

25.3.2 MMC/SD Mode Single-Block Write Operation Using CPU

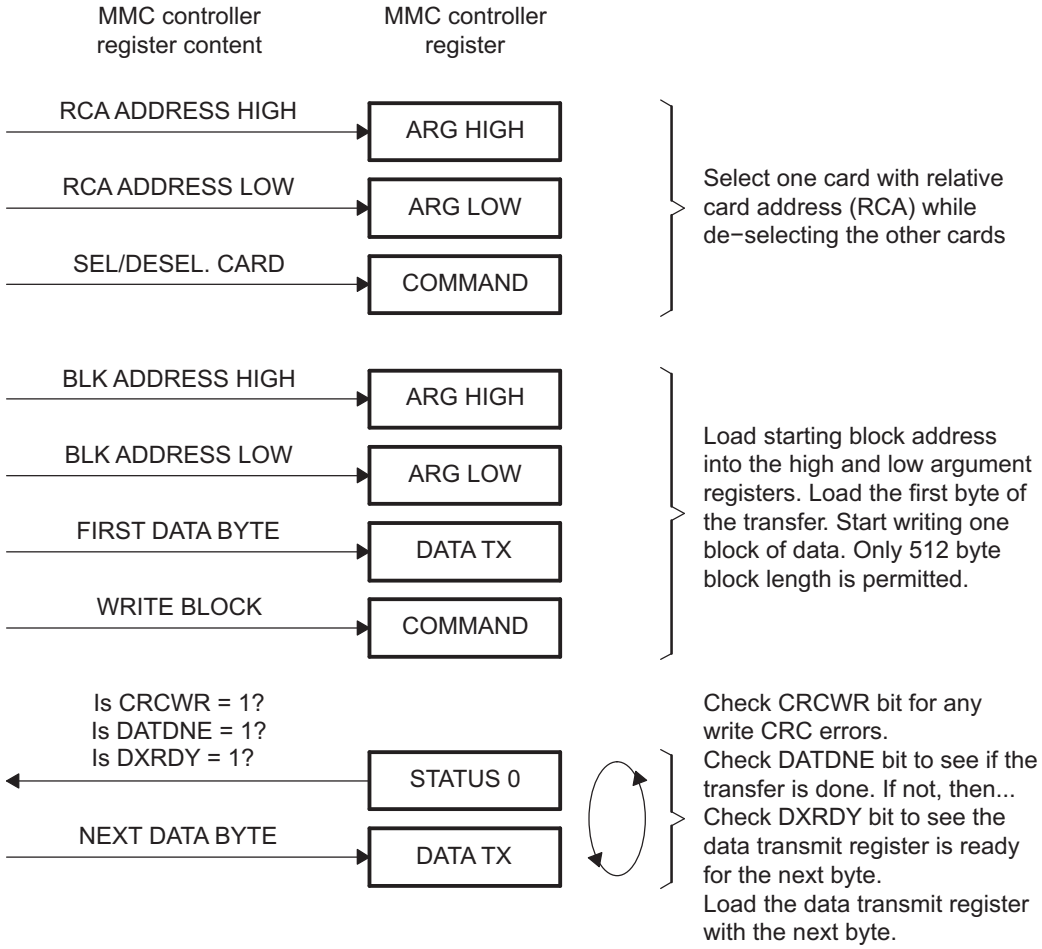
To perform a single-block write, the block length must be 512 bytes and the same length needs to be set in both the MMC/SD controller and the memory card. The procedure for this operation is:

1. Write the card's relative address to the MMC argument registers (MMCARGH and MMCARGL). Load the higher part of the address to MMCARGH and the low part of the address to MMCARGL.
2. Use the MMC command register (MMCCMD) to send the SELECT/DESELECT_CARD broadcast command. This selects the addressed card and deselects the others.
3. Write the destination start address to the MMC argument registers. Load the high part to the MMCARGH register and the low part to MMCARGL.
4. Read the card CSD to determine the card's maximum block length.
5. Use MMCCMD to send the SET_BLOCKLEN command (if the block length is different than the length used in the previous operation). The block length must be a multiple of 512 bytes and less then the maximum block length specified in the CSD.
6. Reset the FIFO (FIFORST bit in MMCFIFOCTL).
7. Set the FIFO direction to transmit (FIFODIR bit in MMCFIFOCTL).
8. Set the access width (ACCWD bits in MMCFIFOCTL).
9. Enable the MMC interrupt.
10. Enable the DXRDYINT interrupt.
11. Write the first 32 bytes of the data block to the data transmit register (MMCDXR).
12. Use MMCCMD to send the WRITE_BLOCK command to the card.
13. Set the DMATRIG bit in MMCCMD to trigger the first data transfer.
14. Wait for the MMC interrupt.
15. Use the MMC status register 0 (MMCST0) to check for errors and the status of the FIFO. If all of the data has not been written and if the FIFO is not full, go to Step 16. If all of the data has been written, stop.
16. Write the next *n* bytes (this depends on the setting of the FIFOLEV bit in MMCFIFOCTL: 0 = 32 bytes, 1 = 64 bytes) of the data block to the MMC data transmit register (MMCDXR) and return to Step 14.

The sequence of events in this operation is shown in Figure 25-13.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 25-13. MMC/SD Mode Single-Block Write Operation



Copyright © 2013–2016, Texas Instruments Incorporated

25.3.3 MMC/SD Mode Single-Block Write Operation Using the EDMA

To perform a single-block write, the block length must be 512 bytes and the same length must be set in both the MMC/SD controller and the card.

The procedure for this operation is as follows:

1. Write the card's relative address to the MMC argument registers (MMCARGH and MMCARGL). Load the high part of the address to MMCARGH and the low part of the address to MMCARGL.
2. Read the card CSD to determine the card's maximum block length.
3. Use the MMC command register (MMCCMD) to send the SET_BLOCKLEN command (if the block length is different than the length used in the previous operation). The block length must be a multiple of 512 bytes and less then the maximum block length specified in the CSD.
4. Reset the FIFO (FIFORST bit in MMCFIFOCTL).
5. Set the FIFO direction to transmit (FIFODIR bit in MMCFIFOCTL).
6. Set the access width (ACCWD bits in MMCFIFOCTL).
7. Set the FIFO threshold (FIFOLEV bit in MMCFIFOCTL).
8. Set up the DMA (DMA size must be greater than or equal to the FIFOLEV setting).
9. Use MMCCMD to send the WRITE _BLOCK command to the card.
10. Set the DMATRIG bit in MMCCMD to trigger the first data transfer.
11. Wait for the DMA sequence to complete or for the DATADNE flag in the MMC status register 0 (MMCST0) to be set.
12. Use MMCST0 to check for errors.

25.3.4 MMC/SD Mode Single-Block Read Operation Using the CPU

To perform a single-block read, the same block length must be set in both the MMC/SD controller and the card.

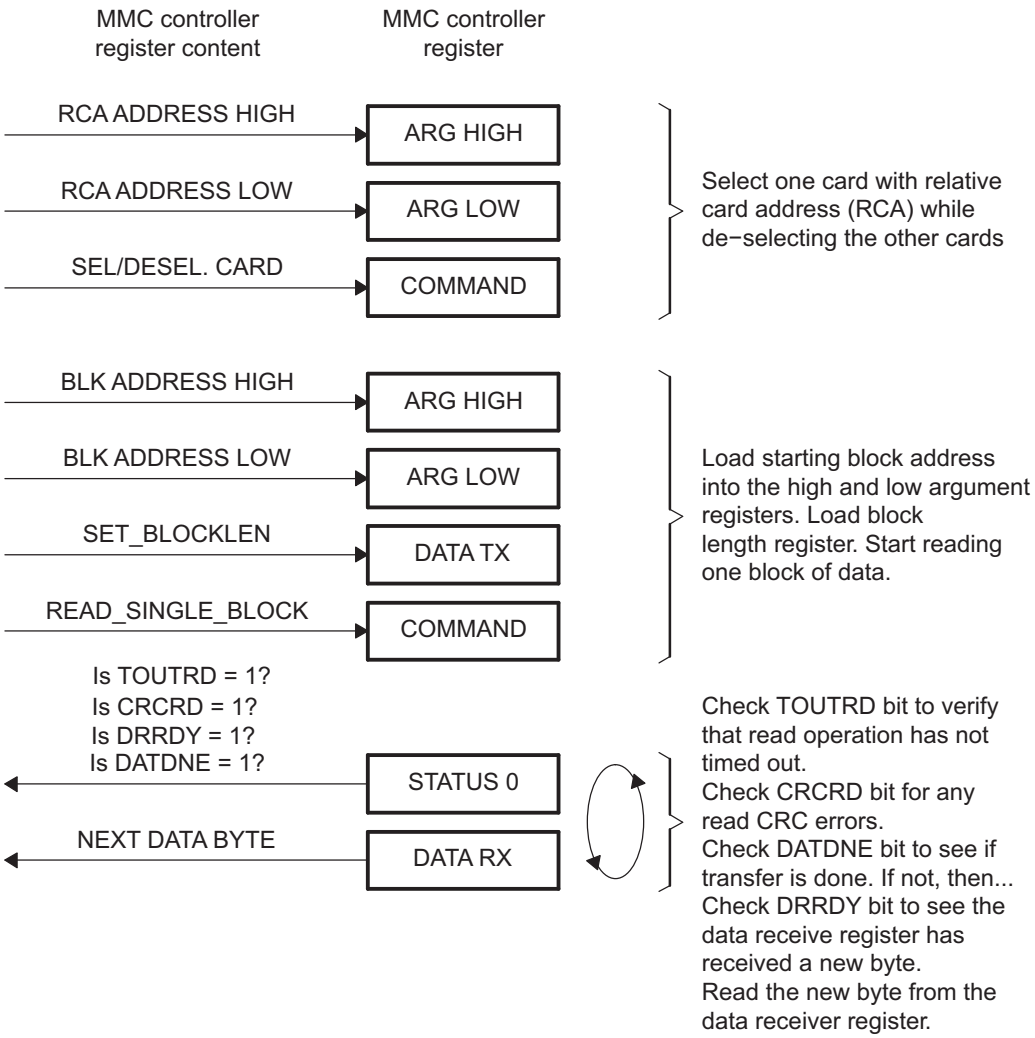
The procedure for this operation is as follows:

1. Write the card's relative address to the MMC argument registers (MMCARGH and MMCARGL). Load the high part of the address to MCARGH and the low part of the address to MMCARGL.
2. Use the MMC command register (MMCCMD) to send the SELECT/DESELECT_CARD broadcast command. This selects the addressed card and deselects the others.
3. Write the source start address to the MMC argument registers. Load the high part to MMCARGH and the low part to MMCARGL.
4. Read card CSD to determine the card's maximum block length.
5. Use MMCCMD to send the SET_BLOCKLEN command (if the block length is different than the length used in the previous operation). The block length must be a multiple of 512 bytes and less then the maximum block length specified in the CSD.
6. Reset the FIFO (FIFORST bit in MMCFIFOCTL).
7. Set the FIFO direction to receive (FIFODIR bit in MMCFIFOCTL).
8. Set the access width (ACCWD bits in MMCFIFOCTL).
9. Set the FIFO threshold (FIFOLEV bit in MMCFIFOCTL).
10. Enable the MMC interrupt.
11. Enable the DRRDYINT interrupt.
12. Use MMCCMD to send the READ_SINGLE_BLOCK command.
13. Set the DMATRIG bit in MMCCMD to trigger the first data transfer.
14. Wait for the MMC interrupt.
15. Use the MMC status register 0 (MMCST0) to check for errors and the status of the FIFO. If the FIFO is not empty, go to Step 16. If the all of the data has been read, stop.
16. Read the next *n* bytes of data (this depends on the setting of the FIFOLEV bit in MMCFIFOCTL: 0 = 32 bytes, 1 = 64 bytes) from the MMC data receive register (MMCDRR) and return to Step 14.

Copyright © 2013–2016, Texas Instruments Incorporated

The sequence of events in this operation is shown in Figure 25-14.

Figure 25-14. MMC/SD Mode Single-Block Read Operation



Copyright © 2013–2016, Texas Instruments Incorporated

25.3.5 MMC/SD Mode Single-Block Read Operation Using EDMA

To perform a single-block read, the same block length needs to be set in both the MMC/SD controller and the card. The procedure for this operation is:

1. Write the card's relative address to the MMC argument registers (MMCARGH and MMCARGL). Load the high part of the address to MMCARGH and the low part of the address to MMCARGL.
2. Read card CSD to determine the card's maximum block length.
3. Use the MMC command register (MMCCMD) to send the SET_BLOCKLEN command (if the block length is different than the length used in the previous operation). The block length must be a multiple of 512 bytes and less then the maximum block length specified in the CSD.
4. Reset the FIFO (FIFORST bit in MMCFIFOCTL).
5. Set the FIFO direction to receive (FIFODIR bit in MMCFIFOCTL).
6. Set the access width (ACCWD bits in MMCFIFOCTL).
7. Set the FIFO threshold (FIFOLEV bit in MMCFIFOCTL).
8. Set up DMA (DMA size needs to be greater than or equal to FIFOLEV setting).
9. Use MMCCMD to send the READ _BLOCK command to the card.
10. Set the DMATRIG bit in MMCCMD to trigger the first data transfer.
11. Wait for DMA sequence to complete.
12. Use the MMC status register 0 (MMCST0) to check for errors.

25.3.6 MMC/SD Mode Multiple-Block Write Operation Using CPU

NOTE: This procedure uses a STOP_TRANSMISSION command to end the block transfer. This assumes that the value in the MMC number of blocks counter register (MMCNBLK) is 0. A multiple-block operation terminates itself if you load MMCNBLK with the exact number of blocks you want transferred.

To perform a multiple-block write, the same block length needs to be set in both the MMC/SD controller and the card.

The procedure for this operation is:

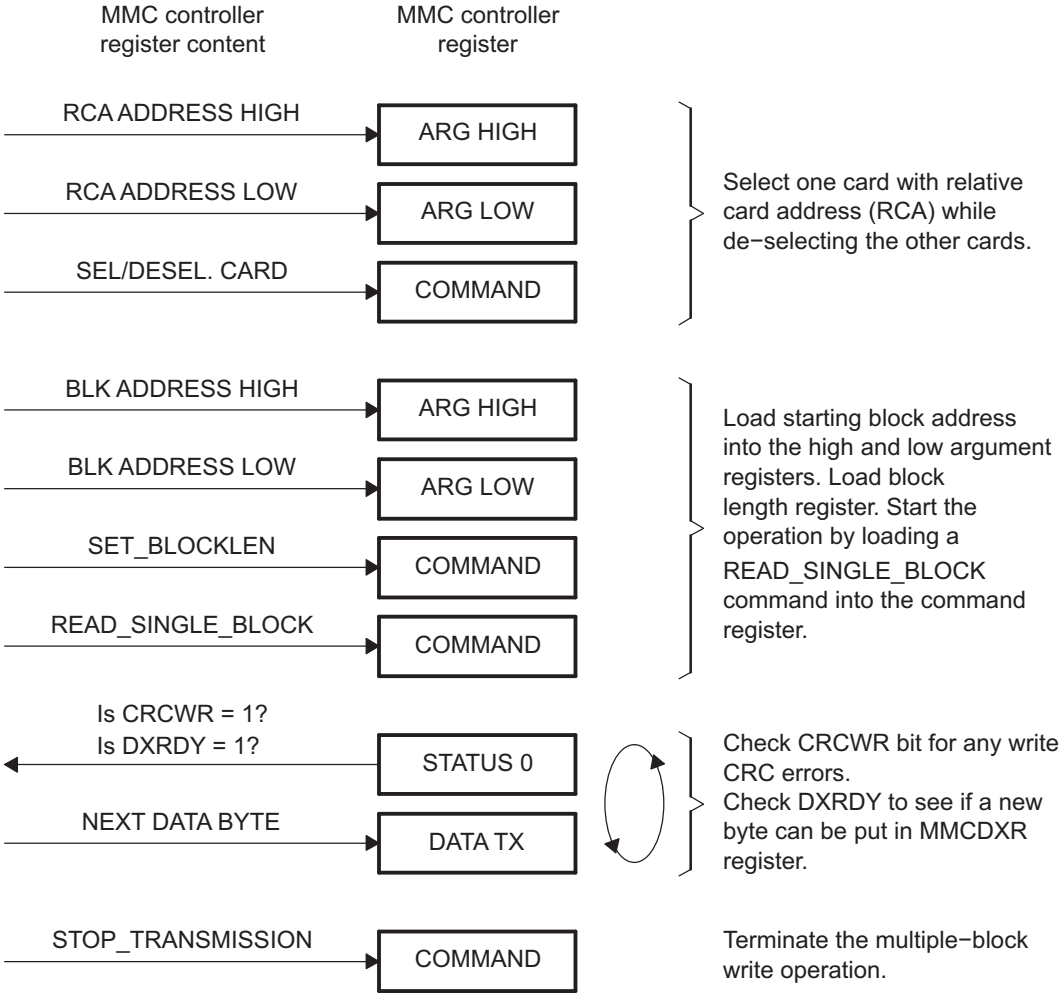
1. Write the card's relative address to the MMC argument registers (MMCARGH and MMCARGL). Load the high part of the address to MMCARGH and the low part of the address to MMCARGL.
2. Read card CSD to determine the card's maximum block length.
3. Use the MMC command register (MMCCMD) to send the SET_BLOCKLEN command (if the block length is different than the length used in the previous operation). The block length must be a multiple of 512 bytes and less then the maximum block length specified in the CSD.
4. Reset the FIFO (FIFORST bit in MMCFIFOCTL).
5. Set the FIFO direction to transmit (FIFODIR bit in MMCFIFOCTL).
6. Set the access width (ACCWD bits in MMCFIFOCTL).
7. Set the FIFO threshold (FIFOLEV bit in MMCFIFOCTL).
8. Enable the MMC interrupt.
9. Enable DXRDYINT interrupt.
10. Write the first 32 bytes of the data block to the MMC data transmit register (MMCDXR).
11. Use MMCCMD to send the WRITE_MULTI_BLOCK command to the card.
12. Set the DMATRIG bit in MMCCMD to trigger the first data transfer.
13. Wait for MMC interrupt.
14. Use the MMC status register 0 (MMCST0) to check for errors and to determine the status of the FIFO. If more bytes are to be written and the FIFO is not full, go to Step 15. If the all of the data has been written, go to Step 16.

Copyright © 2013–2016, Texas Instruments Incorporated

15. Write the next *n* bytes (depends on setting of FIFOLEV in MMCFIFOCTL: 0 = 32 bytes, 1 = 64 bytes) of the data block to MMCDXR, and return to Step 13.
16. Use MMCCMD to send the STOP_TRANSMISSION command.

The sequence of events in this operation is shown in Figure 25-15.

Figure 25-15. MMC/SD Multiple-Block Write Operation



Copyright © 2013–2016, Texas Instruments Incorporated

25.3.7 MMC/SD Mode Multiple-Block Write Operation Using EDMA

To perform a multiple-block write, the same block length needs to be set in both the MMC/SD controller and the card. The procedure for this operation is:

1. Write the card's relative address to the MMC argument registers (MMCARGH and MMCARGL). Load the high part of the address to MMCARGH and the low part of the address to MMCARGL.
2. Read card CSD to determine the card's maximum block length.
3. Use the MMC command register (MMCCMD) to send the SET_BLOCKLEN command (if the block length is different than the length used in the previous operation). The block length must be a multiple of 512 bytes and less then the maximum block length specified in the CSD.
4. Reset the FIFO (FIFORST bit in MMCFIFOCTL).
5. Set the FIFO direction to transmit (FIFODIR bit in MMCFIFOCTL).
6. Set the FIFO threshold (FIFOLEV bit in MMCFIFOCTL).
7. Set the access width (ACCWD bits in MMCFIFOCTL).
8. Set up DMA (DMA size needs to be greater than or equal to FIFOLEV setting).
9. Use MMCCMD to send the WRITE_MULTI_BLOCK command to the card.
10. Set the DMATRIG bit in MMCCMD to trigger the first data transfer.
11. Wait for DMA sequence to complete or the DATADNE flag in the MMC status register 0 (MMCST0) is set.
12. Use MMCST0 to check for errors.
13. Use MMCCMD to send the STOP_TRANSMISSION command.

25.3.8 MMC/SD Mode Multiple-Block Read Operation Using CPU

NOTE: This procedure uses a STOP_TRANSMISSION command to end the block transfer. This assumes that the value in the MMC number of blocks counter register (MMCNBLK) is 0. A multiple-block operation terminates itself if you load MMCNBLK with the exact number of blocks you want transferred.

To perform a multiple-block read, the same block length needs to be set in both the MMC/SD controller and the card.

The procedure for this operation is:

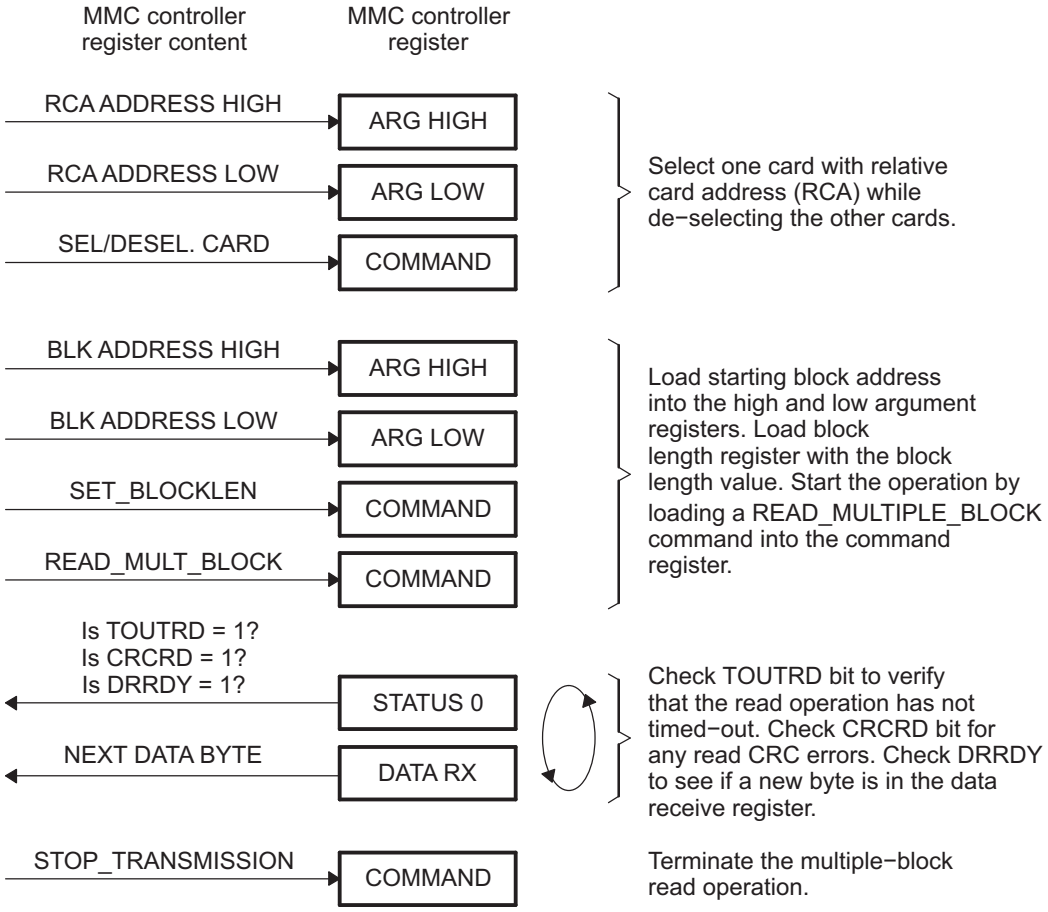
1. Write the card's relative address to the MMC argument registers (MMCARGH and MMCARGL). Load the high part of the address to MMCARGH and the low part of the address to MMCARGL.
2. Read card CSD to determine the card's maximum block length.
3. Use the MMC command register (MMCCMD) to send the SET_BLOCKLEN command (if the block length is different than the length used in the previous operation). The block length must be a multiple of 512 bytes and less then the maximum block length specified in the CSD.
4. Reset the FIFO (FIFORST bit in MMCFIFOCTL).
5. Set the FIFO direction to receive (FIFODIR bit in MMCFIFOCTL).
6. Set FIFO threshold (FIFOLEV bit in MMCFIFOCTL).
7. Set the access width (ACCWD bits in MMCFIFOCTL).
8. Enable the MMC interrupt.
9. Enable DRRDYINT interrupt.
10. Use MMCCMD to send the READ_MULT_BLOCKS command.
11. Set the DMATRIG bit in MMCCMD to trigger the first data transfer.
12. Wait for MMC interrupt.
13. Use the MMC status register 0 (MMCST0) to check for errors and to determine the status of the FIFO. If FIFO is not empty and more bytes are to be read, go to Step 14. If the all of the data has been read, go to Step 15.

Copyright © 2013–2016, Texas Instruments Incorporated

14. Read *n* bytes (depends on setting of FIFOLEV in MMCFIFOCTL: 0 = 32 bytes, 1 = 64 bytes) of data from the MMC data receive register (MMCDRR) and return to Step 10.
15. Use MMCCMD to send the STOP_TRANSMISSION command.

The sequence of events in this operation is shown in Figure 25-16.

Figure 25-16. MMC/SD Mode Multiple-Block Read Operation



Copyright © 2013–2016, Texas Instruments Incorporated

25.3.9 MMC/SD Mode Multiple-Block Read Operation Using EDMA

To perform a multiple-block read, the same block length must be set in both the MMC/SD controller and the card.

The procedure for this operation is as follows:

1. Write the card's relative address to the MMC argument registers (MMCARGH and MMCARGL). Load the high part of the address to MMCARGH and the low part of the address to MMCARGL.
2. Read card CSD to determine the card's maximum block length.
3. Use the MMC command register (MMCCMD) to send the SET_BLOCKLEN command (if the block length is different than the length used in the previous operation). The block length must be a multiple of 512 bytes and less then the maximum block length specified in the CSD.
4. Reset the FIFO (FIFORST bit in MMCFIFOCTL).
5. Set the FIFO direction to receive (FIFODIR bit in MMCFIFOCTL).
6. Set the FIFO threshold (FIFOLEV bit in MMCFIFOCTL).
7. Set the access width (ACCWD bits in MMCFIFOCTL).
8. Set up DMA (DMA size needs to be greater than or equal to FIFOLEV setting).
9. Use MMCCMD to send the READ_MULTI_BLOCK command to the card.
10. Set the DMATRIG bit in MMCCMD to trigger the first data transfer.
11. Wait for DMA sequence to complete.
12. Use the MMC status register 0 (MMCST0) to check for errors.
13. Use MMCCMD to send the STOP_TRANSMISSION command.

25.3.10 SDIO Card Function

To support the SDIO card, the following features are available in the MMC/SD controller:

- Read wait operation request
- Interrupt to CPU at the start of read wait operation
- Interrupt to CPU at the detection of SDIO interrupt

When in 1-bit mode and the transfer clock (memory clock) is off, this peripheral cannot recognize an SDIO interrupt from SD_DATA1 line. Two options are available to deal with this situation:

1. Do not turn off the memory clock in 1-bit mode. The clock is enabled by the CLKEN bit in the MMC memory clock control register (MMCCLK).
2. If the memory clock needs to be turned off, physically connect a GPIO signal and SD_DATA1, and use the GPIO as an external interrupt input. When the memory clock is enabled, disable the GPIO interrupt and enable the SDIO interrupt. When the memory clock is disabled, enable the GPIO interrupt and disable the SDIO interrupt by software.

25.3.10.1 SDIO Control Register (SDIOCTL)

The SDIO card control register (SDIOCTL) is used to configure the read wait operation using the SD_DATA2 line.

25.3.10.2 SDIO Status Register 0 (SDIOST0)

The SDIO card status register 0 (SDIOST0) is used to check the status of the SD_DATA1 signal, check the status of being in an interrupt period, or check the status of being in a read wait operation.

Copyright © 2013–2016, Texas Instruments Incorporated

25.3.10.3 SDIO Interrupt Control Registers (SDIOIEN, SDIOIST)

The SDIO card controller issues an interrupt to the CPU when the read wait operation starts or when an SDIO interrupt is detected on the SD_DATA1 line.

Interrupt flags of each case are checked with the SDIO interrupt status register (SDIOIST). To issue an actual interrupt to the CPU, enabling each interrupt in the SDIO interrupt enable register (SDIOIEN) is required.

When both interrupts are enabled, they are both reported to the CPU as an interrupt (whether one or both occurred). The interrupt(s) that occurred are determined by reading SDIOIST.

25.4 Registers

Table 25-5 lists the memory-mapped registers for the multimedia card/secure digital (MMC/SD) card controller. See your device-specific data manual for the memory address of these registers.

Table 25-5. Multimedia Card/Secure Digital (MMC/SD) Card Controller Registers

Offset	Acronym	Register Description	Section
0h	MMCCTL	MMC Control Register	Section 25.4.1
4h	MMCCLK	MMC Memory Clock Control Register	Section 25.4.2
8h	MMCST0	MMC Status Register 0	Section 25.4.3
Ch	MMCST1	MMC Status Register 1	Section 25.4.4
10h	MMCIM	MMC Interrupt Mask Register	Section 25.4.5
14h	MMCTOR	MMC Response Time-Out Register	Section 25.4.6
18h	MMCTOD	MMC Data Read Time-Out Register	Section 25.4.7
1Ch	MMCBLEN	MMC Block Length Register	Section 25.4.8
20h	MMCNBLK	MMC Number of Blocks Register	Section 25.4.9
24h	MMCNBLC	MMC Number of Blocks Counter Register	Section 25.4.10
28h	MMCDRR	MMC Data Receive Register	Section 25.4.11
2Ch	MMCDXR	MMC Data Transmit Register	Section 25.4.12
30h	MMCCMD	MMC Command Register	Section 25.4.13
34h	MMCARGHL	MMC Argument Register	Section 25.4.14
38h	MMCRSP01	MMC Response Register 0 and 1	Section 25.4.15
3Ch	MMCRSP23	MMC Response Register 2 and 3	Section 25.4.15
40h	MMCRSP45	MMC Response Register 4 and 5	Section 25.4.15
44h	MMCRSP67	MMC Response Register 6 and 7	Section 25.4.15
48h	MMCDRSP	MMC Data Response Register	Section 25.4.16
50h	MMCCIDX	MMC Command Index Register	Section 25.4.17
64h	SDIOCTL	SDIO Control Register	Section 25.4.18
68h	SDIOST0	SDIO Status Register 0	Section 25.4.19
6Ch	SDIOIEN	SDIO Interrupt Enable Register	Section 25.4.20
70h	SDIOIST	SDIO Interrupt Status Register	Section 25.4.21
74h	MMCFIFOCTL	MMC FIFO Control Register	Section 25.4.22

Copyright © 2013–2016, Texas Instruments Incorporated

25.4.1 MMC Control Register (MMCCTL)

The MMC control register (MMCCTL) is used to enable or configure various modes of the MMC controller. Set or clear the DATRST and CMDRST bits at the same time to reset or enable the MMC controller.

The MMC control register (MMCCTL) is shown in Figure 25-17 and described in Table 25-6.

Figure 25-17. MMC Control Register (MMCCTL)

31	16
Reserved	
R-0	

15	11	10	9	8
Reserved		PERMDX	PERMDR	WIDTH1
R-0		R/W-0	R/W-0	R-0

7	6	5	3	2	1	0
DATEG		Reserved		WIDTH0	CMDRST	DATRST
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 25-6. MMC Control Register (MMCCTL) Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	0	Reserved
10	PERMDX		Endian select when writing.
		0	Little endian is selected.
		1	Big endian is selected.
9	PERMDR		Endian select when reading.
		0	Little endian is selected.
		1	Big endian is selected.
8	WIDTH1	0-3h	Data bus width 1 (MMC mode only). Used in conjunction with the WIDTH0 bit.
		0	Data bus has 1 bit (only MMCSD_DAT0 is used).
		1h	Data bus has 4 bits (only MMCSD_DAT0-3 are used).
		2h	Data bus has 8 bits (MMCSD_DAT0-7 are used).
		3h	Reserved
7-6	DATEG	0-3h	MMCSD_DAT3 edge detection select.
		0	MMCSD_DAT3 edge detection is disabled.
		1h	MMCSD_DAT3 rising-edge detection is enabled.
		2h	MMCSD_DAT3 falling-edge detection is enabled.
		3h	MMCSD_DAT3 rising-edge and falling-edge detections are enabled.
5-3	Reserved	0	Reserved
2	WIDTH0	0-3h	Data bus width 0 (MMC mode only). Used in conjunction with the WIDTH1 bit.
1	CMDRST		CMD logic reset.
		0	CMD line portion is enabled.
		1	CMD line portion is disabled and in reset state.
0	DATRST		DAT logic reset.
		0	DAT line portion is enabled.
		1	DAT line portion is disabled and in reset state.

Copyright © 2013–2016, Texas Instruments Incorporated

25.4.2 MMC Memory Clock Control Register (MMCCLK)

The MMC memory clock control register (MMCCLK) is used to:

- Select whether the MMCSD_CLK pin is enabled or disabled (CLKEN bit).
- Select how much the function clock is divided-down to produce the memory clock (CLKRT bits). When the MMCSD_CLK pin is enabled, the MMC controller drives the memory clock on this pin to control the timing of communications with attached memory cards. For more details about clock generation, see Section 25.2.1.

The MMC memory clock control register (MMCCLK) is shown in Figure 25-18 and described in Table 25-7.

Figure 25-18. MMC Memory Clock Control Register (MMCCLK)

31	16
Reserved	
R-0	

15	10	9	8	7	0
Reserved		DIV4	CLKEN	CLKRT	
R-0		R/W-0	R/W-0	R/W−FFh	

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

Table 25-7. MMC Memory Clock Control Register (MMCCLK) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reserved
9	DIV4		DIV4 option
		0	MMC clock = function clock/2 × (CLKRT + 1)
		1	MMC clock = function clock/4 × (CLKRT + 1)
8	CLKEN		MMCSD_CLK pin enable
		0	MMCSD_CLK pin is disabled and fixed low
		1	The MMCSD_CLK pin is enabled; it shows the memory clock signal.
7-0	CLKRT	0−FFh	Clock rate. Use this field to set the divide-down value for the memory clock. The function clock is divided down as follows to produce the memory clock: memory clock frequency = function clock frequency/(2 × (CLKRT + 1))

Copyright © 2013–2016, Texas Instruments Incorporated

25.4.3 MMC Status Register 0 (MMCST0)

The MMC status register 0 (MMCST0) records specific events or errors. The transition from 0 to 1 on each bit in MMCST0 can cause an interrupt signal to be sent to the CPU. If an interrupt is desired, set the corresponding interrupt enable bit in the MMC interrupt mask register (MMCIM).

In most cases, when a status bit is read, it is cleared. The two exceptions are the DRRDY bit and the DXRDY bit; these bits are cleared only in response to the functional events described for them in Table 25-8, or in response to a hardware reset.

The MMC status register 0 (MMCST0) is shown in Figure 25-19 and described in Table 25-8.

NOTE: 1) As the command portion and the data portion of the MMC/SD controller are independent, any command such as CMD0 (GO_IDLE_STATE) or CMD12 (STOP_TRANSMISSION) can be sent to the card, even during block transfer. In this situation, the data portion detects this and waits, releasing the busy state only when the command sent was R1b (to be specific, command with BSYEXP bit), otherwise it continues transferring data.

2) Bit 12 (TRNDNE) indicates that the last byte of a transfer has been completed. Bit 0 (DATDNE) occurs at end of a transfer, but not until the CRC check and programming has completed.

Figure 25-19. MMC Status Register 0 (MMCST0)

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		CCS	TRNDNE	DATED	DRRDY	DXRDY	Reserved
R-0		R-0	R-0	RC-0	R-0	R-1	R-0

7	6	5	4	3	2	1	0
CRCRS	CRCRD	CRCWR	TOUTRS	TOUTRD	RSPDNE	BSYDNE	DATDNE
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; RC = Cleared to 0 when read; -*n* = value after reset

Table 25-8. MMC Status Register 0 (MMCST0) Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Reserved
13	CCS		Command completion signal
		0	Command completion signal is not completed.
		1	Command completion signal is completed.
12	TRNDNE		Transfer done.
		0	No data transfer is done.
		1	Data transfer of specified length is done.
11	DATED		MMCSD_DAT3 edge detected. DATED is cleared when read by CPU.
		0	An MMCSD_DAT3 edge has not been detected.
		1	An MMCSD_DAT3 edge has been detected.
10	DRRDY		Data receive ready. DRRDY is cleared to 0 when the DAT logic is reset (DATRST = 1 in MMCCTL), when a command is sent with data receive/transmit clear (DCLR = 1 in MMCCMD), or when data is read from the MMC data receive register (MMCDRR).
		0	MMCDRR is not ready.
		1	MMCDRR is ready. New data has arrived and can be read by the CPU or by the DMA controller.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 25-8. MMC Status Register 0 (MMCST0) Field Descriptions (continued)

Bit	Field	Value	Description
9	DXRDY		Data transmit ready. DXRDY is set to 1 when the DAT logic is reset (DATRST = 1 in MMCCTL), when a command is sent with data receive/transmit clear (DCLR = 1 in MMCCMD), or when data is written to the MMC data transmit register (MMCDXR).
		0	MMCDXR is not ready.
		1	MMCDXR is ready. The data in MMCDXR has been transmitted; MMCDXR can accept new data from the CPU or from the DMA controller.
8	Reserved	0	Reserved
7	CRCRS		Response CRC error.
		0	A response CRC error has not been detected.
		1	A response CRC error has been detected.
6	CRCRD		Read-data CRC error.
		0	A read-data CRC error has not been detected.
		1	A read-data CRC error has been detected.
5	CRCWR		Write-data CRC error.
		0	A write-data CRC error has not been detected.
		1	A write-data CRC error has been detected.
4	TOUTRS		Response time-out event.
		0	A response time-out event has not occurred.
		1	A time-out event has occurred while the MMC controller was waiting for a response to a command.
3	TOUTRD		Read-data time-out event.
		0	A read-data time-out event has not occurred.
		1	A time-out event has occurred while the MMC controller was waiting for data.
2	RSPDNE		Command/response done.
		0	No receiving response is done.
		1	Response successfully has received or command has sent without response.
1	BSYDNE		Busy done.
		0	No busy releasing is done.
		1	Released from busy state or expected busy is not detected.
0	DATDNE		Data done
		0	The data has not been fully transmitted.
		1	The data has been fully transmitted.

Copyright © 2013–2016, Texas Instruments Incorporated

25.4.4 MMC Status Register 1 (MMCST1)

The MMC status register 1 (MMCST1) records specific events or errors. There are no interrupts associated with these events or errors.

The MMC status register 1 (MMCST1) is shown in Figure 25-20 and described in Table 25-9.

Figure 25-20. MMC Status Register 1 (MMCST1)

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	FIFOFUL	FIFOEMP	DAT3ST	DRFUL	DXEMP	CLKSTP	BUSY
R-0	R-0	R-0	R-0	R-0	R-0	R-1	R-0

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

Table 25-9. MMC Status Register 1 (MMCST1) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reserved
6	FIFOFUL		FIFO is full.
		0	FIFO is not full.
		1	FIFO is full.
5	FIFOEMP		FIFO is empty.
		0	FIFO is not empty.
		1	FIFO is empty.
4	DAT3ST		MMCSD_DAT3 status.
		0	The signal level on the MMCSD_DAT3 pin is a logic-low level.
		1	The signal level on the MMCSD_DAT3 pin is a logic-high level.
3	DRFUL		Data receive register (MMCDRR) is full.
		0	A data receive register full condition is not detected. The data receive shift register is not full.
		1	A data receive register full condition is detected. The data receive shift register is full. No new bits can be shifted in from the memory card.
2	DXEMP		Data transmit register (MMCDXR) is empty.
		0	A data transmit register empty condition is not detected. The data transmit shift register is not empty.
		1	A data transmit register empty condition is detected. The data transmit shift register is empty. No bits are available to be shifted out to the memory card.
1	CLKSTP		Clock stop status.
		0	MMCSD_CLK is active. The memory clock signal is being driven on the pin.
		1	MMCSD_CLK is held low because of a manual stop (CLKEN = 0 in MMCCLK), receive shift register is full, or transmit shift register is empty.
0	BUSY		Busy.
		0	No busy signal is detected.
		1	A busy signal is detected (the memory card is busy).

Copyright © 2013–2016, Texas Instruments Incorporated

25.4.5 MMC Interrupt Mask Register (MMCIM)

The MMC interrupt mask register (MMCIM) is used to enable (bit = 1) or disable (bit = 0) status interrupts. If an interrupt is enabled, the transition from 0 to 1 of the corresponding interrupt bit in the MMC status register 0 (MMCST0) can cause an interrupt signal to be sent to the CPU.

The MMC interrupt mask register (MMCIM) is shown in Figure 25-21 and described in Table 25-10.

Figure 25-21. MMC Interrupt Mask Register (MMCIM)

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		ECCS	ETRNDNE	EDATED	EDRRDY	EDXRDY	Reserved
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

7	6	5	4	3	2	1	0
ECRCRS	ECRCRD	ECRCWR	ETOUTRS	ETOUTRD	ERSPDNE	EBSYDNE	EDATDNE
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 25-10. MMC Interrupt Mask Register (MMCIM) Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Reserved
13	ECCS		Command completion signal interrupt enable.
		0	Command completion signal interrupt is disabled.
		1	Command completion signal interrupt is enabled.
12	ETRNDNE		Transfer done (TRNDNE) interrupt enable.
		0	Transfer done interrupt is disabled.
		1	Transfer done interrupt is enabled.
11	EDATED		MMCSD_DAT3 edge detect (DATED) interrupt enable.
		0	MMCSD_DAT3 edge detect interrupt is disabled.
		1	MMCSD_DAT3 edge detect interrupt is enabled.
10	EDRRDY		Data receive register ready (DRRDY) interrupt enable.
		0	Data receive register ready interrupt is disabled.
		1	Data receive register ready interrupt is enabled.
9	EDXRDY		Data transmit register (MMCDXR) ready interrupt enable.
		0	Data transmit register ready interrupt is disabled.
		1	Data transmit register ready interrupt is enabled.
8	Reserved	0	Reserved
7	ECRCRS		Response CRC error (CRCRS) interrupt enable.
		0	Response CRC error interrupt is disabled.
		1	Response CRC error interrupt is enabled.
6	ECRCRD		Read-data CRC error (CRCRD) interrupt enable.
		0	Read-data CRC error interrupt is disabled.
		1	Read-data CRC error interrupt is enabled.
5	ECRCWR		Write-data CRC error (CRCWR) interrupt enable.
		0	Write-data CRC error interrupt is disabled.
		1	Write-data CRC error interrupt is disabled.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 25-10. MMC Interrupt Mask Register (MMCIM) Field Descriptions (continued)

Bit	Field	Value	Description
4	ETOUTRS		Response time-out event (TOUTRS) interrupt enable.
		0	Response time-out event interrupt is disabled.
		1	Response time-out event interrupt is enabled.
3	ETOUTRD		Read-data time-out event (TOUTRD) interrupt enable.
		0	Read-data time-out event interrupt is disabled.
		1	Read-data time-out event interrupt is enabled.
2	ERSPDNE		Command/response done (RSPDNE) interrupt enable.
		0	Command/response done interrupt is disabled.
		1	Command/response done interrupt is enabled.
1	EBSYDNE		Busy done (BSYDNE) interrupt enable.
		0	Busy done interrupt is disabled.
		1	Busy done interrupt is enabled.
0	EDATDNE		Data done (DATDNE) interrupt enable.
		0	Data done interrupt is disabled.
		1	Data done interrupt is enabled.

Copyright © 2013–2016, Texas Instruments Incorporated

25.4.6 MMC Response Time-Out Register (MMCTOR)

The MMC response time-out register (MMCTOR) defines how long the MMC controller waits for a response from a memory card before recording a time-out condition in the TOUTRS bit of the MMC status register 0 (MMCST0). If the corresponding ETOUTRS bit in the MMC interrupt mask register (MMCIM) is set, an interrupt is generated when the TOUTRS bit is set in MMCST0. If a memory card should require a longer time-out period than MMCTOR can provide, a software time-out mechanism can be implemented.

The MMC response time-out register (MMCTOR) is shown in Figure 25-22 and described in Table 25-11.

Figure 25-22. MMC Response Time-Out Register (MMCTOR)

31	18	17	16
Reserved		TOD_25_16	
R-0		R/W-0	

15	8	7	0
TOD_25_16		TOR	
R/W-0		R/W-0	

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

Table 25-11. MMC Response Time-Out Register (MMCTOR) Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-8	TOD_25_16	0-3FFh	Data read time-out count upper 10 bits. Used in conjunction with the TOD_15_0 bits in MMCTOD to form a 26-bit count (1 CLK clock cycle to 67 108 863 CLK clock cycles). See MMCTOD (Section 25.4.7).
		0	No time out
		1h-3FF FFFFh	1 CLK clock cycle to 67 108 863 CLK clock cycles
7-0	TOR	0-FFh	Time-out count for response.
		0	No time out
		1h-FFh	1 CLK clock cycle to 255 CLK clock cycles

Copyright © 2013–2016, Texas Instruments Incorporated

25.4.7 MMC Data Read Time-Out Register (MMCTOD)

The MMC data read time-out register (MMCTOD) defines how long the MMC controller waits for the data from a memory card before recording a time-out condition in the TOUTRD bit of the MMC status register 0 (MMCST0). If the corresponding ETOUTRD bit in the MMC interrupt mask register (MMCIM) is set, an interrupt is generated when the TOUTRD bit is set in MMCST0. If a memory card should require a longer time-out period than MMCTOD can provide, a software time-out mechanism can be implemented.

The MMC data read time-out register (MMCTOD) is shown in Figure 25-23 and described in Table 25-12.

Figure 25-23. MMC Data Read Time-Out Register (MMCTOD)

31	16
Reserved	
R-0	

15	0
TOD_15_0	
R/W-0	

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

Table 25-12. MMC Data Read Time-Out Register (MMCTOD) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	TOD_15_0	0-FFFFh	Data read time-out count. Used in conjunction with the TOD_25_16 bits in MMCTOR to form a 26-bit count (1 CLK clock cycle to 67 108 863 CLK clock cycles). See MMCTOR (Section 25.4.6).
		0	No time out
		1h-3FF FFFFh	1 CLK clock cycle to 67 108 863 CLK clock cycles

Copyright © 2013–2016, Texas Instruments Incorporated

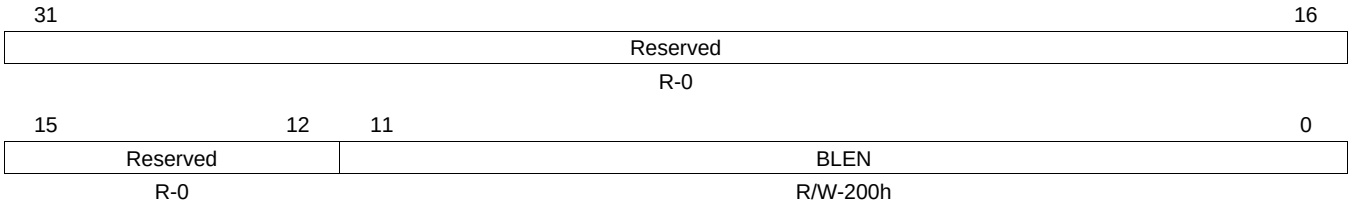
25.4.8 MMC Block Length Register (MMCBLEN)

The MMC block length register (MMCBLEN) specifies the data block length in bytes. This value must match the block length setting in the memory card.

The MMC block length register (MMCBLEN) is shown in Figure 25-24 and described in Table 25-13.

NOTE: The BLEN bits value must be the same as the CSD register settings in the MMC/SD card. To be precise, it should match the value of the READ_BL_LEN field for read, or WRITE_BL_LEN field for write.

Figure 25-24. MMC Block Length Register (MMCBLEN)



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

Table 25-13. MMC Block Length Register (MMCBLEN) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reserved
11−0	BLEN	1h−FFFh	Block length. This field is used to set the block length, which is the byte count of a data block. The value 0 is prohibited.

Copyright © 2013–2016, Texas Instruments Incorporated

25.4.9 MMC Number of Blocks Register (MMCNBLK)

The MMC number of blocks register (MMCNBLK) specifies the number of blocks for a multiple-block transfer.

The MMC number of blocks register (MMCNBLK) is shown in Figure 25-25 and described in Table 25-14.

Figure 25-25. MMC Number of Blocks Register (MMCNBLK)

31	16
Reserved	
R-0	

15	0
NBLK	
R/W-0	

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

Table 25-14. MMC Number of Blocks Register (MMCNBLK) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	NBLK	0-FFFFh	Number of blocks. This field is used to set the total number of blocks to be transferred.
		0	Infinite number of blocks. The MMC controller reads/writes blocks of data until a STOP_TRANSMISSION command is written to the MMC command register (MMCCMD).
		1h−FFFFh	*n* blocks. The MMC controller reads/writes only *n* blocks of data, even if the STOP_TRANSMISSION command has not been written to the MMC command register (MMCCMD).

25.4.10 MMC Number of Blocks Counter Register (MMCNBLC)

The MMC number of blocks counter register (MMCNBLC) is a down-counter for tracking the number of blocks remaining to be transferred during a multiple-block transfer.

The MMC number of blocks counter register (MMCNBLC) is shown in Figure 25-26 and described in Table 25-15.

Figure 25-26. MMC Number of Blocks Counter Register (MMCNBLC)

31	16
Reserved	
R-0	

15	0
NBLC	
R-FFFFh	

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

Table 25-15. MMC Number of Blocks Counter Register (MMCNBLC) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	NBLC	0−FFFFh	Read this field to determine the number of blocks remaining to be transferred.

Copyright © 2013–2016, Texas Instruments Incorporated

25.4.11 MMC Data Receive Register (MMCDRR)

The MMC data receive register (MMCDRR) is used for storing the received data from the MMC controller. The CPU or the DMA controller can read data from this register. MMCDRR expects the data in little-endian format.

The MMC data receive register (MMCDRR) is shown in Figure 25-27 and described in Table 25-16.

Figure 25-27. MMC Data Receive Register (MMCDRR)

31	0
DRR	
R/W-0	

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

Table 25-16. MMC Data Receive Register (MMCDRR) Field Descriptions

Bit	Field	Value	Description
31-0	DRR	0-FFFF FFFFh	Data receive.

25.4.12 MMC Data Transmit Register (MMCDXR)

The MMC data transmit register (MMCDXR) is used for storing the data to be transmitted from the MMC controller to the memory card. The CPU or the DMA controller can write data to this register to be transmitted. MMCDXR expects the data in little-endian format.

The MMC data transmit register (MMCDXR) is shown in Figure 25-28 and described in Table 25-17.

Figure 25-28. MMC Data Transmit Register (MMCDXR)

31	0
DXR	
R/W-0	

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

Table 25-17. MMC Data Transmit Register (MMCDXR) Field Descriptions

Bit	Field	Value	Description
31-0	DXR	0-FFFF FFFFh	Data transmit.

Copyright © 2013–2016, Texas Instruments Incorporated

25.4.13 MMC Command Register (MMCCMD)

NOTE: Writing to the MMC command register (MMCCMD) causes the MMC controller to send the programmed command. Therefore, the MMC argument register (MMCARGHL) must be loaded properly before a write to MMCCMD.

The MMC command register (MMCCMD) specifies the type of command to be sent and defines the operation (command, response, additional activity) for the MMC controller. The content of MMCCMD is kept after the transfer to the transmit shift register. The MMC command register (MMCCMD) is shown in Figure 25-29 and described in Table 25-18.

When the CPU writes to MMCCMD, the MMC controller sends the programmed command, including any arguments in the MMC argument register (MMCARGHL). For the format of a command (index, arguments, and other bits), see Figure 25-30 and Table 25-19.

Figure 25-29. MMC Command Register (MMCCMD)

31	30	29	28	27	26	25	24
Reserved							
R-0							

23	22	21	20	19	18	17	16
Reserved							DMATRIG
R-0							R/W-0

15	14	13	12	11	10	9	8
DCLR	INITCK	WDATX	STRMTP	DTRW	RSPFMT		BSYEXP
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
PPLEN	Reserved	CMD					
R/W-0	R-0	R/W-0					

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

Table 25-18. MMC Command Register (MMCCMD) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16	DMATRIG		Data transfer triggering. (Read back as 0.)
		0	Data transfer has not been triggered.
		1	Data transfer is triggered.
15	DCLR		Data receive/transmit clear. Use this bit to clear the data receive ready (DRRDY) bit and the data transmit ready (DXRDY) bit in the MMC status register 0 (MMCST0) before a new read or write sequence. This clears any previous status.
		0	Do not clear DRRDY and DXRDY bits in MMCST0.
		1	Clear DRRDY and DXRDY bits in MMCST0.
14	INITCK		Initialization clock cycles.
		0	Do not insert initialization clock cycles.
		1	Insert initialization clock cycles; insert 80 CLK cycles before sending the command specified in the CMD bits. These dummy clock cycles are required for resetting a card after power on.
13	WDATX		Data transfer indicator.
		0	There is no data transfer.
		1	There is a data transfer associated with the command.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 25-18. MMC Command Register (MMCCMD) Field Descriptions (continued)

Bit	Field	Value	Description
12	STRMTP		Stream enable.
		0	If WDATX = 1, the data transfer is a block transfer. The data transfer stops after the movement of the programmed number of bytes (defined by the programmed block size and the programmed number of blocks).
		1	If WDATX = 1, the data transfer is a stream transfer. Once the data transfer is started, the data transfer does not stop until the MMC controller issues a stop command to the memory card.
11	DTRW		Write enable.
		0	If WDATX = 1, the data transfer is a read operation.
		1	If WDATX = 1, the data transfer is a write operation.
10-9	RSPFMT	0-3h	Response format (expected type of response to the command).
		0	No response.
		1h	R1, R4, R5, or R6 response. 48 bits with CRC.
		2h	R2 response. 136 bits with CRC.
		3h	R3 response. 48 bits with no CRC.
8	BSYEXP		Busy expected. If an R1b (R1 with busy) response is expected, set RSPFMT = 1h and BSYEXP = 1.
		0	A busy signal is not expected.
		1	A busy signal is expected.
7	PPLEN		Push pull enable.
		0	Push pull driver of CMD line is disabled (open drain).
		1	Push pull driver of CMD line is enabled.
6	Reserved	0	Reserved.
5-0	CMD	0-3Fh	Command index. This field contains the command index for the command to be sent to the memory card.

Figure 25-30. Command Format

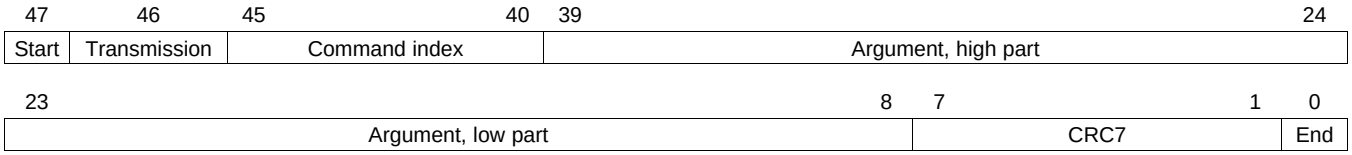


Table 25-19. Command Format

Bit Position of Command	Register	Description
47	-	Start bit
46	-	Transmission bit
45-40	MMCCMD(5-0)	Command index (CMD)
39-24	MMCARGHL	Argument, high part (ARGH)
23-8	MMCARGHL	Argument, low part (ARGL)
7-1	-	CRC7
0	-	End bit

Copyright © 2013–2016, Texas Instruments Incorporated

25.4.14 MMC Argument Register (MMCARGHL)

NOTE: Do not modify the MMC argument register (MMCARGHL) while it is being used for an operation.

The MMC argument register (MMCARGHL) specifies the arguments to be sent with the command specified in the MMC command register (MMCCMD). Writing to MMCCMD causes the MMC controller to send a command; therefore, MMCARGHL must be configured before writing to MMCCMD. The content of MMCARGHL is kept after the transfer to the shift register; however, modification to MMCARGHL is not allowed during a sending operation. For the format of a command, see Figure 25-30 and Table 25-19.

The MMC argument register (MMCARGHL) is shown in Figure 25-31 and described in Table 25-20.

Figure 25-31. MMC Argument Register (MMCARGHL)

31	16
ARGH	
R/W-0	

15	0
ARGL	
R/W-0	

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

Table 25-20. MMC Argument Register (MMCARGHL) Field Descriptions

Bit	Field	Value	Description
31-16	ARGH	0-FFFFh	Argument, high part.
15-0	ARGL	0-FFFFh	Argument, low part.

Copyright © 2013–2016, Texas Instruments Incorporated

25.4.15 MMC Response Registers (MMCRSP0-MMCRSP7)

Each command has a preset response type. When the MMC controller receives a response, it is stored in some or all of the eight MMC response registers (MMCRSP7-MMCRSP0). The response registers are updated as the responses arrive, even if the CPU has not read the previous contents.

As shown in Figure 25-32, Figure 25-33, Figure 25-34, and Figure 25-35 each of the MMC response registers holds up to 16 bits. Table 25-21 and Table 25-22 show the format for each type of response and which MMC response registers are used for the bits of the response. The first byte of the response is a command index byte and is stored in the MMC command index register (MMCCIDX).

Figure 25-32. MMC Response Register 0 and 1 (MMCRSP01)

31	16
MMCRSP1	
R/W-0	

15	0
MMCRSP0	
R/W-0	

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

Figure 25-33. MMC Response Register 2 and 3 (MMCRSP23)

31	16
MMCRSP3	
R/W-0	

15	0
MMCRSP2	
R/W-0	

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

Figure 25-34. MMC Response Register 4 and 5 (MMCRSP45)

31	16
MMCRSP5	
R/W-0	

15	0
MMCRSP4	
R/W-0	

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

Figure 25-35. MMC Response Register 6 and 7 (MMCRSP67)

31	16
MMCRSP7	
R/W-0	

15	0
MMCRSP6	
R/W-0	

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

Copyright © 2013–2016, Texas Instruments Incorporated

Table 25-21. R1, R3, R4, R5, or R6 Response (48 Bits)

Bit Position of Response	Register
47-40	MMCCIDX
39-24	MMCRSP7
23-8	MMCRSP6
7-0	MMCRSP5[(1)]
-	MMCRSP4-0

(1) Bits 7-0 of the response are stored to bits 7-0 of MMCRSP5.

Table 25-22. R2 Response (136 Bits)

Bit Position of Response	Register
135-128	MMCCIDX
127-112	MMCRSP7
111-96	MMCRSP6
95-80	MMCRSP5
79-64	MMCRSP4
63-48	MMCRSP3
47-32	MMCRSP2
31-16	MMCRSP1
15-0	MMCRSP0

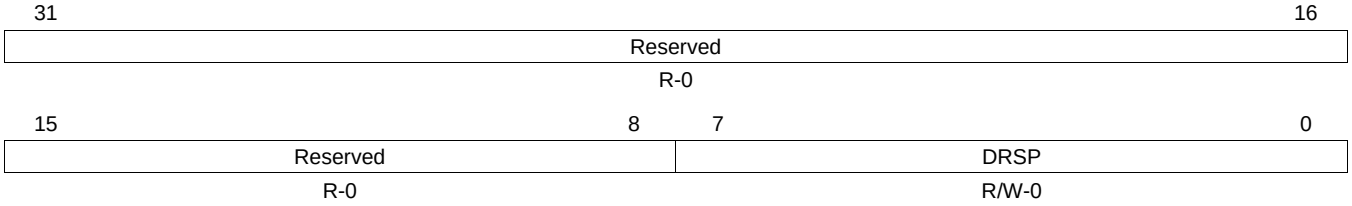
Copyright © 2013–2016, Texas Instruments Incorporated

25.4.16 MMC Data Response Register (MMCDRSP)

After the MMC controller sends a data block to a memory card, the return byte from the memory card is stored in the MMC data response register (MMCDRSP).

The MMC data response register (MMCDRSP) is shown in Figure 25-36 and described in Table 25-23.

Figure 25-36. MMC Data Response Register (MMCDRSP)



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

Table 25-23. MMC Data Response Register (MMCDRSP) Field Descriptions

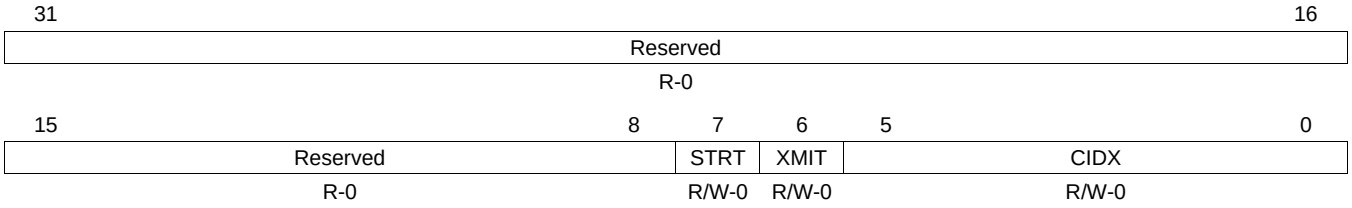
Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	DRSP	0-FFh	During a write operation (see Section 25.2.3.1), the CRC status token is stored in DRSP.

25.4.17 MMC Command Index Register (MMCCIDX)

The MMC command index register (MMCCIDX) stores the first byte of a response from a memory card. Table 25-21 and Table 25-22 show the format for each type of response.

The MMC command index register (MMCCIDX) is shown in Figure 25-37 and described in Table 25-24.

Figure 25-37. MMC Command Index Register (MMCCIDX)



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

Table 25-24. MMC Command Index Register (MMCCIDX) Field Descriptions

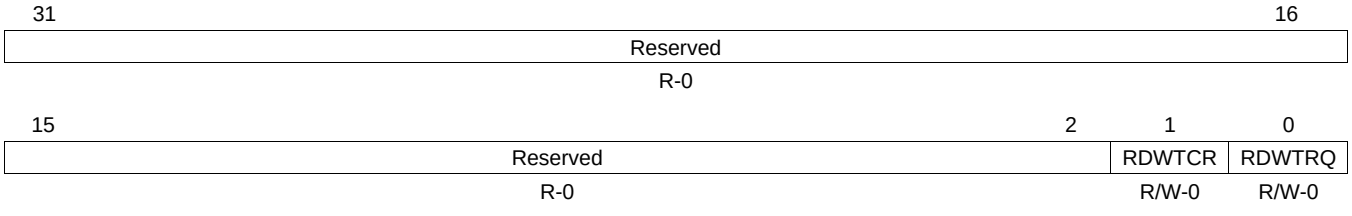
Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	STRT	0-1	Start bit. When the MMC controller receives a response, the start bit is stored in STRT.
6	XMIT	0-1	Transmission bit. When the MMC controller receives a response, the transmission bit is stored in XMIT.
5-0	CIDX	0-3Fh	Command index. When the MMC controller receives a response, the command index is stored in CIDX.

Copyright © 2013–2016, Texas Instruments Incorporated

25.4.18 SDIO Control Register (SDIOCTL)

The SDIO control register (SDIOCTL) is shown in Figure 25-38 and described in Table 25-25.

Figure 25-38. SDIO Control Register (SDIOCTL)



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

Table 25-25. SDIO Control Register (SDIOCTL) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	RDWTCR		Read wait enable for CRC error. To end the read wait operation, write 0 to RDWTRQ. (No need to clear RDWTCR).
		0	Read wait is disabled.
		1	Automatically start read wait on CRC error detection during multiple block read access and not the last block to be transferred. RDWTRQ is automatically set to 1.
0	RDWTRQ		Read wait request. To end the read wait operation, write 0 to RDWTRQ.
		0	End read wait operation and release MMCSD_DAT2.
		1	Set a read wait request. Read wait operation starts 2 clocks after the end of the read data block. MMCIF asserts low level on MMCSD_DAT2 until RDWTRQ is cleared to 0.

Copyright © 2013–2016, Texas Instruments Incorporated

25.4.19 SDIO Status Register 0 (SDIOST0)

The SDIO status register 0 (SDIOST0) is shown in Figure 25-39 and described in Table 25-26.

Figure 25-39. SDIO Status Register 0 (SDIOST0)

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		RDWTST	INTPRD	DAT1
R-0		R-0	R-0	R-1

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

Table 25-26. SDIO Status Register 0 (SDIOST0) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	RDWTST		Read wait status.
		0	Read wait operation not in progress.
		1	Read wait operation in progress.
1	INTPRD		Interrupt period.
		0	Interrupt not in progress.
		1	Interrupt in progress.
0	DAT1		This bit reflects the external state of the SD_DATA1 pin.
		0	Logic-low level on the SD_DATA1 pin.
		1	Logic-high level on the SD_DATA1 pin.

Copyright © 2013–2016, Texas Instruments Incorporated

25.4.20 SDIO Interrupt Enable Register (SDIOIEN)

The SDIO interrupt enable register (SDIOIEN) is shown in Figure 25-40 and described in Table 25-27.

Figure 25-40. SDIO Interrupt Enable Register (SDIOIEN)

31			16
Reserved			
R-0			

15	2	1	0
Reserved		RWSEN	IOINTEN
R-0		R/W-0	R/W-0

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

Table 25-27. SDIO Interrupt Enable Register (SDIOIEN) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	RWSEN		Read wait interrupt enable.
		0	Read wait interrupt is disabled.
		1	Read wait interrupt is enabled.
0	IOINTEN		SDIO card interrupt enable.
		0	SDIO card interrupt is disabled.
		1	SDIO card interrupt is enabled.

25.4.21 SDIO Interrupt Status Register (SDIOIST)

The SDIO interrupt status register (SDIOIST) is shown in Figure 25-41 and described in Table 25-28.

Figure 25-41. SDIO Interrupt Status Register (SDIOIST)

31			16
Reserved			
R-0			

15	2	1	0
Reserved		RWS	IOINT
R-0		R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing 0 has no effect); -*n* = value after reset

Table 25-28. SDIO Interrupt Status Register (SDIOIST) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	RWS		Read wait interrupt status. Write a 1 to clear this bit.
		0	Read wait interrupt did not occur.
		1	Read wait interrupt occurred. Read wait operation starts and read wait interrupt is enabled (RWSEN = 1 in SDIOIEN).
0	IOINT		SDIO card interrupt status. Write a 1 to clear this bit.
		0	SDIO card interrupt did not occur.
		1	SDIO card interrupt occurred. SDIO card interrupt is detected and SDIO card interrupt is enabled (IOINTEN = 1 in SDIOIEN).

Copyright © 2013–2016, Texas Instruments Incorporated

25.4.22 MMC FIFO Control Register (MMCFIFOCTL)

The MMC FIFO control register (MMCFIFOCTL) is shown in Figure 25-42 and described in Table 25-29.

Figure 25-42. MMC FIFO Control Register (MMCFIFOCTL)

31	16
Reserved	
R-0	

15	5	4	3	2	1	0
Reserved		ACCWD		FIFOLEV	FIFODIR	FIFORST
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 25-29. MMC FIFO Control Register (MMCFIFOCTL) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4-3	ACCWD	0-3h	Access width. Used by FIFO control to determine full/empty flag.
		0	CPU/EDMA access width of 4 bytes.
		1h	CPU/EDMA access width of 3 bytes.
		2h	CPU/EDMA access width of 2 bytes.
		3h	CPU/EDMA access width of 1 byte.
2	FIFOLEV		FIFO level. Sets the threshold level that determines when the EDMA request and the FIFO threshold interrupt are triggered.
		0	EDMA request every 256 bits (32 bytes) sent/received.
		1	EDMA request every 512 bits (64 bytes) sent/received.
1	FIFODIR		FIFO direction. Determines if the FIFO is being written to or read from.
		0	Read from FIFO.
		1	Write to FIFO.
0	FIFORST		FIFO reset. Resets the internal state of the FIFO.
		0	FIFO reset is disabled.
		1	FIFO reset is enabled.

Copyright © 2013–2016, Texas Instruments Incorporated

Real-Time Clock (RTC)

This chapter describes the real-time clock (RTC).

Topic **Page**

26.1 Introduction **1182**
26.2 Architecture **1183**
26.3 Registers **1189**

Copyright © 2013–2016, Texas Instruments Incorporated

26.1 Introduction

26.1.1 Purpose of the Peripheral

The real-time clock (RTC) provides a time reference to an application running on the device. The current date and time is tracked in a set of counter registers that update once per second. The time can be represented in 12-hour or 24-hour mode. The calendar and time registers are buffered during reads and writes so that updates do not interfere with the accuracy of the time and date.

Alarms are available to interrupt the CPU at a particular time, or at periodic time intervals, such as once per minute or once per day. In addition, the RTC can interrupt the CPU every time the calendar and time registers are updated, or at programmable periodic intervals.

26.1.2 Features

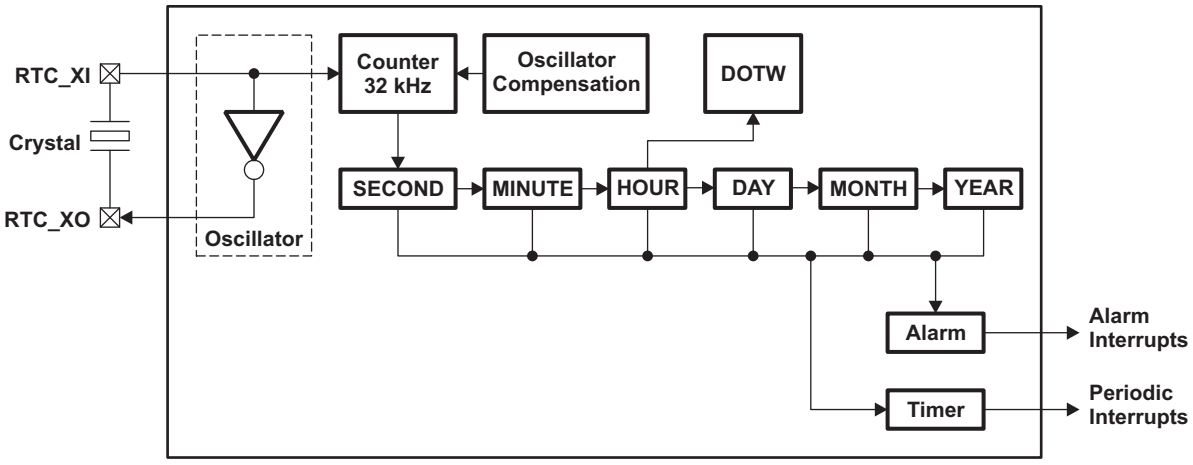
The real-time clock (RTC) provides the following features:

- 100-year calendar (xx00 to xx99)
- Counts seconds, minutes, hours, day of the week, date, month, and year with leap year compensation
- Binary-coded-decimal (BCD) representation of time, calendar, and alarm
- 12-hour clock mode (with AM and PM) or 24-hour clock mode
- Alarm interrupt
- Periodic interrupt
- Single interrupt to the CPU
- Supports external 32.768-kHz crystal or external clock source of the same frequency
- Isolated power supply

26.1.3 Block Diagram

Figure 26-1 shows a block diagram of the RTC.

Figure 26-1. Real-Time Clock Block Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

26.2 Architecture

26.2.1 Clock Source

The clock reference for the RTC is an external 32.768-kHz crystal or an external clock source of the same frequency. The RTC also has a separate power supply that is isolated from the rest of the system. When the CPU and other peripherals are without power, the RTC can remain powered to preserve the current time and calendar information.

The source for the RTC reference clock may be provided by a crystal or by an external clock source. The RTC has an internal oscillator buffer to support direct operation with a crystal. The crystal is connected between pins RTC_XI and RTC_XO. RTC_XI is the input to the on-chip oscillator and RTC_XO is the output from the oscillator back to the crystal. For more information about the RTC crystal connection, see your device-specific data manual.

An external 32.768-kHz clock source may be used instead of a crystal. In such a case, the clock source is connected to RTC_XI, and RTC_XO is left unconnected.

If the RTC is not used, the RTC_XI pin should be held low and RTC_XO should be left unconnected. The RTCDISABLE bit in the control register (CTRL) can be set to save power; however, the RTCDISABLE bit should not be cleared once it has been set. If the application requires the RTC module to stop and continue, the RUN bit in CTRL should be used instead.

26.2.2 Signal Descriptions

The RTC signals are listed in Table 26-1.

Table 26-1. Real-Time Clock Signals

Signal	I/O	Description
RTC_XI	I	RTC time base input signal. RTC_XI can either be driven with a 32.768-kHz reference clock, or RTC_XI and RTC_XO can be connected to an external crystal. This signal is the input to the RTC internal oscillator.
RTC_XO	O	RTC time base output signal. RTC_XO is the output from the RTC internal oscillator. If a crystal is not used as the time base for RTC_XI, RTC_XO should be left unconnected.

26.2.3 Isolated Power Supply

The RTC has a power supply that is isolated from the rest of the system. This allows the RTC to continue to run while the rest of the system is not powered. In this state, the RTC time and calendar counters continue to run, but the powered down CPU is not able to receive RTC interrupts. Separate power supply pins for the RTC are provided on the device package.

26.2.3.1 Split-Power Circuitry

To decrease power consumption, RTC includes leakage-isolation circuitry that is activated by setting the SPLITPOWER bit in the control register (CTRL). Because of its isolated power supply, RTC does not have a power-on hardware reset signal. Therefore, upon initial device power-on, the RTC is in an unknown state until it has been properly configured. After the RTC module has been configured once, it functions as programmed as long as its power supply and clock source are provided.

26.2.3.2 Power Considerations

The RTC leakage-isolation circuitry requires that the CPU supply be powered down to VSS when the RTC is powered on while the rest of the device is powered off. A floating CPU supply creates undesired RTC leakage current. Also, the RTC power consumption is higher when the CPU is powered on versus the RTC power consumption when the CPU is powered off. Therefore, if the RTC module is expected to run from a small-capacity power supply (ex. watch battery) while the rest of the device is powered off, a power system should be implemented such that the RTC is powered from a high-capacity power supply when the CPU is powered on.

Copyright © 2013–2016, Texas Instruments Incorporated

26.2.4 Operation

26.2.4.1 Using the Real-Time Clock Time and Calendar Registers

The current time and date are maintained in the RTC time and calendar registers.

26.2.4.1.1 Time/Calendar Data Format

The time and calendar data in the RTC is stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. Although most of the time/calendar registers have 4 bits assigned to each BCD digit, some of the register fields are shorter since the range of valid numbers may be limited. For example, only 3 bits are required to represent the day since only BCD numbers 1 though 7 are required.

The following time and calendar registers are supported (BCD Format):

- SECOND - Second Count (00-59)
- MINUTE - Minute Count (00-59)
- HOUR - Hour Count (12HR: 01-12; 24HR: 00-23)
- DAY - Day of the Month Count (01-31)
- MONTH - Month Count (01-12; JAN = 1)
- YEAR - Year Count (00-99)
- DOTW - Day of the Week Count (0-6; SUN = 0)

Note that the ALARM registers which share the names above also share the same BCD formatting.

26.2.4.1.2 12-Hour and 24-Hour Modes

The current time can be represented in 12-hour or 24-hour mode by configuring the HOURMODE bit in the control register (CTRL):

- When HOURMODE = 0, 24-hour mode is selected. The hours are represented as 00 through 23. The MERIDIEM bit in the HOURS register has no function and should be cleared.
- When HOURMODE = 1, 12-hour mode is selected. The hours are represented as 00 through 12. MERIDIEM = 0 indicates ante meridiem (AM), and MERIDIEM = 1 indicates post meridiem (PM).

26.2.4.1.3 Reading from Time/Calendar Registers

The time/calendar registers are updated every second as the time changes. During a read of the SECOND register, the RTC copies the current values of the time/date registers into shadow read registers. This isolation assures that the CPU can capture all the time/date values at the moment of the SECOND read request and not be subject to changing register values from time updates.

If desired, the RTC also provides a one-time-triggered minute-rounding feature to round the MINUTE:SECOND registers to the nearest minute (with zero seconds). This feature is enabled by setting the ROUNDMIN bit in the control register (CTRL); the RTC automatically rounds the time values to the nearest minute upon the next read of the SECOND register.

26.2.4.1.4 Writing to Time/Calendar Registers

When setting the RTC time and date, values are written directly to the time/calendar registers. Therefore, care must be taken to avoid writing to the time/calendar registers while the time is updating to the next second. This can be accomplished in one of two ways:

1. The RTC can be stopped by clearing the RUN bit in the control register (CTRL). When stopped, there is no danger of contention during writes because the registers do not auto-update.
2. The BUSY bit in the status register (STATUS) is low when a time update does not take place for at least 15 μs. By checking for a low BUSY bit before writing to registers, the CPU is assured of a 15 μs window of time during which multiple accesses to the time/calendar registers can be performed.

Copyright © 2013–2016, Texas Instruments Incorporated

After writing to a time/calendar register, the RTC requires four peripheral clock cycles to update the register value. Any reads that take place within four peripheral clock cycles of a write returns old data.

Note that all registers in the RTC except for KICK*n*R have write-protection. See Section 26.2.6 for information on unlocking registers.

26.2.4.2 Real-Time Clock Update Cycle

The RTC executes an update cycle once per second to update the current time in the time/calendar registers. The update cycle also compares each alarm register with the corresponding time register. These comparisons are done to determine when to trigger an alarm. The BUSY bit in the status register (STATUS) provides a mechanism to indicate when the time/calendar registers are updated. When the BUSY bit is high, an update takes place within 15 μs. When BUSY returns low again, the update has been completed.

The BUSY bit should be checked when writing to any of the following registers while RTC is running:

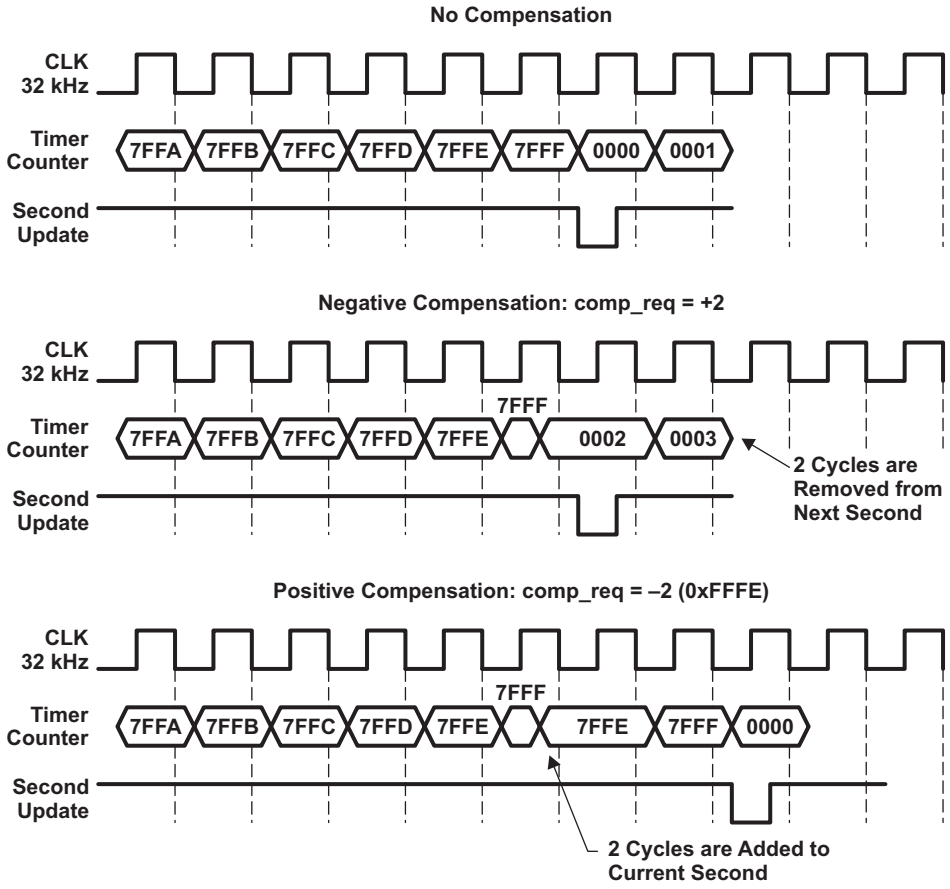
- SECOND
- MINUTE
- HOUR
- DAY
- MONTH
- YEAR
- DOTW
- ALARMSECOND (when ALARM interrupt is enabled)
- ALARMMINUTE (when ALARM interrupt is enabled)
- ALARMHOUR (when ALARM interrupt is enabled)
- ALARMDAY (when ALARM interrupt is enabled)
- ALARMMONTH (when ALARM interrupt is enabled)
- ALARMYEAR (when ALARM interrupt is enabled)
- CTRL (SET32COUNTER field only -- the other fields in CTRL do not require BUSY to be low)
- INTERRUPT
- COMPLSB (when oscillator drift compensation is enabled)
- COMPMSB (when oscillator drift compensation is enabled)

26.2.4.3 Oscillator Drift Compensation

If the RTC 32.768-kHz reference clock is susceptible to oscillator drift, the RTC provides the ability to compensate the update cycle by subtracting oscillator periods. The COMPMSB and COMPLSB registers hold the number of two's complement reference periods to subtract from the update cycle every hour. For example, Figure 26-2 shows how programming the value of 2h into the compensation registers shortens the update cycle by two 32.786-kHz reference periods every hour. Figure 26-2 also shows how programming the value of FFFEh (decimal negative 2) into the compensation register lengthens the update cycle by two reference periods every hour. To enable the oscillator compensation, the AUTOCOMP bit in the control register (CTRL) must be set.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 26-2. 32-kHz Oscillator Counter Compensation



26.2.5 Interrupt Requests

The RTC provides the ability to interrupt the CPU based on two events: a periodic interrupt and an alarm interrupt. Although two interrupt sources are available, the RTC makes a single interrupt request to the CPU.

When the device is initially powered on, the RTC may issue spurious interrupt signals to the CPU. To avoid issues, a software reset should be performed on the RTC module before the CPU interrupt controller is initialized. See Section 26.2.10 for more information on reset considerations.

26.2.5.1 Alarm Interrupt Enable and Status Bits

The ALARM bit in the interrupt register (INTERRUPT) enables the alarm interrupt. When the current time and date match the ALARMSECOND, ALARMMINUTE, ALARMHOUR, ALARMDAY, ALARMMONTH, and ALARMYEAR registers, the RTC issues an interrupt to the CPU and sets the ALARM bit in the status register (STATUS). Once set, the ALARM status bit stays high until cleared by a write of 1 to the ALARM bit.

As with writing to time and calendar registers (Section 26.2.4.1.4), writes to the INTERRUPT and STATUS registers should only be done when the RTC is stopped or when the BUSY bit is low.

Note that all registers in the RTC except for KICK*n*R have write-protection. See Section 26.2.6 for information on unlocking registers.

Copyright © 2013–2016, Texas Instruments Incorporated

26.2.5.2 Periodic Interrupt Enable and Status Bits

The TIMER bit and EVERY field in the interrupt register (INTERRUPT) work together to enable periodic interrupts. When the TIMER bit is enabled, interrupts are issued at a time period indicated by the EVERY field (0 = Second, 1h = Minute, 2h = Hour, 3h = Day). Regardless of the period selected in the EVERY field, the periodic timer status bits (DAYEVT, HREVT, MINEVT, SECEVT) are set in the status register (STATUS) whenever they are valid. Note that the appropriate status bits are set when the TIME bit is enabled, not when the desired interrupt is generated. Active periodic status bits remain high as long as the TIMER bit is enabled.

For example, if daily periodic interrupts are enabled and the time (in HH:MM:SS format) transitions from 23:59:59 to 00:00:00, the STATUS register sets all four periodic status bits (DAYEVT, HREVT, MINEVT, and SECEVT) because all four time periods were incremented. These bits all remain high until:

1. The TIME bit is cleared and all four status bits clear to zero until TIME is set again **OR**
2. The current time reaches 00:00:01. At that point, the SECEVT remains set while the DAYEVT, HREVT, and MINEVT bits are cleared. The next interrupt is not generated until the next day transition.

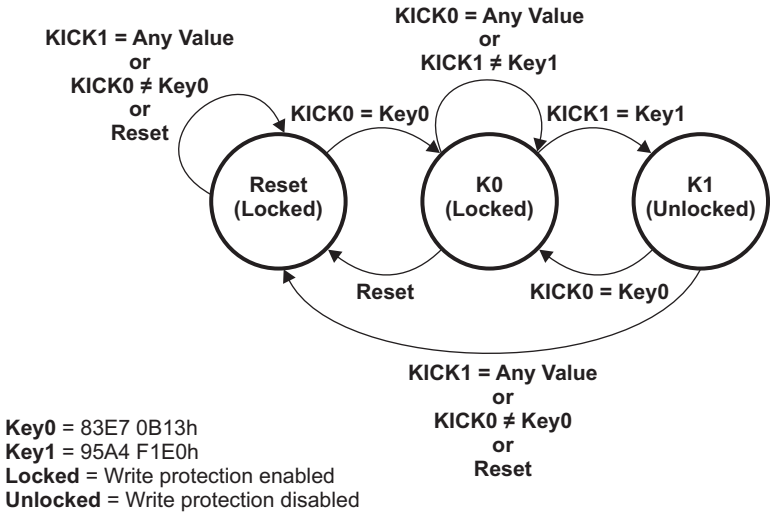
As with writing to time and calendar registers (Section 26.2.4.1.4), writes to the INTERRUPT and STATUS registers should only be done when the RTC is stopped or when the BUSY bit is low.

Note that all registers in the RTC except for KICK*n*R have write-protection. See Section 26.2.6 for information on unlocking registers.

26.2.6 Register Protection Against Spurious Writes

All registers in the RTC except for the KICK*n*R registers are protected from spurious writes. Out of reset, writes to protected registers are disabled until the registers are unlocked using the KICK*n*R registers. To unlock the registers, a key of 83E7 0B13h needs to be written to KICK0R, followed by a write of 95A4 F1E0h to KICK1R. Registers remain unlocked until write protection is enabled again by writing any value to KICK0R or KICK1R. The write protection state machine is shown in Figure 26-3.

Figure 26-3. Kick State Machine



Copyright © 2013–2016, Texas Instruments Incorporated

26.2.7 General-Purpose Scratch Registers

The RTC provides three general-purpose registers (SCRATCH*n*) that can be used to store 32-bit words -- these registers have no functional purpose for the RTC. Software using the RTC may find the SCRATCH*n* registers to be useful in indicating RTC states. For example, the SCRATCH*n* registers may be used to indicate write-protection lock status or unintentional power downs.

To indicate write-protection, the software should write a unique value to one of the SCRATCH*n* registers when write-protection is disabled and another unique value when write-protection is enabled again. In this way, the lock-status of the registers can be determined quickly by reading the SCRATCH register.

To indicate unintentional power downs, the software should write a unique value to one of the SCRATCH*n* registers when RTC is configured and enabled. If the RTC is unintentionally powered down, the value written to the SCRATCH register is cleared.

26.2.8 Real-Time Clock Response to Low Power Modes (Idle Configurations)

The device is divided into idle domains that can be programmed to be idle or active. The state of all domains is called the idle configuration. The RTC runs on its own external clock source and is not affected by any of the other device idle domains.

26.2.9 Emulation Modes of the Real-Time Clock

The RTC always continues to run regardless of the state (running/halted) of the emulation debugger software.

26.2.10 Reset Considerations

When the device is initially powered on, the RTC may issue spurious interrupt signals to the CPU. To avoid issues, a software reset should be performed on the RTC module before the CPU interrupt controller is initialized.

As the RTC is configured, the SPLITPOWER bit in the control register (CTRL) should be set.

A software reset is performed on the RTC by setting the SWRESET bit in the oscillator register (OSC). The software reset applies to all registers except the oscillator (OSC) and kick (KICK*n*R) registers. The RTC requires three 32.768-kHz reference clocks to pass before RTC registers can be accessed.

Copyright © 2013–2016, Texas Instruments Incorporated

26.3 Registers

Table 26-2 lists the memory-mapped registers for the RTC. See your device-specific data manual for the memory address of these registers.

Table 26-2. Real-Time Clock (RTC) Registers

Address Offset	Acronym	Register Description	Section
0h	SECOND	Seconds Register	Section 26.3.1
4h	MINUTE	Minutes Register	Section 26.3.2
8h	HOUR	Hours Register	Section 26.3.3
Ch	DAY	Day of the Month Register	Section 26.3.4
10h	MONTH	Month Register	Section 26.3.5
14h	YEAR	Year Register	Section 26.3.6
18h	DOTW	Day of the Week Register	Section 26.3.7
20h	ALARMSECOND	Alarm Seconds Register	Section 26.3.8
24h	ALARMMINUTE	Alarm Minutes Register	Section 26.3.9
28h	ALARMHOUR	Alarm Hours Register	Section 26.3.10
2Ch	ALARMDAY	Alarm Days Register	Section 26.3.11
30h	ALARMMONTH	Alarm Months Register	Section 26.3.12
34h	ALARMYEAR	Alarm Years Register	Section 26.3.13
40h	CTRL	Control Register	Section 26.3.14
44h	STATUS	Status Register	Section 26.3.15
48h	INTERRUPT	Interrupt Enable Register	Section 26.3.16
4Ch	COMPLSB	Compensation (LSB) Register	Section 26.3.17
50h	COMPMSB	Compensation (MSB) Register	Section 26.3.18
54h	OSC	Oscillator Register	Section 26.3.19
60h	SCRATCH0	Scratch 0 Register (General-Purpose)	Section 26.3.20
64h	SCRATCH1	Scratch 1 Register (General-Purpose)	Section 26.3.20
68h	SCRATCH2	Scratch 2 Register (General-Purpose)	Section 26.3.20
6Ch	KICK0R	Kick 0 Register (Write Protect)	Section 26.3.21
70h	KICK1R	Kick 1 Register (Write Protect)	Section 26.3.21

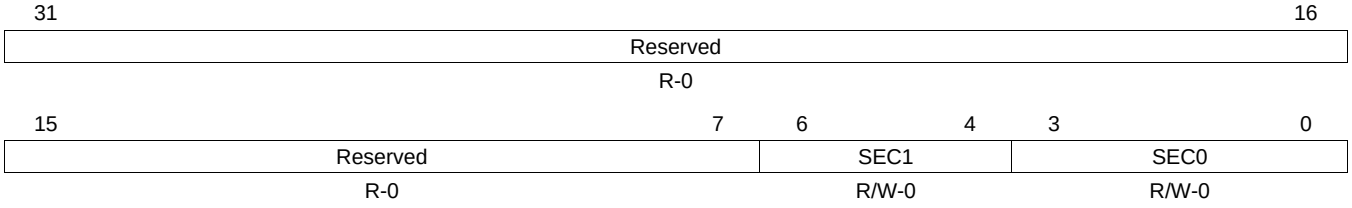
Copyright © 2013–2016, Texas Instruments Incorporated

26.3.1 Second Register (SECOND)

NOTE: Out of reset, the second register (SECOND) is write-protected. To disable write protection, correct keys must be written to the KICK*n*R registers (see Section 26.2.6).

The second register (SECOND) sets the second value of the current time. Seconds are stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The SECOND register is shown in Figure 26-4 and described in Table 26-3.

Figure 26-4. Second Register (SECOND)



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

Table 26-3. Second Register (SECOND) Field Descriptions

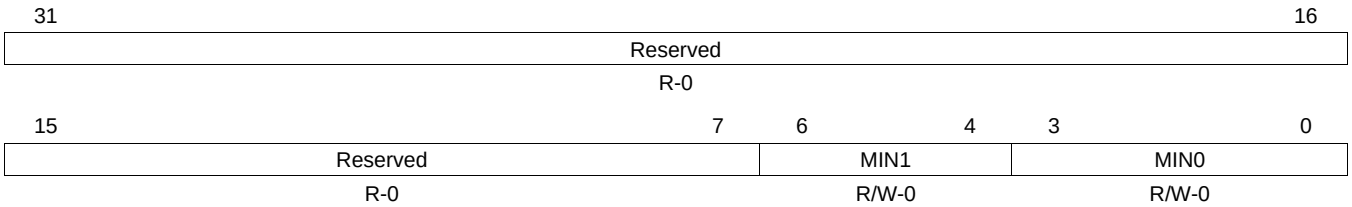
Bit	Field	Value	Description
31-7	Reserved	0	Reserved.
6-4	SEC1	0-5h	Most significant digit of second value. Range for SEC1:SEC0 is 00-59.
3-0	SEC0	0-9h	Least significant digit of second value. Range for SEC1:SEC0 is 00-59.

26.3.2 Minute Register (MINUTE)

NOTE: Out of reset, the minute register (MINUTE) is write-protected. To disable write protection, correct keys must be written to the KICK*n*R registers (see Section 26.2.6).

The minute register (MINUTE) sets the minute value of the current time. Minutes are stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The MINUTE register is shown in Figure 26-5 and described in Table 26-4.

Figure 26-5. Minute Register (MINUTE)



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

Table 26-4. Minute Register (MINUTE) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reserved.
6-4	MIN1	0-5h	Most significant digit of minute value. Range for MIN1:MIN0 is 00-59.
3-0	MIN0	0-9h	Least significant digit of minute value. Range for MIN1:MIN0 is 00-59.

Copyright © 2013–2016, Texas Instruments Incorporated

26.3.3 Hour Register (HOUR)

NOTE: Out of reset, the hour register (HOUR) is write-protected. To disable write protection, correct keys must be written to the KICK*n*R registers (see Section 26.2.6).

The hour register (HOUR) sets the hour value of the current time. Hours are stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The HOUR register is shown in Figure 26-6 and described in Table 26-5.

Figure 26-6. Hour Register (HOUR)

31	16
Reserved	
R-0	

15	8	7	6	5	4	3	0
Reserved		MERIDIEM	Rsvd	HOUR1		HOUR0	
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 26-5. Hour Register (HOUR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved.
7	MERIDIEM		Determines whether the hour is ante meridiem (AM) or post meridiem (PM) when 12-hour mode is enabled.
		0	Hour is AM
		1	Hour is PM
6	Reserved	0	Reserved.
5-4	HOUR1	0-2h	Most significant digit of hours value. Range for HOUR1:HOUR0 is 00-24.
3-0	HOUR0	0-9h	Least significant digit of hours value. Range for HOUR1:HOUR0 is 00-24.

Copyright © 2013–2016, Texas Instruments Incorporated

26.3.4 Day of the Month Register (DAY)

NOTE: Out of reset, the day of the month register (DAY) is write-protected. To disable write protection, correct keys must be written to the KICK*n*R registers (see Section 26.2.6).

The day of the month register (DAY) sets the day of the month value of the current date. Days are stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The DAY register is shown in Figure 26-7 and described in Table 26-6.

Figure 26-7. Days Register (DAY)

31	16
Reserved	
R-0	

15	6	5	4	3	0
Reserved		DAY1		DAY0	
R-0		R/W-0		R/W-1	

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

Table 26-6. Day Register (DAY) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reserved.
5-4	DAY1	0-3h	Most significant digit of day of the month value. Range for DAY1:DAY0 is 01-31.
3-0	DAY0	0-9h	Least significant digit of day of the month value. Range for DAY1:DAY0 is 01-31.

26.3.5 Month Register (MONTH)

NOTE: Out of reset, the month register (MONTH) is write-protected. To disable write protection, correct keys must be written to the KICK*n*R registers (see Section 26.2.6).

The month register (MONTH) sets the month in the year value of the current date. The month is stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The MONTH register is shown in Figure 26-8 and described in Table 26-7.

Figure 26-8. Month Register (MONTH)

31	16
Reserved	
R-0	

15	5	4	3	0
Reserved		MONTH1	MONTH0	
R-0		R/W-0	R/W-1	

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

Table 26-7. Month Register (MONTH) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved.
4	MONTH1	0-1h	Most significant digit of months value. For MONTH1:MONTH0, JAN = 01 and DEC = 12.
3-0	MONTH0	0-9h	Least significant digit of months value. For MONTH1:MONTH0, JAN = 01 and DEC = 12.

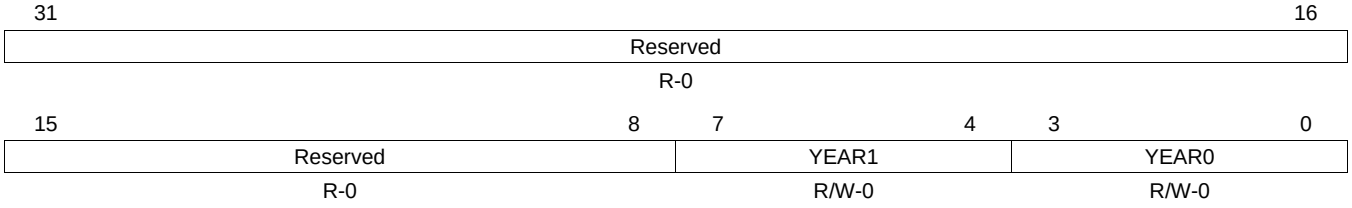
Copyright © 2013–2016, Texas Instruments Incorporated

26.3.6 Year Register (YEAR)

NOTE: Out of reset, the year register (YEAR) is write-protected. To disable write protection, correct keys must be written to the KICK*n*R registers (see Section 26.2.6).

The year register (YEAR) sets the year value of the current date. The year is stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The YEAR register is shown in Figure 26-9 and described in Table 26-8.

Figure 26-9. Year Register (YEAR)



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

Table 26-8. Year Register (YEAR) Field Descriptions

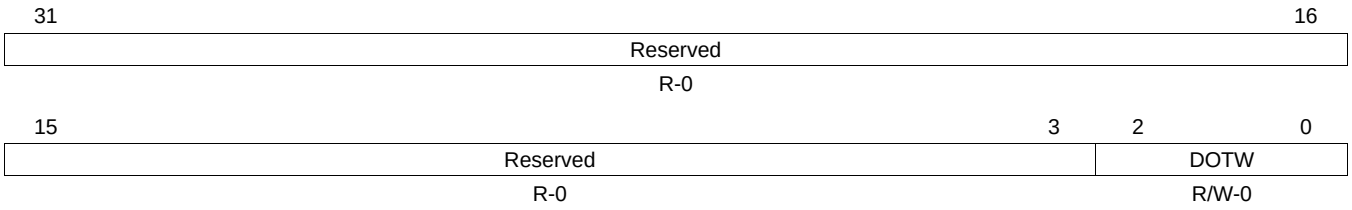
Bit	Field	Value	Description
31-8	Reserved	0	Reserved.
7-4	YEAR1	0-9h	Most significant digit of years value. Range for YEAR1:YEAR0 is 00-99.
3-0	YEAR0	0-9h	Least significant digit of years value. Range for YEAR1:YEAR0 is 00-99.

26.3.7 Day of the Week Register (DOTW)

NOTE: Out of reset, the day of the week register (DOTW) is write-protected. To disable write protection, correct keys must be written to the KICK*n*R registers (see Section 26.2.6).

The day of the week register (DOTW) sets the day of the week value of the current date. The day of the week is stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The DOTW register is shown in Figure 26-10 and described in Table 26-9.

Figure 26-10. Day of the Week Register (DOTW)



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

Table 26-9. Day of the Week Register (DOTW) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved.
2-0	DOTW	0-6h	Day of the week. Sunday = 0, Saturday = 6h.

Copyright © 2013–2016, Texas Instruments Incorporated

26.3.8 Alarm Second Register (ALARMSECOND)

NOTE: Out of reset, the alarm second register (ALARMSECOND) is write-protected. To disable write protection, correct keys must be written to the KICK*n*R registers (see Section 26.2.6).

The alarm second register (ALARMSECOND) sets the second value for the alarm interrupt. Seconds are stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The ALARMSECOND register is shown in Figure 26-11 and described in Table 26-10.

Figure 26-11. Alarm Second Register (ALARMSECOND)

31	16
Reserved	
R-0	

15	7	6	4	3	0
Reserved		SEC1		SEC0	
R-0		R/W-0		R/W-0	

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

Table 26-10. Alarm Second Register (ALARMSECOND) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reserved.
6-4	SEC1	0-5h	Most significant digit of alarm seconds value. Range for SEC1:SEC0 is 00-59.
3-0	SEC0	0-9h	Least significant digit of alarm seconds value. Range for SEC1:SEC0 is 00-59.

26.3.9 Alarm Minute Register (ALARMMINUTE)

NOTE: Out of reset, the alarm minute register (ALARMMINUTE) is write-protected. To disable write protection, correct keys must be written to the KICK*n*R registers (see Section 26.2.6).

The alarm minute register (ALARMMINUTE) sets the minute value for the alarm interrupt. Minutes are stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The ALARMMINUTE register is shown in Figure 26-12 and described in Table 26-11.

Figure 26-12. Alarm Minute Register (ALARMMINUTE)

31	16
Reserved	
R-0	

15	7	6	4	3	0
Reserved		MIN1		MIN0	
R-0		R/W-0		R/W-0	

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

Table 26-11. Alarm Minute Register (ALARMMINUTE) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reserved.
6-4	MIN1	0-5h	Most significant digit of alarm minutes value. Range for MIN1:MIN0 is 00-59.
3-0	MIN0	0-9h	Least significant digit of alarm minutes value. Range for MIN1:MIN0 is 00-59.

Copyright © 2013–2016, Texas Instruments Incorporated

26.3.10 Alarm Hour Register (ALARMHOUR)

NOTE: Out of reset, the alarm hour register (ALARMHOUR) is write-protected. To disable write protection, correct keys must be written to the KICK*n*R registers (see Section 26.2.6).

The alarm hour register (ALARMHOUR) sets the hour value for the alarm interrupt. Hours are stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The ALARMHOUR register is shown in Figure 26-13 and described in Table 26-12.

Figure 26-13. Alarm Hour Register (ALARMHOUR)

31	16
Reserved	
R-0	

15	8	7	6	5	4	3	0
Reserved		MERIDIEM	Rsvd	HOUR1		HOUR0	
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 26-12. Alarm Hour Register (ALARMHOUR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved.
7	MERIDIEM		Determines whether the hour is ante meridiem (AM) or post meridiem (PM) when 12-hour mode is enabled.
		0	Hour is AM
		1	Hour is PM
6	Reserved	0	Reserved.
5-4	HOUR1	0-2h	Most significant digit of hours value. Range for HOUR1:HOUR0 is 00-24.
3-0	HOUR0	0-9h	Least significant digit of hours value. Range for HOUR1:HOUR0 is 00-24.

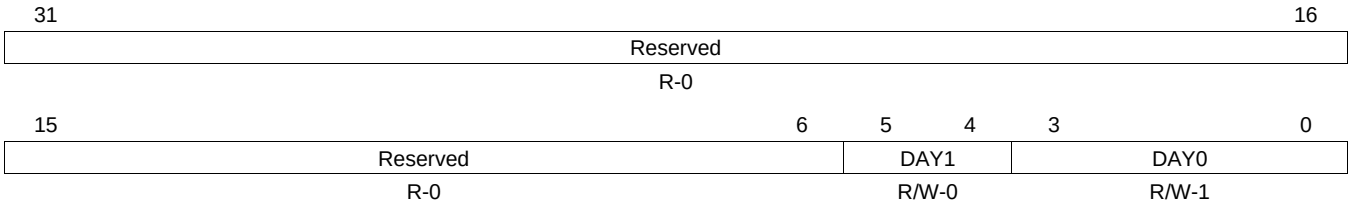
Copyright © 2013–2016, Texas Instruments Incorporated

26.3.11 Alarm Day of the Month Register (ALARMDAY)

NOTE: Out of reset, the alarm day of the month register (ALARMDAY) is write-protected. To disable write protection, correct keys must be written to the KICK*n*R registers (see Section 26.2.6).

The alarm day of the month register (ALARMDAY) sets the day of the month value for the alarm interrupt. Days are stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The ALARMDAYS register is shown in Figure 26-14 and described in Table 26-13.

Figure 26-14. Alarm Day Register (ALARMDAY)



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

Table 26-13. Alarm Day Register (ALARMDAY) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reserved.
5-4	DAY1	0-3h	Most significant digit of day of the month value. Range for DAY1:DAY0 is 01-31.
3-0	DAY0	0-9h	Least significant digit of day of the month value. Range for DAY1:DAY0 is 01-31.

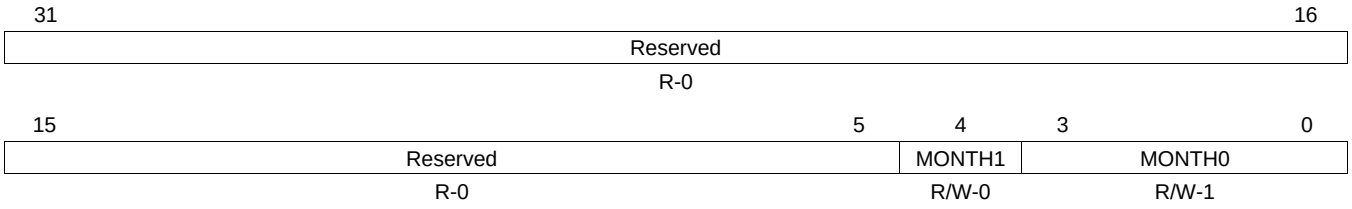
Copyright © 2013–2016, Texas Instruments Incorporated

26.3.12 Alarm Month Register (ALARMMONTH)

NOTE: Out of reset, the alarm month register (ALARMMONTH) is write-protected. To disable write protection, correct keys must be written to the KICK*n*R registers (see Section 26.2.6).

The alarm month register (ALARMMONTH) sets the month in the year value for the alarm interrupt. The month is stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The ALARMMONTH register is shown in Figure 26-15 and described in Table 26-14.

Figure 26-15. Alarm Month Register (ALARMMONTH)



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

Table 26-14. Alarm Month Register (ALARMMONTH) Field Descriptions

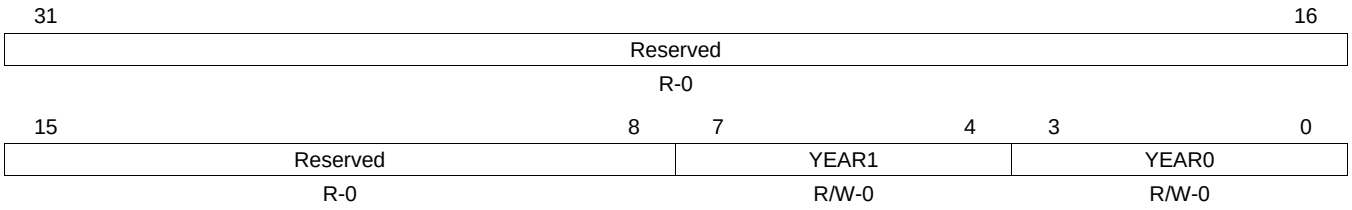
Bit	Field	Value	Description
31-5	Reserved	0	Reserved.
4	MONTH1	0-1h	Most significant digit of months value. For MONTH1:MONTH0, JAN = 01 and DEC = 12.
3-0	MONTH0	0-9h	Least significant digit of months value. For MONTH1:MONTH0, JAN = 01 and DEC = 12.

26.3.13 Alarm Year Register (ALARMYEAR)

NOTE: Out of reset, the alarm year register (ALARMYEAR) is write-protected. To disable write protection, correct keys must be written to the KICK*n*R registers (see Section 26.2.6).

The alarm year register (ALARMYEAR) sets the year for the alarm interrupt. The year is stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The ALARMYEAR register is shown in Figure 26-16 and described in Table 26-15.

Figure 26-16. Alarm Year Register (ALARMYEAR)



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

Table 26-15. Alarm Years Register (ALARMYEARS) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved.
7-4	YEAR1	0-9h	Most significant digit of years value. Range for YEAR1:YEAR0 is 00-99.
3-0	YEAR0	0-9h	Least significant digit of years value. Range for YEAR1:YEAR0 is 00-99.

Copyright © 2013–2016, Texas Instruments Incorporated

26.3.14 Control Register (CTRL)

NOTE: Out of reset, the control register (CTRL) is write-protected. To disable write protection, correct keys must be written to the KICK*n*R registers (see Section 26.2.6).

The control register (CTRL) is shown in Figure 26-17 and described in Table 26-16.

Figure 26-17. Control Register (CTRL)

31							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
SPLITPOWER	RTCDISABLE	SET32COUNTER	Reserved	HOURMODE	AUTOCOMP	ROUNDMIN	RUN
W-0	R/W-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; W = Write only; -*n* = value after reset

Table 26-16. Control Register (CTRL) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved.
7	SPLITPOWER		Enable leakage-isolation circuitry used for isolated power schemes. **Write-only bit. Read-modify-write updates to the control register may unintentionally clear the SPLITPOWER bit because the bit always reads back 0.**
		0	Disable split power.
		1	Enable split power.
6	RTCDISABLE		Disable RTC module and gate 32-kHz reference clock. RTC should only be disabled using this bit if the module will never be used and saving power is desired.
		0	RTC is functional.
		1	RTC is disabled and 32-kHz reference clock is gated.
5	SET32COUNTER		Set the 32-kHz counter with the value stored in the compensation registers when the SET32COUNTER bit is set. RTC does not run normally when the SET32COUNTER bit is high so this bit should be toggled low-high-low when used.
		0	No action.
		1	Set 32-kHz counter with compensation register value.
4	Reserved	0	Reserved.
3	HOURMODE		Enable 12-hour mode for HOURS and ALARMHOURS registers.
		0	24 Hour Mode (Valid hours 00-24).
		1	12 Hour Mode (Valid hours 00-12; MERIDIEM bit in HOURS and ALARMHOURS must be used to denote AM or PM).
2	AUTOCOMP		Enable oscillator compensation mode. Compensation takes place once every hour.
		0	Auto compensation is disabled.
		1	Auto compensation is enabled.
1	ROUNDMIN		Enable one-time rounding to nearest minute on next time register read.
		0	Minute rounding disabled.
		1	Rounding to nearest minute enabled.
0	RUN		Stop the RTC 32-kHz counter. RTC should be stopped using this bit for stopping and resuming the counter.
		0	Stop RTC counter.
		1	Run RTC counter.

Copyright © 2013–2016, Texas Instruments Incorporated

26.3.15 Status Register (STATUS)

The STATUS register is shown in Figure 26-18 and described in Table 26-17.

NOTE: Out of reset, the STATUS register is write-protected. To disable write protection, correct keys must be written to the KICK*n*R registers (see Section 26.2.6).

Figure 26-18. Status Register (STATUS)

31							16
Reserved							
R-0							

15	7	6	5	4	3	2	1	0
Reserved		ALARM	DAYEVT	HREVT	MINEVT	SECEVT	RUN	BUSY
R-1		R/W1C-0	R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R = Read only; W1C = Write 1 to clear bit; -*n* = value after reset

Table 26-17. Status Register (STATUS) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	1	Reserved
6	ALARM		Indicates if an alarm interrupt has been generated. Write a 1 to clear the ALARM status.
		0	No new alarm interrupt was generated.
		1	New alarm interrupt was generated.
5	DAYEVT		When the (TIMER = 1) in the INTERRUPTS register, DAYEVT indicates if the DAYS register incremented during the most recent time update.
		0	DAYS register did not increment during the last time update.
		1	DAYS register incremented during the last time update.
4	HREVT		When the (TIMER = 1) in the INTERRUPTS register, HREVT indicates if the HOURS register incremented during the most recent time update.
		0	HOURS register did not increment during the last time update.
		1	HOURS register incremented during the last time update.
3	MINEVT		When the (TIMER = 1) in the INTERRUPTS register, MINEVT indicates if the MINUTES register incremented during the most recent time update.
		0	MINUTES register did not increment during the last time update.
		1	MINUTES register incremented during the last time update.
2	SECEVT		When the (TIMER = 1) in the INTERRUPTS register, SECEVT indicates if the SECONDS register incremented during the most recent time update.
		0	SECONDS register did not increment during the last time update.
		1	SECONDS register incremented during the last time update.
1	RUN		Indicates if RTC is running or stopped.
		0	RTC is stopped.
		1	RTC is running.
0	BUSY		Indicates if RTC is busy updating or is within 15 µs of updating the time and calendar registers.
		0	RTC is free. The time, calendar, and control registers can be written to without contention.
		1	RTC is or will soon be busy updating the time and calendar registers.

Copyright © 2013–2016, Texas Instruments Incorporated

26.3.16 Interrupt Register (INTERRUPT)

The INTERRUPT register is shown in Figure 26-19 and described in Table 26-18.

NOTE: Out of reset, the INTERRUPT register is write-protected. To disable write protection, correct keys must be written to the KICK*n*R registers (see Section 26.2.6).

Figure 26-19. Interrupt Register (INTERRUPT)

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		ALARM	TIMER	EVERY	
R-0		R/W-0	R/W-0	R/W-0	

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

Table 26-18. Interrupt Register (INTERRUPT) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved.
3	ALARM		Enable interrupt generation for when the alarm time and date match the current time and date
		0	Alarm interrupt is disabled.
		1	Alarm interrupt is enabled.
2	TIMER		Enable periodic timer interrupt generation. Period is determined by the EVERY field.
		0	Periodic timer interrupt is disabled.
		1	Periodic timer interrupt is enabled.
1-0	EVERY	0-3h	Selects the time period desired when periodic timer interrupts are enabled by the TIMER bit.
		0	Second
		1h	Minute
		2h	Hour
		3h	Day

Copyright © 2013–2016, Texas Instruments Incorporated

26.3.17 Compensation (LSB) Register (COMPLSB)

NOTE: Out of reset, the compensation (LSB) register (COMPLSB) is write-protected. To disable write protection, correct keys must be written to the KICK*n*R registers (see Section 26.2.6).

The compensation (LSB) register (COMPLSB) works together with the COMPMSB register to set the hourly oscillator compensation value. The AUTOCOMP bit in the control register (CTRL) must be enabled for compensation to take place. The COMPLSB register is shown in Figure 26-20 and described in Table 26-19.

Figure 26-20. Compensation (LSB) Register (COMPLSB)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		COMPLSB	
R-0		R/W-0	

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

Table 26-19. Compensations Register (COMPLSB) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved.
7-0	COMPLSB	0-FFh	Lower bits of the 16-bit compensation value. The COMPMSB:COMPLSB register value is subtracted from the 32-kHz period. Compensation values are two's complement. The COMPMSB:COMPLSB value of 7F:FFh is not allowed.

Copyright © 2013–2016, Texas Instruments Incorporated

26.3.18 Compensation (MSB) Register (COMPMSB)

NOTE: Out of reset, the compensation (MSB) register (COMPMSB) is write-protected. To disable write protection, correct keys must be written to the KICK*n*R registers (see Section 26.2.6).

The compensation (MSB) register (COMPMSB) works together with the COMPLSB register to set the hourly oscillator compensation value. The AUTOCOMP bit in the control register (CTRL) must be enabled for compensation to take place. The COMPMSB register is shown in Figure 26-21 and described in Table 26-20.

Figure 26-21. Compensation (MSB) Register (COMPMSB)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		COMPMSB	
R-0		R/W-0	

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

Table 26-20. Compensations Register (COMPMSB) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved.
7-0	COMPMSB	0-FFh	Lower bits of the 16-bit compensation value. The COMPMSB:COMPLSB register value is subtracted from the 32-kHz period. Compensation values are two's complement. The COMPMSB:COMPLSB value of 7F:FFh is not allowed.

Copyright © 2013–2016, Texas Instruments Incorporated

26.3.19 Oscillator Register (OSC)

NOTE: Out of reset, the oscillator register (OSC) is write-protected. To disable write protection, correct keys must be written to the KICK*n*R registers (see Section 26.2.6).

The oscillator register (OSC) is shown in Figure 26-22 and described in Table 26-21.

Figure 26-22. Oscillator Register (OSC)

31				16
Reserved				
R-0				

15	6	5	4	0
Reserved		SWRESET	Reserved	
R-0		W-0	R/W-7h	

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

Table 26-21. Oscillator Register (OSC) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reserved.
5	SWRESET		Software reset. Always reads back 0.
		0	No action.
		1	Reset RTC module and registers (except for OSC and KICK*n*R registers). Registers must not be accessed for three 32-kHz reference periods after reset is asserted.
4-0	Reserved	7h	Reserved. This field is writeable, but should only be programmed to the value of 7h.

Copyright © 2013–2016, Texas Instruments Incorporated

26.3.20 Scratch Registers (SCRATCH0-SCRATCH2)

NOTE: Out of reset, the scratch registers (SCRATCH*n*) are write-protected. To disable write protection, correct keys must be written to the KICK*n*R registers (see Section 26.2.6).

The scratch registers (SCRATCH*n*) are 32-bit general-purpose registers that have no effect on RTC functionality. They can be used by the user arbitrarily. The SCRATCH*n* register is shown in Figure 26-23 and described in Table 26-22.

Figure 26-23. Scratch Registers (SCRATCH*n*)

31	0
SCRATCH*n*	
R/W-0	

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

Table 26-22. Scratch Registers (SCRATCH*n*) Field Descriptions

Bit	Field	Value	Description
31-0	SCRATCH*n*	0-FFFF FFFFh	General-purpose 32-bit registers that have no effect on RTC functionality.

26.3.21 Kick Registers (KICK0R, KICK1R)

The kick registers (KICK*n*R) are used to enable and disable write protection on the RTC registers. Out of reset, the RTC registers are write-protected. To disable write protection, correct keys must be written to the KICK*n*R registers (see Section 26.2.6). The KICK*n*R register is shown in Figure 26-24 and described in Table 26-23.

Figure 26-24. Kick Registers (KICK*n*R)

31	0
KICK*n*	
R/W-0	

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

Table 26-23. Kick Registers (KICK*n*R) Field Descriptions

Bit	Field	Value	Description
31-0	KICK*n*	0-FFFF FFFFh	To disable RTC register write protection, the value of 83E7 0B13h must be written to KICK0R, followed by the value of 95A4 F1E0h written to KICK1R. RTC register write protection is enabled when any value is written to KICK0R.

Copyright © 2013–2016, Texas Instruments Incorporated

Serial Peripheral Interface (SPI)

This chapter describes the serial peripheral interface (SPI) module. See your device-specific data manual to determine how many SPIs are available on your device.

Topic **Page**

27.1 Introduction **1206**
27.2 Architecture **1208**
27.3 Registers **1234**

Copyright © 2013–2016, Texas Instruments Incorporated

27.1 Introduction

27.1.1 Purpose of the Peripheral

The SPI is a high-speed synchronous serial input/output port that allows a serial bit stream of programmed length (2 to 16 bits) to be shifted into and out of the device at a programmed bit-transfer rate. The SPI is normally used for communication between the device and external peripherals. Typical applications include interface to external I/O or peripheral expansion via devices such as shift registers, display drivers, SPI EPROMS and analog-to-digital converters.

27.1.2 Features

The SPI has the following features:

- 16-bit shift register
- 16-bit Receive buffer register (SPIBUF) and 16-bit Receive buffer emulation 'alias' register (SPIEMU)
- 16-bit Transmit data register (SPIDAT0) and 16-bit Transmit data and format selection register (SPIDAT1)
- 8-bit baud clock generator
- Serial clock (SPIx_CLK) I/O pin
- Slave in, master out (SPIx_SIMO) I/O pin
- Slave out, master in (SPIx_SOMI) I/O pin
- SPI enable ($\overline{\text{SPIx_ENA}}$) I/O pin (4-pin or 5-pin mode only)
- Multiple slave chip select ($\overline{\text{SPIx_SCS[n]}}$) I/O pins (4-pin or 5-pin mode only)
- Programmable SPI clock frequency range
- Programmable character length (2 to 16 bits)
- Programmable clock phase (delay or no delay)
- Programmable clock polarity (high or low)
- Interrupt capability
- DMA support (read/write synchronization events)

The SPI allows software to program the following options:

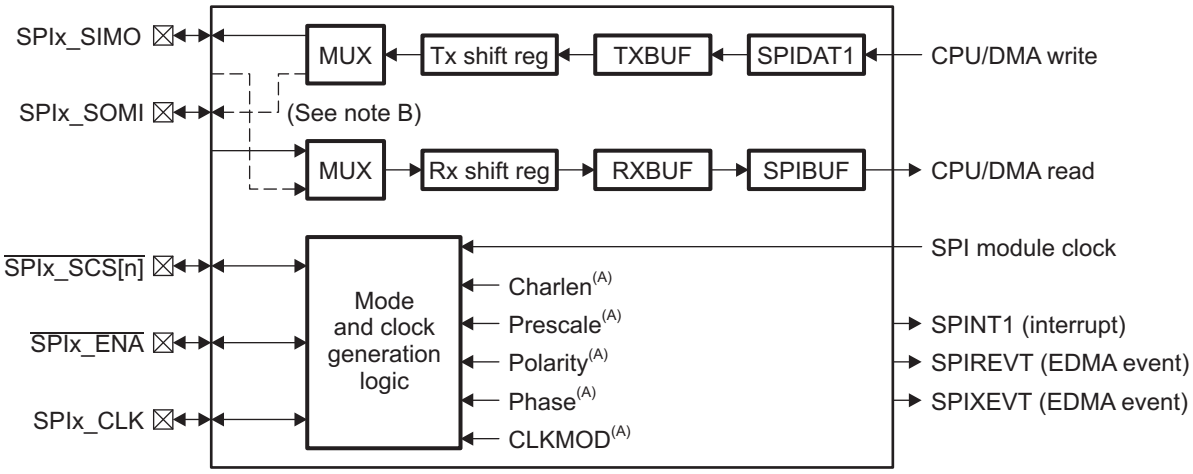
- SPI pins as functional or digital I/O pins
- SPI Master or Slave mode
- SPIx_CLK frequency (SPI module clock/3 through SPI module clock/256)
- 3-pin, 4-pin, and 5-pin options
- Character length (2 to 16 bits) and shift direction (MSB/LSB first)
- Clock phase (delay or no delay) and polarity (high or low)
- Delay between transmissions in master mode.
- Chip select setup and hold times in master mode
- Chip select hold in master mode

Copyright © 2013–2016, Texas Instruments Incorporated

27.1.3 Functional Block Diagram

The Figure 27-1 shows the SPI block diagram.

Figure 27-1. SPI Block Diagram



NOTE: The value *x* indicates the applicable SPI; that is, SPI0, SPI1, etc. See your device-specific data manual to determine how many SPIs are available on your device. The value *n* indicates the SPI pins available. See your device-specific data manual to determine how many SPI pins are available on your device.

A Indicates the log controlled by SPI register bits.

B Solid line represents data flow for SPI master mode. Dashed line represents data flow for SPI slave mode.

27.1.4 Industry Standard(s) Compliance Statement

The programmable configuration capability of the SPI allows it to gluelessly interface to a variety of SPI format devices. The SPI does not conform to a specific industry standard.

Copyright © 2013–2016, Texas Instruments Incorporated

27.2 Architecture

This section describes the SPI operation modes. It gives an overview of SPI operation and then provides details on the 3-pin, 4-pin, and 5-pin options, as well as more specific details on the supported data formats.

27.2.1 Clock

The SPI clock (SPIx_CLK) is derived from the SPI module clock. The maximum clock bit rate supported is SPI module clock/3, as determined by the PRESCALE field in the SPI data format register *n* (SPIFMT*n*). The SPIx_CLK frequency is calculated as:

SPIx_CLK frequency = [SPI module clock] / [SPIFMT*n*.PRESCALE + 1]

27.2.2 Signal Descriptions

Table 27-1 shows the SPI pins used to interface to external devices.

Table 27-1. SPI Pins

Pin[1]	Type	Function
SPIx_SIMO	Input/Output	Serial data input in slave mode, serial data output in master mode
SPIx_SOMI	Input/Output	Serial data output in slave mode, serial data input in master mode
SPIx_CLK	Input/Output	Serial clock input in slave mode, serial clock output in master mode
$\overline{\text{SPIx_SCS[n]}}$ [2]	Input/Output	Slave chip select output in master mode, input in slave mode
$\overline{\text{SPIx_ENA}}$	Input/Output	Input in master mode, output in slave mode indicates slave is ready

(1) The value *x* indicates the applicable SPI; that is, SPI0, SPI1, etc. See your device-specific data manual to determine how many SPIs are available on your device.

(2) The value *n* indicates the SPI pins available; that is, $\overline{\text{SPIx_SCS[0]}}$, $\overline{\text{SPIx_SCS[1]}}$, etc. See your device-specific data manual to determine how many SPI pins are available on your device.

27.2.3 Operation Modes

The SPI operates in master or slave mode. The SPI bus master is the device that drives the SPIx_CLK, SPIx_SIMO, and optionally the $\overline{\text{SPIx_SCS[n]}}$ signals, and therefore initiates SPI bus transfers. The CLKMOD and MASTER bits in the SPI global control register 1 (SPIGCR1) select between master and slave mode. In both master and slave mode, the SPI supports four options:

- 3-pin option
- 4-pin with chip select option
- 4-pin with enable option
- 5-pin with enable and chip select option

The 3-pin option is the basic clock, data in, and data out SPI interface and uses the SPIx_CLK, SPIx_SIMO, and SPIx_SOMI pins. The 4-pin with chip select option adds the $\overline{\text{SPIx_SCS[n]}}$ pin that is used to support multiple SPI slave devices on a single SPI bus. The 4-pin with enable option adds the $\overline{\text{SPIx_ENA}}$ pin that is used to increase the overall throughput by adding hardware handshaking. The 5-pin option uses all the SPI pins and is a superset of the different options.

Copyright © 2013–2016, Texas Instruments Incorporated

27.2.4 Programmable Registers

A general representation of the SPI programmable registers is shown in Table 27-2. For details on registers, see Section 27.3.

Table 27-2. SPI Registers

Offset Address[1]	Acronym	Name	Description	Section
0h	SPIGCR0	Global control register 0	Contains the software reset bit for the module	Section 27.3.1
4h	SPIGCR1	Global control register 1	Controls basic configurations of the module	Section 27.3.2
8h	SPIINT0	Interrupt register	Enable bits for interrupts, error, DMA and other functionality.	Section 27.3.3
Ch	SPILVL	Level register	SPI interrupt levels are set in this register.	Section 27.3.4
10h	SPIFLG	Flag register	Shows the status of several events during the operation.	Section 27.3.5
14h	SPIPC0	Pin control register 0	Determines if pins operate as general I/O or SPI functional pin	Section 27.3.6
18h	SPIPC1	Pin control register 1	Controls the direction of data on the I/O pins	Section 27.3.7
1Ch	SPIPC2	Pin control register 2	Reflects the values on the I/O pins	Section 27.3.8
20h	SPIPC3	Pin control register 3	Controls the values sent to the I/O pins	Section 27.3.9
24h	SPIPC4	Pin control register 4	Sets data values in the SPIPC3 register	Section 27.3.10
28h	SPIPC5	Pin control register 5	Clears values in the SPIPC3 register	Section 27.3.11
38h	SPIDAT0	Transmit data register 0	Transmit data register	Section 27.3.12
3Ch	SPIDAT1	Transmit data register 1	Transmit data with format selection register	Section 27.3.13
40h	SPIBUF	Receive buffer register	Holds received word	Section 27.3.14
44h	SPIEMU	Receive buffer emulation register	Mirror of SPIBUF. Read does not clear flags	Section 27.3.15
48h	SPIDELAY	Delay register	Sets $\overline{\text{SPIx_SCS[n]}}$ mode, $\overline{\text{SPIx_SCS[n]}}$ pre-/post-transfer delay time and $\overline{\text{SPIx_ENA}}$ time-out	Section 27.3.16
4Ch	SPIDEF	Chip select default register	In $\overline{\text{SPIx_SCS[n]}}$ decoded mode only: sets high low/active $\overline{\text{SPIx_SCS[n]}}$ signal	Section 27.3.17
50h	SPIFMT0	Format 0 register	Configuration of data word format 0	Section 27.3.18
54h	SPIFMT1	Format 1 register	Configuration of data word format 1	Section 27.3.18
58h	SPIFMT2	Format 2 register	Configuration of data word format 2	Section 27.3.18
5Ch	SPIFMT3	Format 3 register	Configuration of data word format 3	Section 27.3.18
64h	INTVEC1	Interrupt vector register 1	Interrupt vector for line INT1	Section 27.3.19

(1) The actual address of these registers is device specific and CPU specific. See your device-specific data manual to verify the SPI register addresses.

Copyright © 2013–2016, Texas Instruments Incorporated

27.2.5 Master Mode Settings

The four master mode options are defined by the configuration bit settings listed in Table 27-3. Other configuration bits may take any value in the range listed in Table 27-4. The values listed in Table 27-3 and Table 27-4 should not be changed while the ENABLE bit in the SPI global control register 1 (SPIGCR1) is set to 1. Note that in certain cases the allowed values may still be ignored. For example, Table 27-4 indicates that SPIDELAY may take a range of values in Master 3-pin mode; however, SPIDELAY has no effect in Master 3-pin mode. For complete details on each mode, see the following sections that explain the SPI operation for each of the master modes.

Table 27-3. SPI Register Settings Defining Master Modes

Register	Bit(s)	Master 3-pin	Master 4-pin Chip Select	Master 4-pin Enable	Master 5-pin
SPIGCR0	RESET	1	1	1	1
SPIGCR1	ENABLE	1	1	1	1
SPIGCR1	LOOPBACK	0	0	0	0
SPIGCR1	CLKMOD	1	1	1	1
SPIGCR1	MASTER	1	1	1	1
SPIPC0	SOMIFUN	1	1	1	1
SPIPC0	SIMOFUN	1	1	1	1
SPIPC0	CLKFUN	1	1	1	1
SPIPC0	ENAFUN	0	0	1	1
SPIPC0	SCS0FUN	0	1	0	1

Table 27-4. Allowed SPI Register Settings in Master Modes

Register	Bit(s)	Master 3-pin	Master 4-pin Chip Select	Master 4-pin Enable	Master 5-pin
SPIINT0	ENABLEHIGHZ	0,1	0,1	0,1	0,1
SPIFMT*n*	WDELAY	0 to 3Fh	0 to 3Fh	0 to 3Fh	0 to 3Fh
SPIFMT*n*	PARPOL	0,1	0,1	0,1	0,1
SPIFMT*n*	PARENA	0,1	0,1	0,1	0,1
SPIFMT*n*	WAITENA	0	0	1	1
SPIFMT*n*	SHIFTDIR	0,1	0,1	0,1	0,1
SPIFMT*n*	DISCSTIMERS	0,1	0,1	0,1	0,1
SPIFMT*n*	POLARITY	0,1	0,1	0,1	0,1
SPIFMT*n*	PHASE	0,1	0,1	0,1	0,1
SPIFMT*n*	PRESCALE	2 to FFh	2 to FFh	2 to FFh	2 to FFh
SPIFMT*n*	CHARLEN	2 to 10h	2 to 10h	2 to 10h	2 to 10h
SPIDELAY	C2TDELAY	0 to FFh	0 to FFh	0 to FFh	0 to FFh
SPIDELAY	T2CDELAY	0 to FFh	0 to FFh	0 to FFh	0 to FFh
SPIDELAY	T2EDELAY	0 to FFh	0 to FFh	0 to FFh	0 to FFh
SPIDELAY	C2EDELAY	0 to FFh	0 to FFh	0 to FFh	0 to FFh

27.2.5.1 Master Mode Timing Options

The SPI in master mode supports several options to modify the timing of its generation of the chip select signal ($\overline{\text{SPIx_SCS[n]}}$). This allows the SPI to support the timing requirements of various slave devices without adding additional overhead to the CPU by generating the appropriate delays automatically.

Copyright © 2013–2016, Texas Instruments Incorporated

27.2.5.1.1 Chip Select Setup Time

The master can be configured to provide a (slow) slave device a certain chip select setup time to the first edge on SPIx_CLK. This delay is controlled by the C2TDELAY field in the SPI delay register (SPIDELAY) and can be configured between 3 and 257 SPI module clock cycles. The C2TDELAY is applicable only in 4-pin with chip select and 5-pin SPI master modes. The C2TDELAY begins when the SPI master asserts $\overline{\text{SPIx_SCS[n]}}$. The C2T delay period is specified by:

Maximum duration of C2TDELAY period = SPIDELAY.C2TDELAY + 2 (SPI module clock cycles)

Note that if SPIDELAY.C2TDELAY = 0, then the C2TDELAY period = 0.

The previous value of the CSHOLD bit in the SPI transmit data register (SPIDAT1) must be cleared to 0 for the C2T delay to be enabled.

NOTE: If the SPIDAT1.CSHOLD bit is set within the control field, the current hold time and the following setup time will not be applied in between transaction.

27.2.5.1.2 Chip Select Hold Time

The master can be configured to provide a (slow) slave device a certain chip select hold time after the last edge on SPIx_CLK. This delay is controlled by the T2CDELAY bit in the SPI delay register (SPIDELAY) and can be configured between 2 and 256 SPI module clock cycles. The T2CDELAY is applicable only in 4-pin with chip select and 5-pin SPI master modes. The T2CDELAY begins after the data shifting period ends. The T2C delay period is specified by:

Maximum duration of T2CDELAY period = SPIDELAY.T2CDELAY + 1 (SPI module clock cycle)

Note that if SPIDELAY.T2CDELAY = 0, then the T2CDELAY period = 0. If the PHASE bit in the SPI data format register *n* (SPIFMT*n*) is 0, then the T2CDELAY period lasts for an additional 1/2 SPIx_CLK time over that specified by the above equation.

The current value of the CSHOLD bit in the SPI transmit data register (SPIDAT1) must be cleared to 0 for T2C delay to be enabled.

NOTE: If the SPIDAT1.CSHOLD bit is set within the control field, the current hold time and the following setup time will not be applied in between transaction.

27.2.5.1.3 Automatic Delay Between Transfers

The SPI master can automatically insert a delay of between 2 and 65 SPI module clock cycles between transmissions. This delay is controlled by the WDELAY field in the SPI data format register *n* (SPIFMT*n*) and is enabled by setting the WDEL bit in the SPI transmit data register (SPIDAT1) to 1. The WDELAY period begins when the T2EDELAY period terminates (if T2E delay period is enabled) or when the T2CDELAY period terminates (if T2E delay period was disabled and T2C delay period was enabled) or when the master deasserts $\overline{\text{SPIx_SCS[n]}}$ (if T2E and T2C delay periods are disabled). If a transfer is initiated by writing a 32-bit value to SPIDAT1, then the new values of SPIDAT1.WDEL and SPIFMT*n*.WDELAY are used; otherwise, the old values of SPIDAT1.WDEL and SPIFMT*n*.WDELAY are used. The WDELAY delay period is specified by:

Maximum duration of WDELAY period = SPIFMTn.WDELAY + 2 (SPI module clock cycles)

27.2.5.1.4 Chip Select Hold Option

There are slave devices available that require the chip select signal to be held continuously active during several consecutive data word transfers. Other slave devices require the chip select signal to be deactivated between consecutive data word transfers. The SPI can support both types of slave devices. The CSHOLD bit in the SPI transmit data register (SPIDAT1) selects between the two options.

Copyright © 2013–2016, Texas Instruments Incorporated

If the chip select hold option is enabled, the chip select will not toggle between two consecutive accesses; therefore, the SPIDELAY.T2CDELAY of the first transfer and the SPIDELAY.C2TDELAY of the second transfer will not be applied. However, the wait delay could still be applied between the two transactions, if the WDEL bit in SPIDAT1 is set to 1.

The current and previous values of the CSHOLD bit are retained. Though the current value of the CSHOLD bit is initialized to 0 when the RESET bit in the SPI global control register 0 (SPIGCR0) is cleared to 0, the previous value of the CSHOLD bit is not initialized. The previous value of the CSHOLD bit must be explicitly initialized by writing twice to the CSHOLD bit.

27.2.6 Slave Mode Settings

The four slave mode options are defined by the configuration bit settings listed in Table 27-5. Other configuration bits may take any value in the range listed in Table 27-6. The values listed in Table 27-5 and Table 27-6 should not be changed while the ENABLE bit in the SPI global control register 1 (SPIGCR1) is set to 1. Note that in certain cases the allowed values may still be ignored. For complete details on each mode, see the following sections that explain the SPI operation for each of the slave modes.

Table 27-5. SPI Register Settings Defining Slave Modes

Register	Bit(s)	Slave 3-pin	Slave 4-pin Chip Select	Slave 4-pin Enable	Slave 5-pin
SPIGCR0	RESET	1	1	1	1
SPIGCR1	ENABLE	1	1	1	1
SPIGCR1	LOOPBACK	0	0	0	0
SPIGCR1	CLKMOD	0	0	0	0
SPIGCR1	MASTER	0	0	0	0
SPIPC0	SOMIFUN	1	1	1	1
SPIPC0	SIMOFUN	1	1	1	1
SPIPC0	CLKFUN	1	1	1	1
SPIPC0	ENAFUN	0	0	1	1
SPIPC0	SCS0FUN	0	1	0	1

Table 27-6. Allowed SPI Register Settings in Slave Modes

Register	Bit(s)	Slave 3-pin	Slave 4-pin Chip Select	Slave 4-pin Enable	Slave 5-pin
SPIINT0	ENABLEHIGHZ	0,1	0,1	0,1	0,1
SPIFMT*n* [(1)]	WDELAY	0 to 3Fh	0 to 3Fh	0 to 3Fh	0 to 3Fh
SPIFMT*n* [(1)]	PARPOL	0,1	0,1	0,1	0,1
SPIFMT*n* [(1)]	PARENA	0,1	0,1	0,1	0,1
SPIFMT*n* [(1)]	WAITENA	0,1	0,1	0,1	0,1
SPIFMT*n* [(1)]	SHIFTDIR	0,1	0,1	0,1	0,1
SPIFMT*n* [(1)]	DISCSTIMERS	0,1	0,1	0,1	0,1
SPIFMT*n* [(1)]	POLARITY	0,1	0,1	0,1	0,1
SPIFMT*n* [(1)]	PHASE	0,1	0,1	0,1	0,1
SPIFMT*n* [(1)]	PRESCALE	2 to FFh	2 to FFh	2 to FFh	2 to FFh
SPIFMT*n* [(1)]	CHARLEN	2 to 10h	2 to 10h	2 to 10h	2 to 10h
SPIDELAY	C2TDELAY	0 to FFh	0 to FFh	0 to FFh	0 to FFh
SPIDELAY	T2CDELAY	0 to FFh	0 to FFh	0 to FFh	0 to FFh
SPIDELAY	T2EDELAY	0 to FFh	0 to FFh	0 to FFh	0 to FFh
SPIDELAY	C2EDELAY	0 to FFh	0 to FFh	0 to FFh	0 to FFh

(1) In slave mode, only SPIFMT0 is used. When SPIDAT1 is written, the DFSEL field in SPIDAT1 is cleared to 0 to select SPIFMT0.

Copyright © 2013–2016, Texas Instruments Incorporated

27.2.7 SPI Operation: 3-Pin Mode

NOTE: If only unidirectional communication is required, the SPIx_CLK pin and the two data pins (SPIx_SOMI and SPIx_SIMO) must all be configured as functional pins. A 2-pin unidirectional mode is not supported.

The SPI 3-pin mode uses only the clock (SPIx_CLK) and data (SPIx_SOMI and SPIx_SIMO) pins for bidirectional communication between master and slave devices. Figure 27-2 shows the basic 3-pin SPI option.

To select the 3-pin SPI option, the SPIx_CLK, SPIx_SOMI, and SPIx_SIMO pins should be configured as functional pins by configuring the SPI pin control register 0 (SPIPC0). The $\overline{\text{SPIx_SCS[n]}}$ and $\overline{\text{SPIx_ENA}}$ pins can be used as general-purpose I/O pins by configuring the SPIPC1 through SPIPC5 registers.

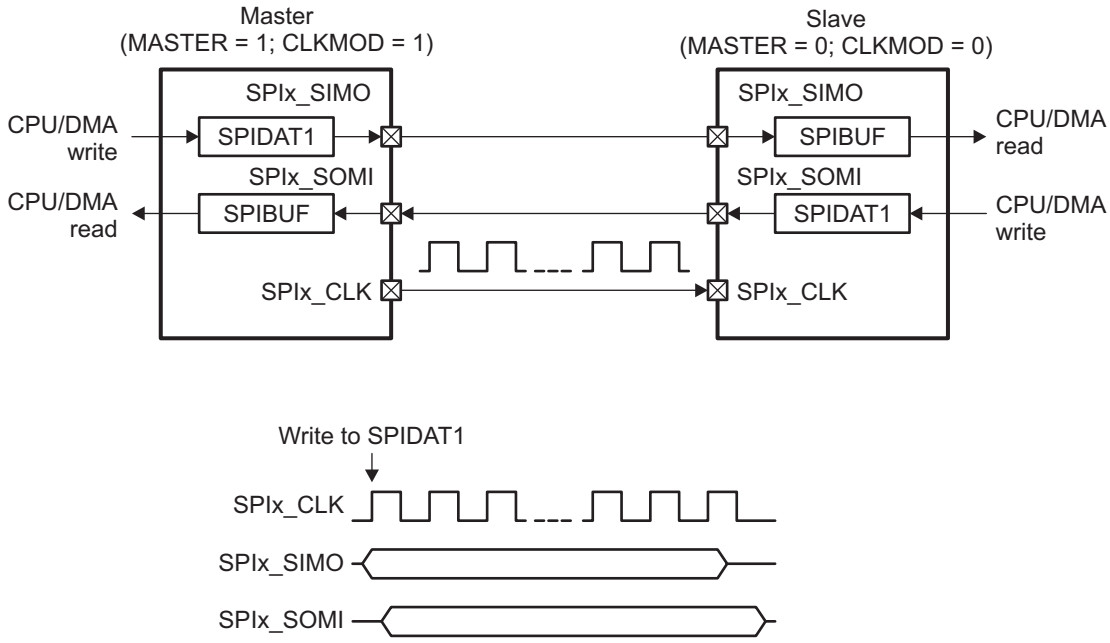
The SPI operates in either master or slave mode. The CLKMOD and MASTER bits in the SPI global control register 1 (SPIGCR1) select between master and slave mode; both must be programmed to 1 to configure the SPI for master mode or to 0 to configure the SPI for slave mode. The SPI bus master is the device that drives the SPIx_CLK signal and initiates SPI bus transfers. In SPI master mode, the SPIx_SOMI pin output buffer is in a high-impedance state and the SPIx_CLK and the SPIx_SIMO pin output buffer is enabled. In SPI slave mode, the SPIx_SIMO and SPIx_CLK pin output buffer is in a high-impedance state and the SPIx_SOMI pin output buffer is enabled.

In master mode with the 3-pin option, the CPU writes transmit data to the SPI transmit data registers (SPIDAT0[15:0] or SPIDAT1[15:0]). This initiates a transfer. A series of clocks pulses will be driven out on the SPIx_CLK pin to complete the transfer. Each clock pulse on the SPIx_CLK pin causes the simultaneous transfer (in both directions) of one bit by both the master and slave SPI devices. CPU writes to the configuration bits in SPIDAT1 (not writing to SPIDAT1[15:0]) do not result in a new transfer. When the selected number of bits has been transmitted, the received data is transferred to the SPI receive buffer register (SPIBUF) for the CPU to read. Data is stored right-justified in SPIBUF.

In slave mode with 3-pin option, CPU writes to SPIDAT0[15:0] or SPIDAT1[15:0] makes the slave ready to transmit. CPU writes to the configuration bits in SPIDAT1 (not writing to SPIDAT1[15:0]) do not make the slave ready to transmit.

NOTE: Either SPIDAT0 or SPIDAT1 can be used on both master and slaves sides.

Figure 27-2. SPI 3-Pin Option



Copyright © 2013–2016, Texas Instruments Incorporated

27.2.8 SPI Operation: 4-Pin with Chip Select Mode

The 4-pin with chip select option is a superset of the 3-pin option and uses the chip select (SPIx_SCS[n]) pin in addition to the clock (SPIx_CLK) and data (SPIx_SOMI and SPIx_SIMO) pins. Figure 27-3 shows the SPI 4-pin chip select option.

To select the 4-pin with chip select option, the SPIx_CLK, SPIx_SOMI, SPIx_SIMO, and SPIx_SCS[n] pins should be configured as functional pins by configuring the SPI pin control register 0 (SPIPC0). The SPIx_ENA pin can be used as a general-purpose I/O pin by configuring the SPIPC1 through SPIPC5 registers.

In SPI master mode, the SPIx_SOMI pin output buffer is in a high-impedance state and the SPIx_CLK, SPIx_SIMO, and SPIx_SCS[n] pin output buffer is enabled. In SPI slave mode, the SPIx_CLK, SPIx_SIMO, and SPIx_SCS[n] pin output buffer is in a high-impedance state, and the SPIx_SOMI pin output buffer is enabled when SPIx_SCS[n] is asserted and in a high-impedance state when SPIx_SCS[n] is deasserted.

In slave mode with the chip select option enabled, the SPI ignores all transactions on the bus unless SPIx_SCS[n] is asserted by the bus master. It also 3-states its output pin when SPIx_SCS[n] is deasserted by the master to avoid conflicting with the active slave device on the bus.

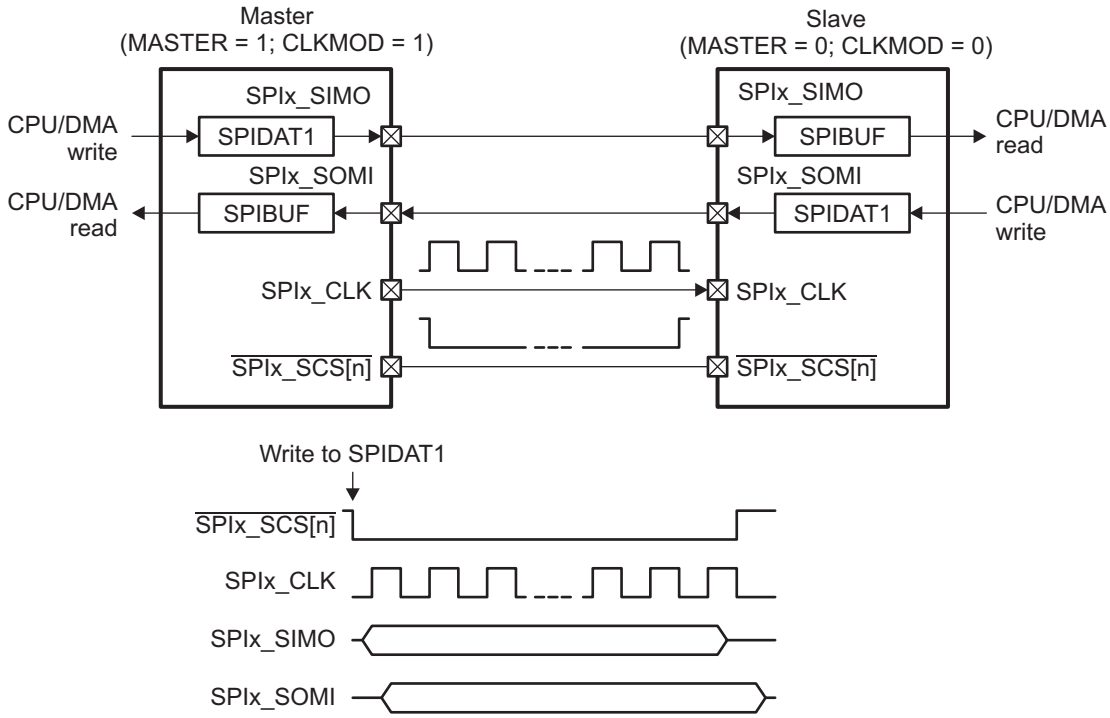
In master mode, the SPIx_SCS[n] pin functions as an output, and toggles when a specific slave device is selected. However, this is most useful on devices that support multiple SPIx_SCS[n] pins.

However, one reason to use the SPIx_SCS[n] pin as a functional pin for the SPI master is to take advantage of the timing parameters that can be set using the SPI delay register (SPIDELAY). The SPIDELAY allows delays to be added automatically so that the slave timing requirements between clock and chip select may be more easily met. Another reason would be to make use of the error detection built into the SPI.

NOTE: Either SPIDAT0 or SPIDAT1 can be used on both master and slaves sides.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 27-3. SPI 4-Pin Option with $\overline{\text{SPIx_SCS[n]}}$



NOTE: During an SPI transfer, if the slave mode SPI detects a deassertion of its chip select even before its internal character length counter overflows, then it 3-states its SPIx_SOMI pin. Once this condition has occurred, if a SPIx_CLK edge is detected while the chip select is deasserted, the SPI stops the transfer and sets an error flag DLENERR (data length) and generates an interrupt if enabled.

Copyright © 2013–2016, Texas Instruments Incorporated

27.2.9 SPI Operation: 4-Pin with Enable Mode

The 4-pin with enable option is a superset of the 3-pin option and uses the enable ($\overline{\text{SPIx_ENA}}$) pin in addition to the clock (SPIx_CLK) and data (SPIx_SOMI and SPIx_SIMO) pins. Figure 27-4 shows the SPI 4-pin enable option.

To select the 4-pin with enable option, the SPIx_CLK, SPIx_SOMI, SPIx_SIMO, and $\overline{\text{SPIx_ENA}}$ pins should be configured as functional pins by configuring the SPI pin control register 0 (SPIPC0). The $\overline{\text{SPIx_SCS[n]}}$ pins can be used as general-purpose I/O pins by configuring the SPIPC1 through SPIPC5 registers.

In SPI master mode, the SPIx_SOMI and $\overline{\text{SPIx_ENA}}$ pin output buffer is in a high-impedance state and the SPIx_CLK and SPIx_SIMO pin output buffer is enabled. In SPI slave mode, the SPIx_CLK and SPIx_SIMO pin output buffer is in a high-impedance state, and the SPIx_SOMI pin output buffer is enabled. In SPI slave mode, the $\overline{\text{SPIx_ENA}}$ pin output buffer enable depends upon the status of the transmit buffer and the configuration of the ENABLEHIGHZ bit in the SPI interrupt register (SPIINT0).

The handshake operation works this way:

- After a transfer completes, both the master and slave SPI modules need to be serviced.
- The slave SPI deasserts $\overline{\text{SPIx_ENA}}$ after the transfer, indicating it requires servicing and is not ready.
- The slave should begin servicing its SPI by first reading receive data from the SPI receive buffer register (SPIBUF).
- Next, the slave device should write transmit data to the SPI transmit data registers (SPIDAT0 or SPIDAT1). This causes the slave SPI to assert $\overline{\text{SPIx_ENA}}$ indicating it is ready for the next transmission.
- In parallel, the master device can service its SPI at any time. It does not need to insert a delay before writing to its SPIDAT0 or SPIDAT1 in order to avoid overrunning the slave device. Instead, the master SPI module will automatically delay the next transfer until the slave has asserted $\overline{\text{SPIx_ENA}}$ again to indicate it is ready for the transmission.

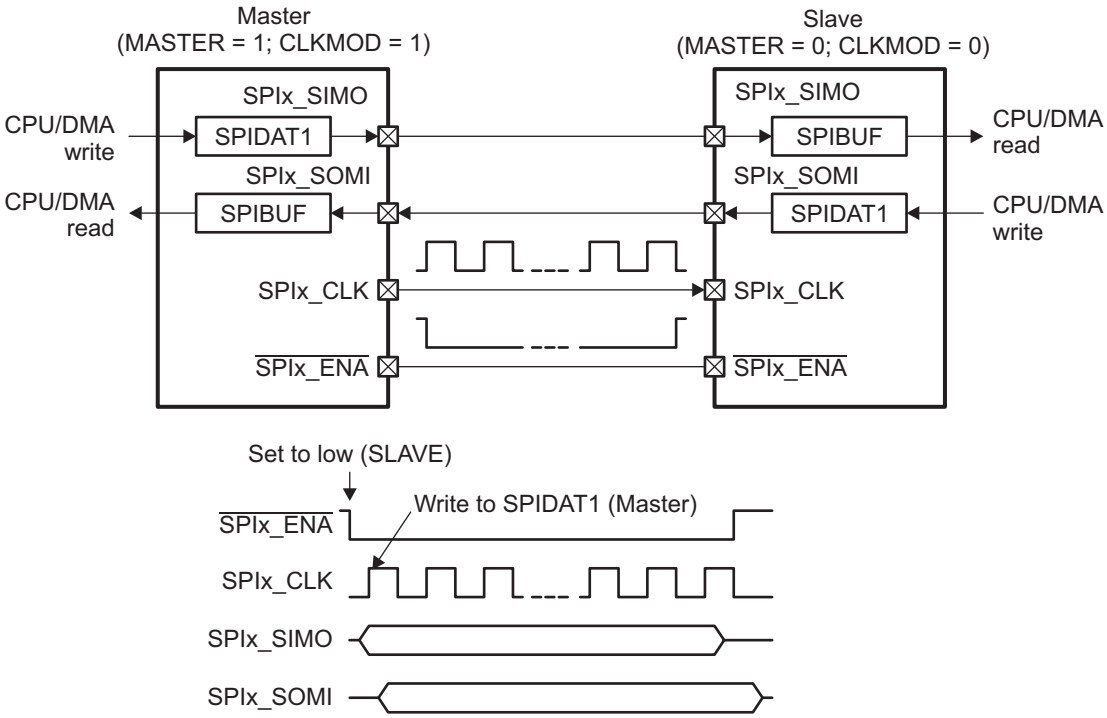
This handshake allows the two SPIs to communicate at the maximum rate possible. Without the handshake pin, the master must insert a delay between each transfer long enough to support the worst case response time of the slave servicing its SPI or risk an overrun condition. With the handshake, the throughput is determined by the average response time of the two devices servicing their SPI ports.

The $\overline{\text{SPIx_ENA}}$ pin can be driven in a push-pull or open-drain mode, depending upon the setting of the ENABLEHIGHZ bit.

NOTE: Either SPIDAT0 or SPIDAT1 can be used on both master and slaves sides.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 27-4. SPI 4-Pin Option with $\overline{\text{SPIx_ENA}}$



Copyright © 2013–2016, Texas Instruments Incorporated

27.2.10 SPI Operation: 5-Pin Mode

The 5-pin mode is a superset of both 4-pin modes. To use the 5-pin mode, both the $\overline{\text{SPIx_ENA}}$ and the $\overline{\text{SPIx_SCS[n]}}$ pins must be configured as functional pins, in addition to the SPIx_CLK, SPIx_SIMO, and SPIx_SOMI pins by configuring the SPI pin control register 0 (SPIPC0). Figure 27-5 shows the SPI 5-pin option.

In SPI master mode, the SPIx_SOMI and $\overline{\text{SPIx_ENA}}$ pin output buffer is in a high-impedance state and the SPIx_CLK, SPIx_SIMO, and $\overline{\text{SPIx_SCS[n]}}$ pin output buffer is enabled. In SPI slave mode, the SPIx_CLK, SPIx_SIMO, and $\overline{\text{SPIx_SCS[n]}}$ pin output buffer is in a high-impedance state, and the SPIx_SOMI pin output buffer is enabled and disabled asynchronously by the $\overline{\text{SPIx_SCS[n]}}$ input and the $\overline{\text{SPIx_ENA}}$ pin output buffer enable depends upon the status of the transmit buffer and the state of the $\overline{\text{SPIx_SCS[n]}}$ input. In SPI slave mode, the assertion of the $\overline{\text{SPIx_ENA}}$ pin by the slave is delayed until the master asserts $\overline{\text{SPIx_SCS[n]}}$, thereby, allowing multiple SPI slaves on a single SPI bus, each slave with its own enable pin.

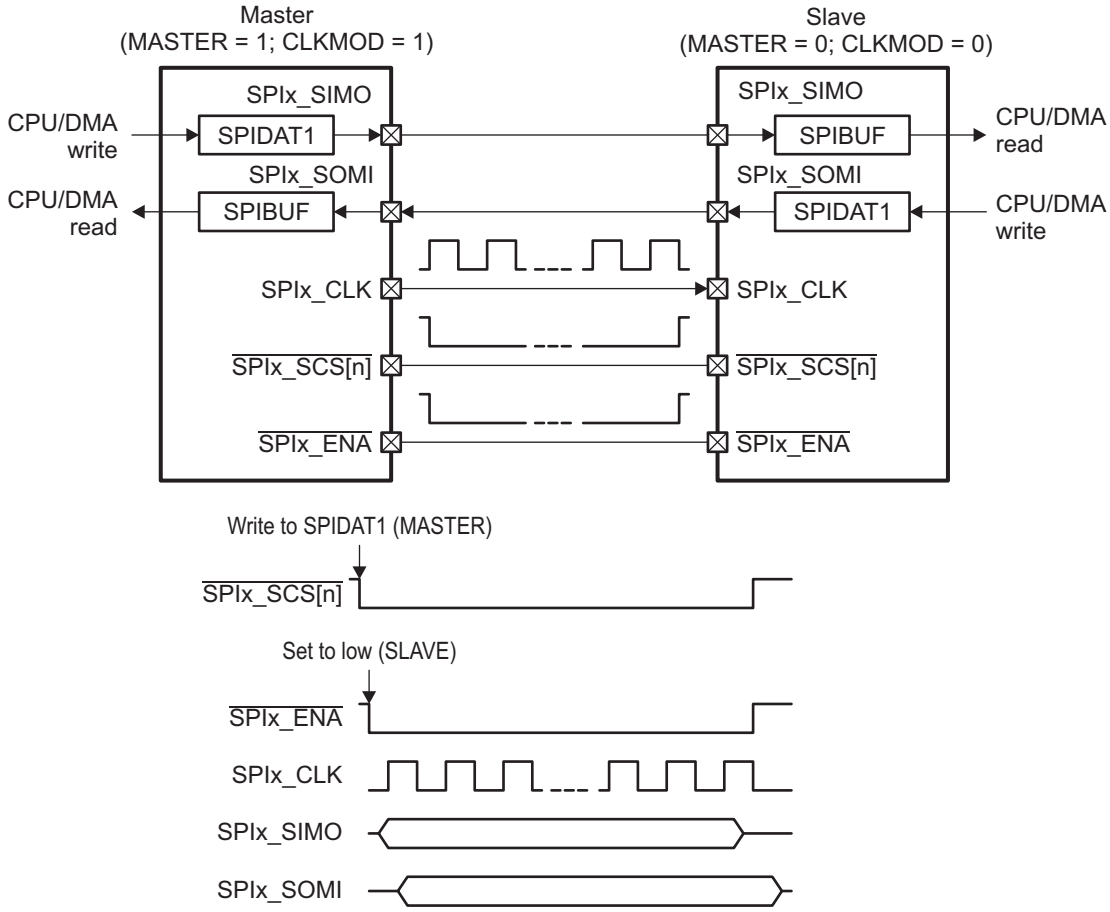
If the $\overline{\text{SPIx_ENA}}$ pin is in high-impedance mode (ENABLEHIGHZ = 1 in the SPI interrupt register (SPIINT0)), the slave SPI will put this signal into the high-impedance state by default. The slave SPI will drive the $\overline{\text{SPIx_ENA}}$ signal low when new data is written to the slave transmit shift register and the slave has been selected by the master ($\overline{\text{SPIx_SCS[n]}}$ is low).

If the $\overline{\text{SPIx_ENA}}$ pin is in push-pull mode (ENABLEHIGHZ = 0), the slave SPI will drive this pin high by default when it is in functional mode. The slave SPI will drive the $\overline{\text{SPIx_ENA}}$ signal low when new data is written to the slave transmit shift register and the slave is selected by the master ($\overline{\text{SPIx_SCS[n]}}$ is low). If the slave is deselected by the master ($\overline{\text{SPIx_SCS[n]}}$ goes high), the slave $\overline{\text{SPIx_ENA}}$ signal is driven high automatically.

NOTE: Either SPIDAT0 or SPIDAT1 can be used on both master and slaves sides.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 27-5. SPI 5-Pin Option with $\overline{\text{SPIx_ENA}}$ and $\overline{\text{SPIx_SCS[n]}}$



NOTE: Push-Pull mode of the $\overline{\text{SPIx_ENA}}$ pin can be used only when there is a single slave in the system. When there are multiple SPI slave devices connected to the common $\overline{\text{SPIx_ENA}}$ pin, all the slaves should configure their $\overline{\text{SPIx_ENA}}$ pins in high-impedance mode.

During an SPI transfer, if slave mode SPI detects a deassertion of its chip select even before its internal character length counter overflows, then it 3-states its SPIx_SOMI and $\overline{\text{SPIx_ENA}}$ (if SPIINT0.ENABLEHIGHZ bit is set to 1) pins. Once this condition has occurred, if a SPIx_CLK edge is detected while the chip select is deasserted, then the SPI stops that transfer and sets an error flag DLENERR (data length) and generates an interrupt if enabled.

Copyright © 2013–2016, Texas Instruments Incorporated

27.2.11 Data Formats

The SPI provides the capability to configure four independent data formats. These formats are configured by programming the corresponding SPI data format registers (SPIFMT*n*). In each data format, the following characteristics of the SPI operation are selected:

- Character length from 2 to 16 bits: The character length is configured by the SPIFMT*n*.CHARLEN field.
- Shift direction (MSB first or LSB first): The shift out direction is configured by the SPIFMT*n*.SHIFTDIR bit.
- Clock polarity: The clock polarity is configured by the SPIFMT*n*.POLARITY bit.
- Clock phase: The clock phase is configured by the SPIFMT*n*.PHASE bit.

The data format is chosen on each transaction. Transmit data is written to the SPI transmit data register 1 (SPIDAT1) and in the same write the data word format select (DFSEL) bit in SPIDAT1 indicates which data format is to be used for the next transaction. Alternatively, the data format can be configured once and applies to all transactions that follow until the data format is changed.

27.2.11.1 Character Length

The character length is configured by the SPIFMT*n*.CHARLEN bit. Legal values are 2 bits (2h) to 16 bits (10h). The character length is independently configured for each of the four data formats; and it must be programmed in both master mode and slave mode.

Transmit data is written to SPIDAT1. The transmit data must be written right-justified irrespective of the character length. The SPI automatically sends out the data correctly based on the chosen data format.

Figure 27-6 shows how a 12-bit word (EC9h) needs to be written to the transmit buffer in order to be transmitted correctly.

Figure 27-6. Format for Transmitting 12-Bit Word

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
x	x	x	x	1	1	1	0	1	1	0	0	1	0	0	1

The data received in SPIBUF is right-justified irrespective of the character length and is padded with 0s when character length is less than 16.

Figure 27-7 shows how a 10-bit word (3A2h) is stored in the buffer once it is received.

Figure 27-7. Format for 10-Bit Received Word

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	1	1	1	0	1	0	0	0	1	0

27.2.11.2 Shift Direction

The shift out direction is configured as most-significant bit (MSB) first or least significant bit (LSB) first. The shift out direction is selected by the SPIFMT*n*.SHIFTDIR bit. The shift out direction is independently configured for each of the four data formats.

- When SPIFMT*n*.SHIFTDIR is 0, the transmit data is shifted out MSB first.
- When SPIFMT*n*.SHIFTDIR is 1, the transmit data is shifted out LSB first.

Copyright © 2013–2016, Texas Instruments Incorporated

27.2.11.3 Clock Phase and Polarity

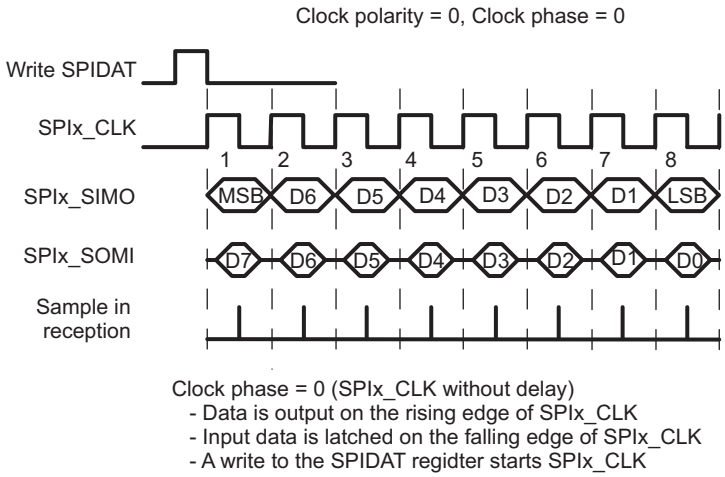
The SPI provides the flexibility to program four different clock mode combinations that SPIx_CLK may operate, enabling a choice of the clock phase (delay or no delay) and the clock polarity (rising edge or falling edge). When operating with PHASE active, the SPI makes the first bit of data available after SPIDAT1 is written and before the first edge of SPIx_CLK. The data input and output edges depend on the values of both the POLARITY and PHASE bits as shown in Table 27-7.

Table 27-7. Clocking Modes

POLARITY	PHASE	Action
0	0	Data is output on the rising edge of SPIx_CLK. Input data is latched on the falling edge.
0	1	Data is output one half-cycle before the first rising edge of SPIx_CLK and on subsequent falling edges. Input data is latched on the rising edge of SPIx_CLK.
1	0	Data is output on the falling edge of SPIx_CLK. Input data is latched on the rising edge.
1	1	Data is output one half-cycle before the first falling edge of SPIx_CLK and on subsequent rising edges. Input data is latched on the falling edge of SPIx_CLK.

Figure 27-8 to Figure 27-11 illustrate the four possible signals of SPIx_CLK corresponding to each mode. Having four signal options allows the SPI to interface with different types of serial devices. Also shown are the SPIx_CLK control bit polarity and phase values corresponding to each signal.

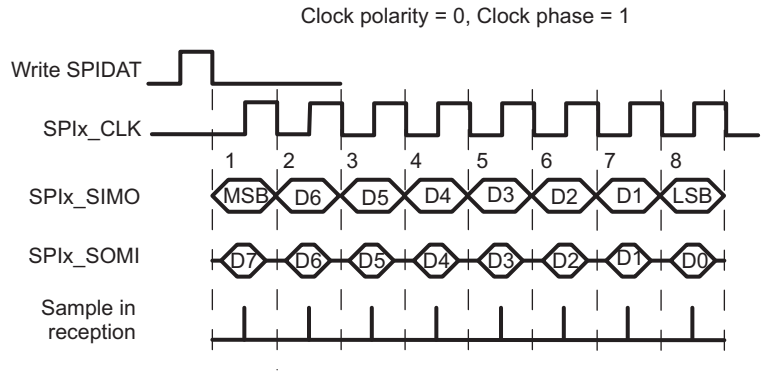
Figure 27-8. Clock Mode with POLARITY = 0 and PHASE = 0



Clock phase = 0 (SPIx_CLK without delay)
- Data is output on the rising edge of SPIx_CLK
- Input data is latched on the falling edge of SPIx_CLK
- A write to the SPIDAT regidter starts SPIx_CLK

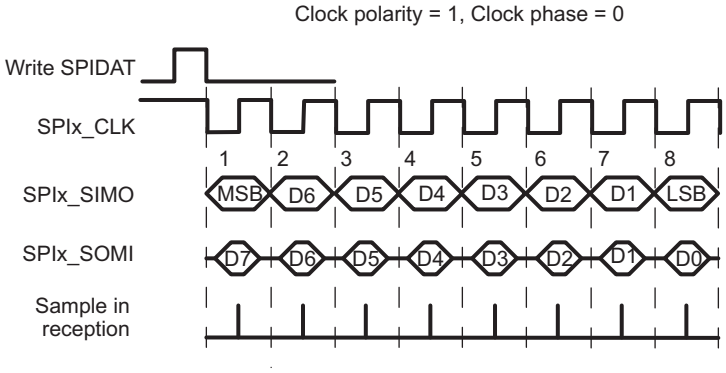
Copyright © 2013–2016, Texas Instruments Incorporated

Figure 27-9. Clock Mode with POLARITY = 0 and PHASE = 1



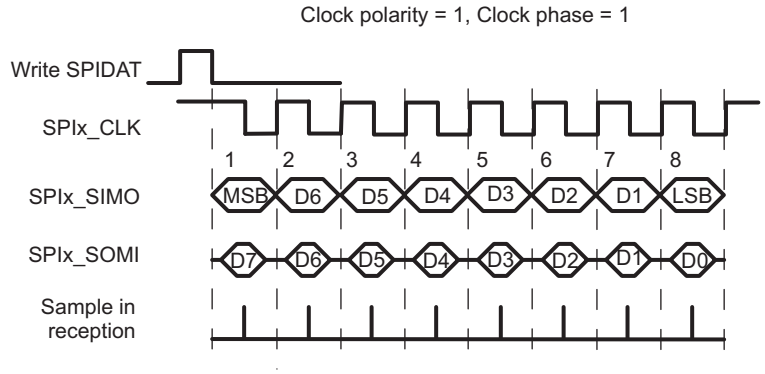
Clock phase = 1 (SPIx_CLK with delay)
- Data is output one-half cycle before the first rising of SPIx_CLK and on subsequent falling edges of SPIx_CLK
- Input data is latched on the rising edge of SPIx_CLK

Figure 27-10. Clock Mode with POLARITY = 1 and PHASE = 0



Clock phase = 0 (SPIx_CLK without delay)
- Data is output on the falling edge of SPIx_CLK
- Input data is latched on the rising edge of SPIx_CLK
- A write to the SPIDAT register starts SPIx_CLK

Figure 27-11. Clock Mode with POLARITY = 1 and PHASE = 1



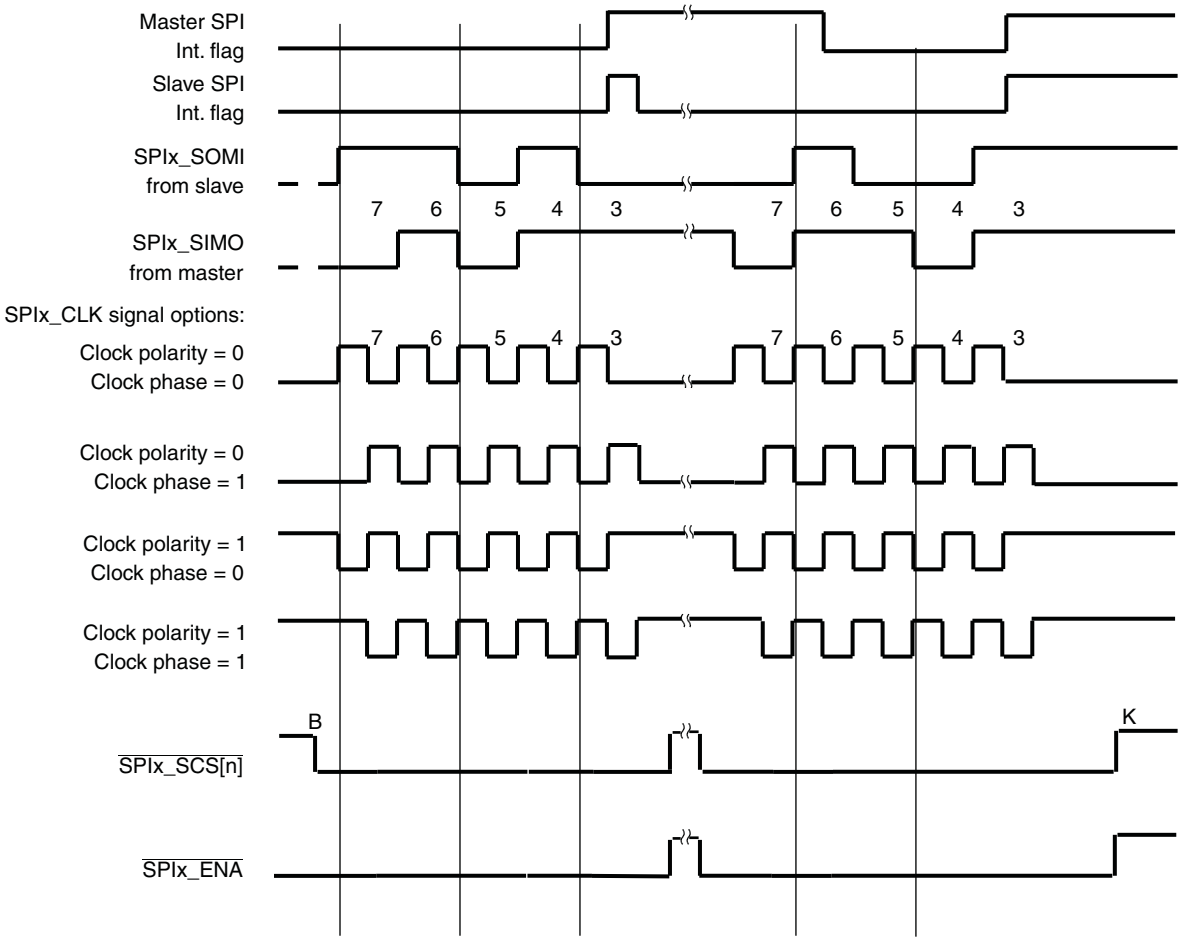
Clock phase = 1 (SPIx_CLK with delay)
- Data is output one-half cycle before the first falling edge of SPIx_CLK and on the subsequent rising edges of SPIx_CLK
- Input data is latched on the falling edge of SPIx_CLK

Copyright © 2013–2016, Texas Instruments Incorporated

27.2.11.4 SPI Data Transfer Example

Figure 27-12 illustrates an SPI data transfer between two devices using a character length of five bits.

Figure 27-12. Five Bits per Character (5-Pin Option)



27.2.12 Interrupt Support

The SPI interrupt system is controlled by three registers:

- The SPI interrupt level register (SPILVL) controls the interrrupt level. The interrupt level must be set to select the level one interrupt (INT1).
- The SPI interrupt register (SPIINT) contains bits to selectively enable/disable each interrupt event.
- The SPI flag register (SPIFLG) contains flags indicating the interrupt conditions that have occurred.

To identify the interrupt source in the SPI peripheral, the CPU reads the SPI flag status register (SPIFLG) or the INTVECT1 code in the SPI interrupt vector register 1 (INTVEC1).

Check your device-specific data manual for details on the exact CPU interrupt numbers assigned to the SPI interrupts.

Copyright © 2013–2016, Texas Instruments Incorporated

27.2.13 DMA Events Support

If handling the SPI message traffic on a character-by-character basis requires too much CPU overhead, then the CPU can configure the system DMA to handle the SPI data transfer.

The SPI module has two DMA synchronization event outputs for receive (REVT) and transmit (XEVT), allowing DMA transfers to be triggered by SPI read receive and write transmit events. The SPI module enables DMA requests by enabling the DMA request enable (DMAREQEN) bit in the SPI interrupt register (SPIINT0).

When a character is to be transmitted the SPI module signals the DMA via the XEVT signal. The DMA controller then transfers the data from the source buffer into the SPI transmit data register (SPIDAT1). When a character is received, the SPI module signals the DMA via the REVT signal. The DMA controller then reads the data from the SPI receive buffer register (SPIBUF) and transfers it to a destination buffer for ready access.

In most cases, if the DMA is being used to service received data from the SPI, the receive interrupt enable (RXINTEN) bit in SPIINT0 should be cleared to 0. This prevents the CPU from responding to the received data in addition to the DMA. For specific SPI synchronization event number assignments and detailed DMA features, see your device-specific data manual.

27.2.14 Robustness Features

The SPI module includes many features to make the SPI communication link robust. A internal loopback test mode can be used to facilitate a power on self test routine. Additionally, the SPI master continually monitors the bus for faults on its data line. The handshaking between master and slave can be monitored as well, and appropriate actions can be taken (interrupt, timeout) when the handshake breaks down. The following sections describe these robustness features in more detail.

27.2.14.1 SPI Internal Loopback Test Mode (Master Only)

CAUTION

The internal loop-back self-test mode should not be entered during a normal data transaction or unpredictable operation may occur.

To select the loopback mode, the SPIx_CLK, SPIx_SOMI, SPIx_SIMO pins should be configured as functional pins by configuring the SPI pin control register 0 (SPIPC0) and by setting the LOOPBACK bit in the SPI global control register 1 (SPIGCR1). The $\overline{\text{SPIx_ENA}}$ and $\overline{\text{SPIx_SCS[n]}}$ pins can be used as general-purpose I/O pins by configuring the SPIPC1 through SPIPC5 registers. The internal loop-back self-test mode can be utilized to test the SPI transmit path and receive path including the transmit and receive buffers. In this mode, the transmit signal is internally fed back to the receiver and the SPIx_SIMO, SPIx_SOMI, and SPIx_CLK pins are in a high-impedance state. This mode allows the CPU to write into the transmit buffer, and check that the receive buffer contains the correct transmit data. If an error occurs the corresponding error is set within the status field.

27.2.14.2 SPI Transmission Continuous Self-Test

During a data transfer, the SPI inputs the value from its data output pin on the appropriate SPIx_CLK edge. This value is compared against the expected value and any difference indicates a fault on the SPI bus. If a fault is detected, then the BITERR bit in the SPI receive buffer register (SPIBUF) and the BITERRFLG bit in the SPI flag register (SPIFLG) are set and an error interrupt is generated if enabled. The SPI continuous self-test mode is not available in SPI loopback mode.

Copyright © 2013–2016, Texas Instruments Incorporated

27.2.14.3 SPI Detection of Slave Desynchronization

In the 4-pin with enable and 5-pin modes, the SPI master can monitor the slave $\overline{\text{SPIx_ENA}}$ activity to detect a desynchronization event.

Some conditions that may cause a desynchronization event are:

- Master or slave device being reset during a transmission.
- Asserting a software reset of the SPI module during transmission.
- Having an incorrect SPI pin configuration, causing the $\overline{\text{SPIx_ENA}}$ pin to behave incorrectly.
- Signal integrity problem causing additional clocks to be recognized by the slave.

The master can detect two desynchronization error conditions on the $\overline{\text{SPIx_ENA}}$ pin:

1. Slave deasserts $\overline{\text{SPIx_ENA}}$ after a transmission has begun, but before it completes.
2. Slave fails to deassert $\overline{\text{SPIx_ENA}}$ within a certain time period after the completion of the last bit of the transmission.

The first error condition is straightforward to detect. To detect the second error condition, the SPI module includes an eight-bit counter with a timeout count that can be configured through the T2EDELAY field in the SPI delay register (SPIDELAY).

When a desynchronization event is detected, the DESYNC bit in the SPI receive buffer register (SPIBUF) and the DESYNCFLG bit in the SPI flag register (SPIFLG) are set and a desynchronization error interrupt is asserted if enabled.

NOTE: Remember that even though the desynchronization is detected by the master device, the problem causing the desynchronization event can be on either the master or the slave device.

The T2EDELAY period begins once the T2CDELAY period terminates or after the data shifting period in case the T2CDELAY is disabled. It defines the maximum time for the slave to deassert the $\overline{\text{SPIx_ENA}}$ signal. If the slave device does not deassert the $\overline{\text{SPIx_ENA}}$ signal before the T2EDELAY timeout value expires, the SPIFLG.DESYNC flag is set and a desynchronization interrupt is asserted if enabled. The T2E delay period does not always complete, sometimes it is skipped or terminated early. The T2E delay period terminates immediately after the $\overline{\text{SPIx_ENA}}$ input is sampled (using the SPI module clock at intervals of SPIFMT*n*.PRESCALE + 2) as deasserted. However, assuming the T2E period completes its duration is specified by:

Maximum duration of T2EDELAY period = SPIDELAY.T2EDELAY + SPIFMTn.PRESCALE + 2 (SPI module clock cycles)

The T2EDELAY period is enabled only when the $\overline{\text{SPIx_ENA}}$ is asserted at the beginning of the T2E delay period, the SPIDELAY.T2EDELAY field has a non-zero value, and SPIFMT*n*.WAITENA bit is set to 1.

27.2.14.4 $\overline{\text{SPIx_ENA}}$ Signal Time-Out

In 5-pin mode, in addition to the slave desynchronization detection, the master can also detect whether the slave fails to respond to the $\overline{\text{SPIx_SCS[n]}}$ signal by asserting $\overline{\text{SPIx_ENA}}$ in a timely manner.

This condition could be the result of a serious error, or it could simply be the result of the slave device taking too long to service its SPI.

To detect this condition, the C2EDELAY field in the SPI delay register (SPIDELAY) is used. The C2EDELAY period begins once the C2TDELAY period terminates or when the master asserts $\overline{\text{SPIx_SCS[n]}}$ (if C2TDELAY is disabled). It defines the maximum time for the addressed slave to respond by activating the $\overline{\text{SPIx_ENA}}$ signal. If the slave does not respond with the $\overline{\text{SPIx_ENA}}$ signal before the timeout value expires, then the TIMEOUT bit in the SPI receive buffer register (SPIBUF) and the TIMEOUTFLG bit in the SPI flag register (SPIFLG) are set, an interrupt is asserted if enabled, and the current transfer is terminated. The C2E delay period does not always complete, sometimes it is skipped or terminated early. The C2E delay period terminates immediately after the $\overline{\text{SPIx_ENA}}$ input is sampled (using the SPI module clock at intervals of SPIFMT*n*.PRESCALE + 2) as asserted. However, assuming the C2E period completes its duration is specified by:

Copyright © 2013–2016, Texas Instruments Incorporated

Maximum duration of C2EDELAY period = SPIDELAY.C2EDELAY + SPIFMTn.PRESCALE + 2 (SPI module clock cycles)

The C2EDELAY period is enabled only when the $\overline{\text{SPIx_ENA}}$ is deasserted at the beginning of the C2E delay period and SPIFMT*n*.WAITENA bit is set to 1. If SPIFMT*n*.WAITENA bit is set to 1 and C2EDELAY is cleared to 0, then the master waits indefinitely for the slave to assert $\overline{\text{SPIx_ENA}}$.

27.2.14.5 SPI Data Length Error

An SPI can generate an error flag by detecting any mismatch in length of received/transmitted data with the programmed character length under certain conditions.

Master Mode: During a data transfer, if the SPI detects a deassertion of the $\overline{\text{SPIx_ENA}}$ pin (by the slave) while the character counter is not overflowed, then an error flag is set indicating the data length error. This can be caused by a slave receiving extra clocks (because of noise on the SPIx_CLK line).

NOTE: In SPI master mode, the data length error will be generated only if the $\overline{\text{SPIx_ENA}}$ pin is used as a functional pin.

Slave Mode: During a transfer, if the SPI detects a deassertion of the $\overline{\text{SPIx_SCS[n]}}$ pin before its character length counter overflows, then an error flag is set indicating the data length error. If the slave SPI misses one or more SPIx_CLK pulses from the master, this situation can occur. This error in slave mode would mean that both the transmitted and received data were not complete.

NOTE: In SPI slave mode, the data length error flag will be generated only if the $\overline{\text{SPIx_SCS[n]}}$ pin is configured as a functional pin.

27.2.15 Reset Considerations

This section describes the software and hardware reset considerations.

27.2.15.1 Software Reset Considerations

The SPI module contains a software reset (RESET) bit in the SPI global control register 0 (SPIGCR0) that is used to reset the SPI module. As a result of a reset, the SPI module register values go to their reset state. The RESET bit must be set before any operation on the SPI is done.

27.2.15.2 Hardware Reset Considerations

In the event of a hardware reset, the SPI module register values go to their reset state and the application software needs to reprogram the registers to the desired values.

27.2.16 Power Management

The SPI module can be put in either local or global low-power mode. Global low-power mode is asserted by the system and is not controlled by the SPI. During global low-power mode, all clocks to the SPI are turned off so the module is completely inactive.

The SPI local low-power mode is asserted by setting the POWERDOWN bit in the SPI global control register 1 (SPIGCR1). Setting this bit stops the clocks to the SPI internal logic and the SPI registers. Setting the POWERDOWN bit causes the SPI to enter local low-power mode and clearing the POWERDOWN bit causes SPI to exit from local low-power mode. All the registers are accessible during local power-down mode as any register access enables the clock to SPI for that particular access alone.

Since entering a low-power mode has the effect of suspending all state machine activities, care must be taken when entering such modes to ensure that a valid state is entered when low-power mode is active. As a result, application software must ensure that a low-power mode is not entered during a transmission or reception of data.

Copyright © 2013–2016, Texas Instruments Incorporated

27.2.17 General-Purpose I/O Pin

Each of the SPI pins may be programmed via the SPI pin control registers (SPIPC0 to SPIPC5) to be a general-purpose I/O (GPIO) pin.

When the SPI pins are not used as functional pins, they may be programmed to be either general input or general output pins by configuring SPIPC0. For example, in 3-pin mode, SPIx_SOMI, SPIx_SIMO, and SPIx_CLK must be configured as SPI pins, while the SPIx_SCS[n] and SPIx_ENA pins should be configured as GPIO pins. The direction is controlled by configuring SPIPC1.

If configured as a general-purpose output, then SPIPC3 controls the output value. There is also a write 1 to set (SPIPC4) and a write 1 to clear (SPIPC5) for the data out value. These registers allow different tasks running on the CPU to manipulate the SPI I/O pins without read-modify-write hazards.

SPIPC2 reflects the current value on the pin when the particular pin is configured as a functional or general-purpose input pin. When the pin is configured as a functional or general-purpose output pin, SPIPC2 indicates the value that is attempted to be driven on the pin.

27.2.18 Emulation Considerations

CAUTION

Viewing or otherwise reading the following SPI registers: SPIBUF, SPIFLG, and INTVEC1 through the JTAG debugger causes their contents to change, possibly invalidating the results of the debug session. Be sure to set up the debugger to avoid reading these registers.

The SPI module does not support soft or hard stop during emulation breakpoints. The SPI module will continue to run if an emulation breakpoint is encountered.

In addition, any status registers that are cleared after reading will be affected if viewed in a memory or watch window of the debugger; since the emulator will read these registers to update the value displayed in the window.

27.2.19 Initialization

Perform the following procedure for initializing the SPI:

1. Reset the SPI by clearing the RESET bit in the SPI global control register 0 (SPIGCR0) to 0.
2. Take the SPI out of reset by setting SPIGCR0.RESET to 1.
3. Configure the SPI for master or slave mode by configuring the CLKMOD and MASTER bits in the SPI global control register 1 (SPIGCR1).
4. Configure the SPI for 3-pin, 4-pin with chip select, 4-pin with enable, or 5-pin mode by configuring the SPI pin control register 0 (SPIPC0).
5. Chose the SPI data format register *n* (SPIFMT*n*) to be used by configuring the DFSEL bit in the SPI transmit data register (SPIDAT1). In slave mode, only SPIFMT0 is supported.
6. Configure the SPI data rate, character length, shift direction, phase, polarity and other format options using SPIFMT*n* selected in step 5.
7. If SPI master, then configure the master delay options using the SPI delay register (SPIDELAY). In slave mode, SPIDELAY is not relevant.
8. Select the error interrupt notifications by configuring the SPI interrupt register (SPIINT0) and the SPI interrupt level register (SPILVL).
9. Enable the SPI communication by setting the SPIGCR1.ENABLE to 1.
10. Setup and enable the DMA for SPI data handling and then enable the DMA servicing for the SPI data requests by setting the SPIINT0.DMAREQEN to 1.
11. Handle SPI data transfer requests using DMA and service any SPI error conditions using the interrupt service routine.

Copyright © 2013–2016, Texas Instruments Incorporated

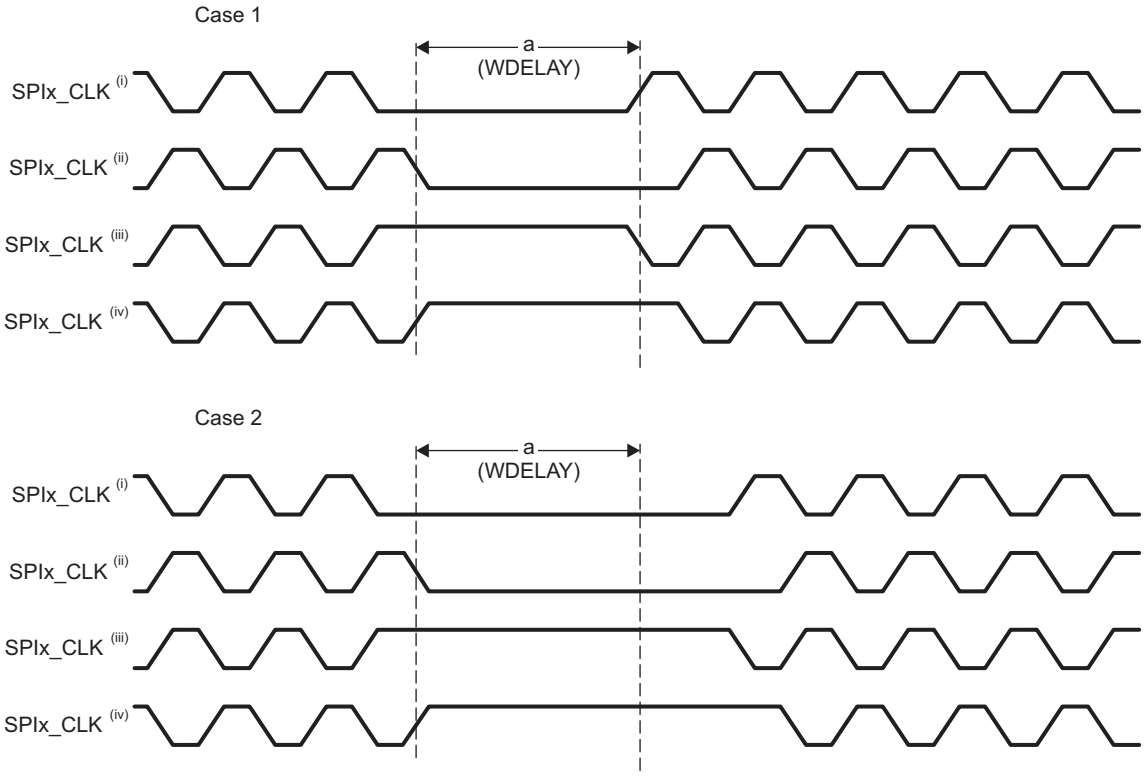
27.2.20 Timing Diagrams

This section contains timing diagrams illustrating the C2TDELAY, C2EDELAY, T2CDELAY, T2EDELAY, and WDELAY delays and their interaction with the SPIx_SCS[n] and SPIx_ENA pins for all SPI modes.

27.2.20.1 SPI 3-Pin Mode

Figure 27-13 illustrates the WDELAY option in SPI 3-pin master mode. This is the only delay available in this mode. In CASE1, a new transfer is initiated during the WDELAY period and the transfer begins immediately after the WDELAY period ends. In CASE2, while WDELAY has completed, a new transfer will not begin until SPIDAT0/SPIDAT1 have been written with new data.

Figure 27-13. SPI 3-Pin Master Mode with WDELAY

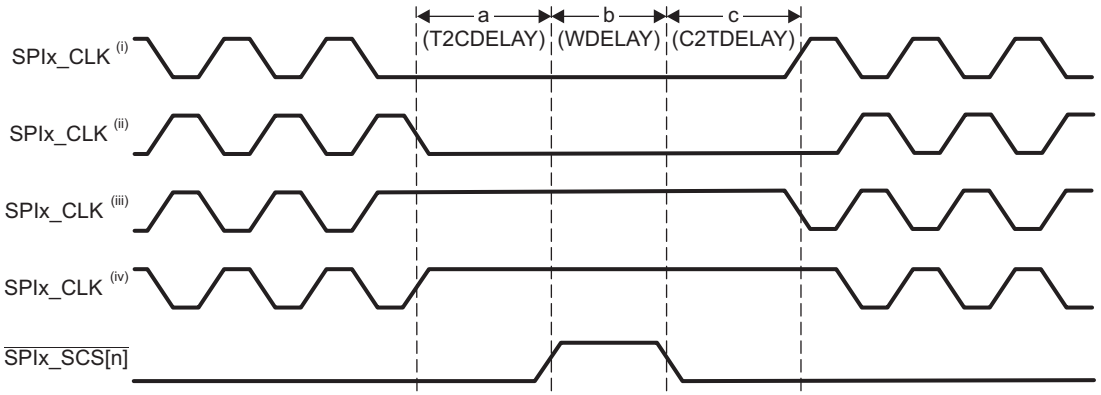


Copyright © 2013–2016, Texas Instruments Incorporated

27.2.20.2 SPI 4-Pin with $\overline{\text{SPIx_SCS[n]}}$ Mode

Figure 27-14 illustrates the T2CDELAY, WDELAY and C2TDELAY delays in SPI 4-pin with $\overline{\text{SPIx_SCS[n]}}$ master mode. C2EDELAY and T2EDELAY are not available in this mode. All the three delay periods T2CDELAY, WDELAY, and C2TDELAY proceed to completion when enabled.

Figure 27-14. SPI 4-Pin with $\overline{\text{SPIx_SCS[n]}}$ Mode with T2CDELAY, WDELAY, and C2TDELAY



27.2.20.3 SPI 4-Pin with $\overline{\text{SPIx_ENA}}$ Mode

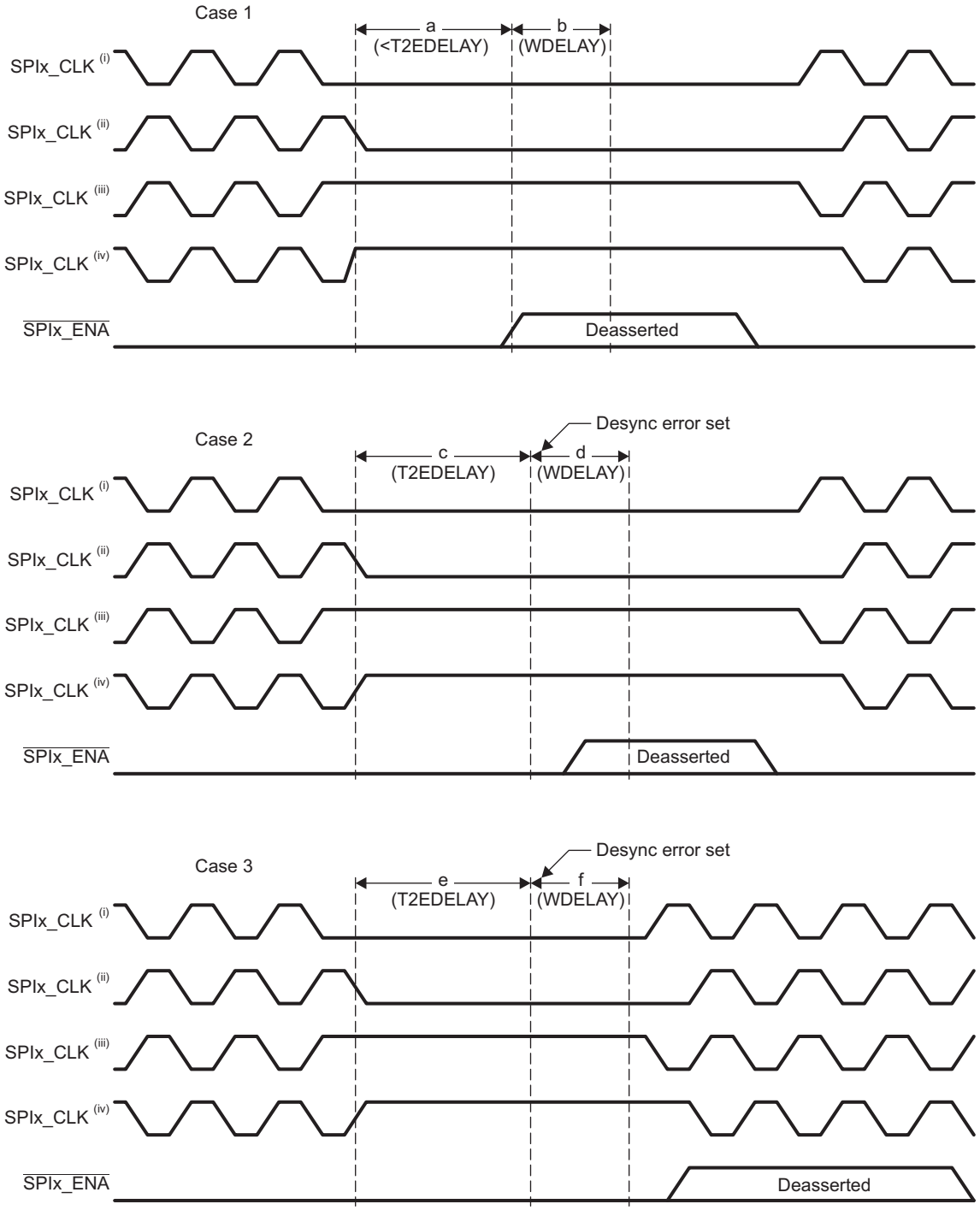
Figure 27-15 shows the T2EDELAY and WDELAY delays in SPI 4-pin with $\overline{\text{SPIx_ENA}}$ master mode. T2CDELAY, C2TDELAY, and C2EDELAY are not available in this mode.

- In CASE1, the $\overline{\text{SPIx_ENA}}$ is deasserted during the T2EDELAY period. Consequently the T2EDELAY period is terminated early (a) and the WDELAY period begins immediately (b) if enabled. The next transfer is initiated as soon as the slave asserts $\overline{\text{SPIx_ENA}}$ again.
- In CASE2, the T2EDELAY period (c) completes before the $\overline{\text{SPIx_ENA}}$ is deasserted. As a result the DESYNC error is set. However since the $\overline{\text{SPIx_ENA}}$ is deasserted during the WDELAY period (d), the master delays the next transfer until the $\overline{\text{SPIx_ENA}}$ is asserted again.
- In CASE3, the T2EDELAY (e) and WDELAY (f) period (if enabled) both expire before the $\overline{\text{SPIx_ENA}}$ input is deasserted. The DESYNC error is set at the end of the T2EDELAY period (e). However in this case the master begins the next transfer immediately after it is initiated and ignores the $\overline{\text{SPIx_ENA}}$ during the transfer even if it is subsequently deasserted.

  If the T2EDELAY delay period is disabled then the DESYNC error is not set. The SPI master behavior in this case depends on whether the $\overline{\text{SPIx_ENA}}$ gets deasserted during the WDELAY period (CASE2) or $\overline{\text{SPIx_ENA}}$ gets deasserted after the WDELAY period completes (CASE3).

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 27-15. SPI 4-Pin with $\overline{\text{SPIx_ENA}}$ Mode Demonstrating T2EDELAY and WDELAY



Copyright © 2013–2016, Texas Instruments Incorporated

27.2.20.4 SPI 5-Pin Mode

Figure 27-16 shows the T2CDELAY, T2EDELAY, and WDELAY delays in SPI 5-pin master mode.

- In CASE1, the SPIx_ENA is deasserted during the T2CDELAY period. However the T2CDELAY period proceeds to completion(a), the T2EDELAY period is skipped (if enabled) and the WDELAY period begins immediately (b) (if enabled). The next transfer is initiated as soon as the slave asserts SPIx_ENA again.
- In CASE2, the SPIx_ENA signal is deasserted by the slave during the T2EDELAY period (d) which begins upon the completion of the T2CDELAY period (c). The deassertion of the SPIx_ENA causes the T2EDELAY period to terminate early and the WDELAY period (e) begins immediately (if enabled) after the T2EDELAY period terminates. The next transfer is initiated as soon as the slave asserts SPIx_ENA again.
- In CASE3, the SPIx_ENA signal is deasserted by the slave during the WDELAY period (h) which begins upon the completion of the T2CDELAY period (f) and T2EDELAY period (g). As a result the DESYNC error is set at the end of the T2EDELAY period (g). However since the SPIx_ENA is deasserted during the WDELAY period (h), the master delays the next transfer until the SPIx_ENA is asserted again.
- In CASE4, the SPIx_ENA signal is not deasserted until after the completion of the T2CDELAY (j), T2EDELAY (k) and WDELAY (m) (if enabled) periods. The DESYNC error is set at the end of the T2EDELAY period (k). However in this case the master begins the next transfer immediately after it is initiated and ignores the SPIx_ENA during the transfer even if it is subsequently deasserted.

  If the T2EDELAY delay period is disabled then the DESYNC error is not set. The SPI master behavior in this case depends on whether the SPIx_ENA gets deasserted during the T2CDELAY period (CASE1), WDELAY period (CASE3) or after the WDELAY period completes (CASE4).

  If the slave deasserts the SPIx_ENA signal before the completion of the configured master delays (T2CDELAY, T2EDELAY, WDELAY) then the master delays the next transfer until the slave asserts the SPIx_ENA again. However if the slave delays the SPIx_ENA deassertion until after the completion of the configured master delays then the master begins the next transfer immediately after it is initiated and ignores the SPIx_ENA during the transfer even if it is subsequently deasserted.

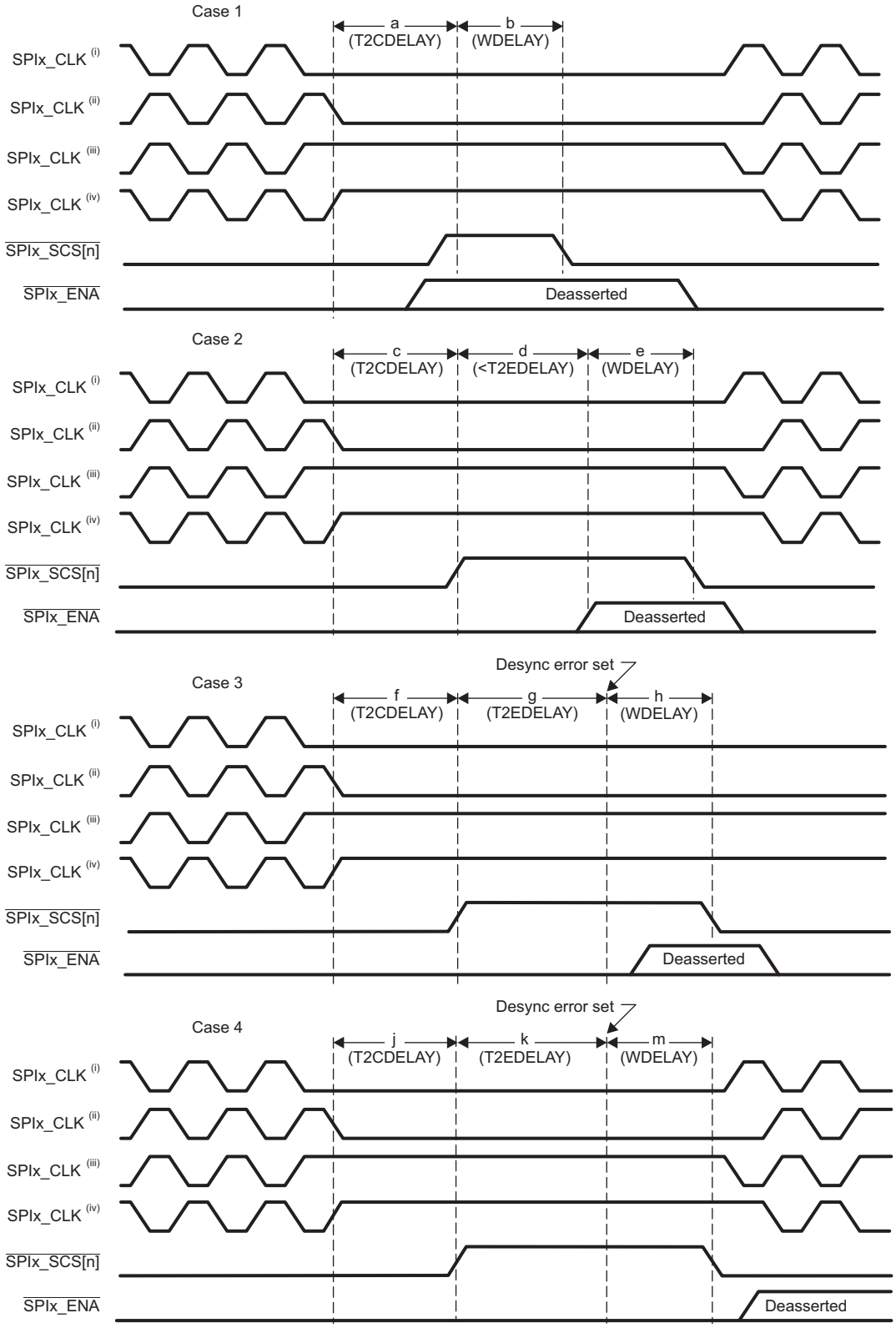
Figure 27-17 shows the C2TDELAY and C2EDELAY in SPI 5-pin master mode.

- In CASE1, the SPIx_ENA signal is asserted during the C2TDELAY period (a). However the C2TDELAY period proceeds to completion(a), the C2EDELAY period is skipped (if enabled) and the master begins generating the SPI clock for transmission.
- In CASE2, the SPIx_ENA signal is asserted during the C2EDELAY period (d) which begins upon the completion of C2TDELAY period (c). The assertion of the SPIx_ENA causes the C2EDELAY period to terminate early and the master begins generating the SPI clock for transmission.
- In CASE3, the SPIx_ENA signal is not asserted until after the completion of the C2TDELAY (f) and C2EDELAY (g) periods. The TIMEOUT error is set at the end of the C2EDELAY period (g). The master deasserts the SPIx_SCS[n] signal immediately and clears the current transmit request.

  If the C2EDELAY delay period is disabled then the SPI master behavior depends on whether the SPIx_ENA gets asserted during the C2TDELAY period (CASE1) or after the C2TDELAY period completes (CASE2). In latter case there is no limit on how long the master will wait for the slave to respond with SPIx_ENA asserted and hence there is no limit on period 'd' shown in CASE2. Thus when C2EDELAY period is disabled the TIMEOUT error is not set.

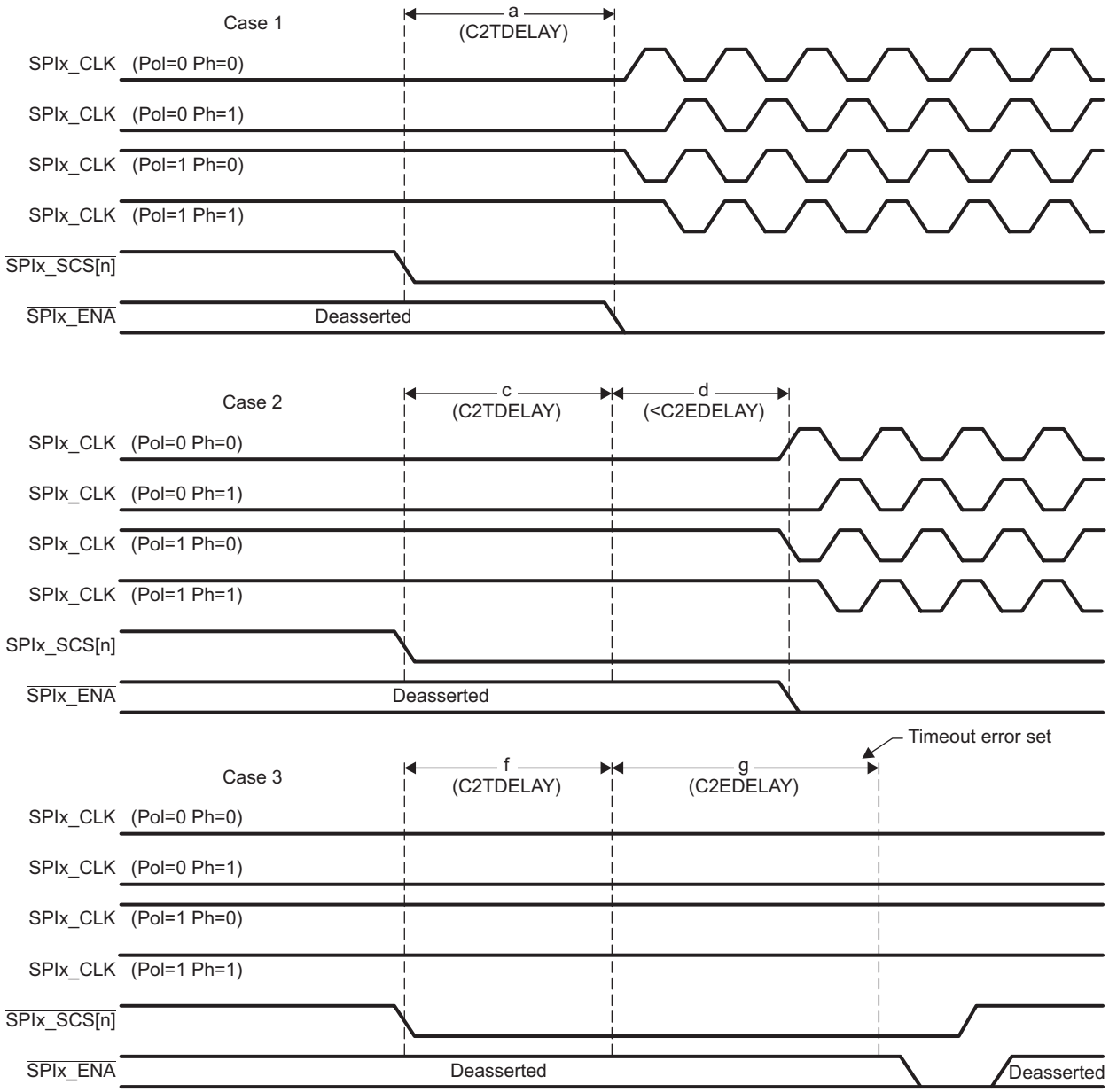
Copyright © 2013–2016, Texas Instruments Incorporated

Figure 27-16. SPI 5-Pin Mode Demonstrating T2CDELAY, T2EDELAY, and WDELAY



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 27-17. SPI 5-Pin Mode Demonstrating C2TDELAY and C2EDELAY



Copyright © 2013–2016, Texas Instruments Incorporated

27.3 Registers

This section describes the SPI control, data, and pin registers. The offset is relative to the associated base address of the module. See your device-specific data manual for the memory address of these registers.

Table 27-8. SPI Registers

Offset Address	Acronym	Register Description	Section
0h	SPIGCR0	SPI Global Control Register 0	Section 27.3.1
4h	SPIGCR1	SPI Global Control Register 1	Section 27.3.2
8h	SPIINT0	SPI Interrupt Register	Section 27.3.3
Ch	SPILVL	SPI Interrupt Level Register	Section 27.3.4
10h	SPIFLG	SPI Flag Register	Section 27.3.5
14h	SPIPC0	SPI Pin Control Register 0 (Function)	Section 27.3.6
18h	SPIPC1	SPI Pin Control Register 1 (Direction)	Section 27.3.7
1Ch	SPIPC2	SPI Pin Control Register 2 (Input)	Section 27.3.8
20h	SPIPC3	SPI Pin Control Register 3 (Output)	Section 27.3.9
24h	SPIPC4	SPI Pin Control Register 4 (Set SPIPC3)	Section 27.3.10
28h	SPIPC5	SPI Pin Control Register 5 (Clear SPIPC3)	Section 27.3.11
38h	SPIDAT0	SPI Data Transmit Register 0	Section 27.3.12
3Ch	SPIDAT1	SPI Data Transmit Register 1 (Data Transmit and Format Select)	Section 27.3.13
40h	SPIBUF	SPI Receive Buffer Register	Section 27.3.14
44h	SPIEMU	SPI Receive Emulation Register	Section 27.3.15
48h	SPIDELAY	SPI Delay Register	Section 27.3.16
4Ch	SPIDEF	SPI Default Chip Select Register	Section 27.3.17
50h	SPIFMT0	SPI Data Format Register 0	Section 27.3.18
54h	SPIFMT1	SPI Data Format Register 1	Section 27.3.18
58h	SPIFMT2	SPI Data Format Register 2	Section 27.3.18
5Ch	SPIFMT3	SPI Data Format Register 3	Section 27.3.18
64h	INTVEC1	SPI Interrupt Vector Register 1	Section 27.3.19

27.3.1 SPI Global Control Register 0 (SPIGCR0)

The SPI global control register 0 (SPIGCR0) is shown in Figure 27-18 and described in Table 27-9.

Figure 27-18. SPI Global Control Register 0 (SPIGCR0)

31	16
Reserved	
R-0	

15	1	0
Reserved		RESET
R-0		R/W-0

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

Table 27-9. SPI Global Control Register 0 (SPIGCR0) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return zero and writes have no effect.
0	RESET		Reset bit for the module. This bit needs to be set to 1 before any operation on SPI can be done.
		0	SPI is in reset state.
		1	SPI is out of reset state.

Copyright © 2013–2016, Texas Instruments Incorporated

27.3.2 SPI Global Control Register 1 (SPIGCR1)

The SPI global control register 1 (SPIGCR1) is shown in Figure 27-19 and described in Table 27-10.

Figure 27-19. SPI Global Control Register 1 (SPIGCR1)

31	25	24
Reserved		ENABLE
R-0		R/W-0

23	17	16
Reserved		LOOPBACK
R-0		R/W-0

15	9	8
Reserved		POWERDOWN
R-0		R/W-0

7	2	1	0
Reserved		CLKMOD	MASTER
R-0		R/W-0	R/W-0

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

Table 27-10. SPI Global Control Register 1 (SPIGCR1) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return zero and writes have no effect.
24	ENABLE		SPI enable. This bit enables the SPI transfers. The other SPI configuration registers except SPIINT0.DMAREQEN should be configured before writing a 1 to this bit. This will prevent the SPI from responding to bus operations erroneously while it is in the process of being configured. The SPIINT0.DMAREQEN should be enabled after setting ENABLE. If SPIINT0.DMAREQEN is enabled before setting ENABLE then the first DMA request that occurs before the SPI is ready for data transfer may get dropped. When ENABLE bit is cleared to 0, the following SPI registers get forced to their default states (to 0s except for RXEMPTY bit in SPIBUF): • Both TX and RX shift registers • The TXDATA fields of SPIDAT0 and SPIDAT1 registers • All the fields of the SPIFLG register • Contents of SPIBUF and the internal RXBUF registers
		0	SPI is not activated for transfers.
		1	Activates SPI.
23-17	Reserved	0	Reads return zero and writes have no effect.
16	LOOPBACK		Internal loop-back test mode. The internal self-test option can be enabled by setting this bit. If the SPIx_SIMO and SPIx_SOMI pins are configured with SPI functionality, then the SPIx_SIMO pin is internally connected to the SPIx_SOMI pin. The transmit data is looped back as receive data and is stored in the receive field of the concerned buffer. Externally, during loop-back operation, the SPIx_CLK pin outputs an inactive value, SPIx_SIMO and SPIx_SOMI pins remain in high-impedance state. The SPI has to be initialized in master mode before the loop-back can be selected. If the SPI is initialized in slave mode or a data transfer is ongoing, errors may result.
		0	Internal loop-back test mode disabled.
		1	Internal loop-back test mode enabled.
15-9	Reserved	0	Reads return zero and writes have no effect.
8	POWERDOWN		When active, the SPI state machine enters a power-down state.
		0	The SPI is in active mode.
		1	The SPI is in power-down mode.
7-2	Reserved	0	Reads return zero and writes have no effect.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 27-10. SPI Global Control Register 1 (SPIGCR1) Field Descriptions (continued)

Bit	Field	Value	Description
1-0	CLKMOD,MASTER	0-3h	These two bits (CLKMOD,MASTER) determine whether the SPI operates in master or slave mode.
		0	SLAVE MODE. SPIx_CLK is an input from the master who initiates the transfers. Data is transmitted on the SPIx_SOMI pin and received on the SPIx_SIMO pin. The $\overline{\text{SPIx_SCS[n]}}$ pin is an input pin if configured as SPI slave chip select. The $\overline{\text{SPIx_ENA}}$ pin is an output pin if configured as the SPI enable pin.
		1h-2h	Reserved
		3h	MASTER MODE. SPIx_CLK is an output and the SPI initiates transfers. Data is transmitted on the SPIx_SIMO pin and received on the SPIx_SOMI pin. The $\overline{\text{SPIx_SCS[n]}}$ pin is an output pin if configured as SPI slave chip select. The $\overline{\text{SPIx_ENA}}$ pin is an input pin if configured as the SPI enable pin.

Copyright © 2013–2016, Texas Instruments Incorporated

27.3.3 SPI Interrupt Register (SPIINT0)

The SPI interrupt register (SPIINT0) is shown in Figure 27-20 and described in Table 27-11.

Figure 27-20. SPI Interrupt Register (SPIINT0)

31	25	24
Reserved		ENABLEHIGHZ
R-0		R/W-0

23	17	16
Reserved		DMAREQEN
R-0		R/W-0

15	10	9	8
Reserved		TXINTENA	RXINTENA
R-0		R/W-0	R/W-0

7	6	5	4	3	2	1	0
Reserved	OVRNINTENA	Reserved	BITERRENA	DESYNCENA	PARERRENA	TIMEOUTENA	DLENERRENA
R-0	R/W-0	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 27-11. SPI Interrupt Register (SPIINT0) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return zero and writes have no effect.
24	ENABLEHIGHZ		$\overline{\text{SPIx_ENA}}$ pin high-impedance enable. If ENABLEHIGHZ is enabled, the $\overline{\text{SPIx_ENA}}$ pin (when it is configured as a WAIT functional output signal in a slave SPI) is forced to place it is output in high-impedance when not driving a low signal. If ENABLEHIGHZ is disabled, then the pin will output both a high and a low signal.
		0	$\overline{\text{SPIx_ENA}}$ pin is pulled high when not active.
		1	$\overline{\text{SPIx_ENA}}$ pin remains in high-impedance when not active.
23-17	Reserved	0	Reads return zero and writes have no effect.
16	DMAREQEN		DMA request enable. Enables the DMA request signal to be generated for both receive and transmit channels. Set DMAREQEN only after setting the SPIGCR1.ENABLE bit to 1.
		0	DMA is not used.
		1	DMA requests will be generated.
			Note: A transmit DMA request will be generated each time a transmit data is copied to the shift register either from TXBUF or directly from SPIDAT0/SPIDAT1.
			Note: A receive DMA request will be generated each time a received data is copied to SPIBUF register either from RXBUF or directly from the shift register.
15-10	Reserved	0	Reads return zero and writes have no effect.
9	TXINTENA		An interrupt is to be generated every time data is written to the shift register, so that a new data can be written to TXBUF. Setting this bit will generate an interrupt if the SPIFLG.TXINTFLG bit is set to 1.
		0	No interrupt will be generated upon SPIFLG.TXINTFLG being set to 1.
		1	Interrupt will be generated upon SPIFLG.TXINTFLG being set to 1.
8	RXINTENA		Receive interrupt enable. An interrupt is to be generated when the SPIFLG.RXINTFLAG bit is set.
		0	Interrupt will not be generated.
		1	Interrupt will be generated.
7	Reserved	0	Reads return zero and writes have no effect.
6	OVRNINTENA		Overrun interrupt enable. An interrupt is to be generated when the SPIFLG.OVRNINTFLG bit is set. The overrun interrupt is not useful if receive data is serviced with CPU interrupts because the overrun and receive events share a common level interrupt signal.
		0	Overrun interrupt will not be generated.
		1	Overrun interrupt will be generated.
5	Reserved	0	Reads return zero and writes have no effect.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 27-11. SPI Interrupt Register (SPIINT0) Field Descriptions (continued)

Bit	Field	Value	Description
4	BITERRENA		Enables interrupt on bit error. An interrupt is to be generated when the SPIFLG.BITERRFLG is set.
		0	No interrupt asserted upon bit error.
		1	Enables an interrupt on a bit error.
3	DESYNCENA		Enables interrupt on desynchronized slave. DESYNCENA is used in master mode only. The desynchronization monitor is active in master mode for the 4-pin with enable and 5-pin options. An interrupt is to be generated when the SPIFLG.DESYNCFLG is set.
		0	No interrupt asserted upon desynchronization error.
		1	Enables an interrupt on desynchronization of the slave.
2	PARERRENA		Enables interrupt on parity error. An interrupt is to be generated when the SPIFLG.PARERRFLG is set.
		0	No interrupt asserted upon parity error.
		1	Enables an interrupt on a parity error.
1	TIMEOUTENA		Enables interrupt on $\overline{\text{SPIx_ENA}}$ signal time-out. An interrupt is to be generated when SPIFLG.TIMEOUTFLG is set.
		0	No interrupt asserted upon $\overline{\text{SPIx_ENA}}$ signal time-out.
		1	Enables an interrupt on a time-out of the $\overline{\text{SPIx_ENA}}$ signal.
0	DLENERRENA		Data length error interrupt enable. A data length error occurs under the following conditions. Master: In a 4-pin with $\overline{\text{SPIx_ENA}}$ mode or 5-pin mode, if the $\overline{\text{SPIx_ENA}}$ pin from the slave is deasserted before the master has completed its transfer, the data length error is set. That is, if the character length counter has not overflowed while $\overline{\text{SPIx_ENA}}$ deassertion is detected, then it means that the slave has neither received full data from the master nor has it transmitted complete data. Slave: In a 4-pin with chip select mode or 5-pin mode, if the incoming valid $\overline{\text{SPIx_SCS}}$[n] pin is deactivated before the character length counter overflows, then data length error is set.
		0	No interrupt is generated upon data length error.
		1	Enables an interrupt when data length error occurs.

Copyright © 2013–2016, Texas Instruments Incorporated

27.3.4 SPI Interrupt Level Register (SPILVL)

The SPI interrupt level register (SPILVL) is shown in Figure 27-21 and described in Table 27-12.

Figure 27-21. SPI Interrupt Level Register (SPILVL)

31	16
Reserved	
R-0	

15	10	9	8
Reserved		TXINTLVL	RXINTLVL
R-0		R/W-0	R/W-0

7	6	5	4	3	2	1	0
Reserved	OVRNINTLVL	Reserved	BITERRLVL	DESYNCLVL	PARERRLVL	TIMEOUTLVL	DLENERRLVL
R-0	R/W-0	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 27-12. SPI Interrupt Level Register (SPILVL) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return zero and writes have no effect.
9	TXINTLVL		Transmit interrupt level.
		0	Reserved
		1	Transmit interrupt is mapped to interrupt line INT1.
8	RXINTLVL		Receive interrupt level.
		0	Reserved
		1	Receive interrupt is mapped to interrupt line INT1.
7	Reserved	0	Reads return zero and writes have no effect.
6	OVRNINTLVL		Receive overrun interrupt level. The overrun interrupt is not useful if receive data is serviced with CPU interrupts because the overrun and receive events share a common level interrupt signal.
		0	Reserved
		1	Receive overrun interrupt is mapped to interrupt line INT1.
5	Reserved	0	Reads return zero and writes have no effect.
4	BITERRLVL		Bit error interrupt level.
		0	Reserved
		1	Bit error interrupt is mapped to interrupt line INT1.
3	DESYNCLVL		Desynchronized slave interrupt level. DESYNCLVL is used in master mode only.
		0	Reserved
		1	An interrupt due to desynchronization of the slave is mapped to interrupt line INT1.
2	PARERRLVL		Parity error interrupt level.
		0	Reserved
		1	A parity error interrupt is mapped to interrupt line INT1.
1	TIMEOUTLVL		$\overline{\text{SPIx_ENA}}$ signal time-out interrupt level.
		0	Reserved
		1	An interrupt on a time-out of the $\overline{\text{SPIx_ENA}}$ signal is mapped to interrupt line INT1.
0	DLENERRLVL		Data length error interrupt enable level.
		0	Reserved
		1	An interrupt on data length error is mapped to interrupt line INT1.

Copyright © 2013–2016, Texas Instruments Incorporated

27.3.5 SPI Flag Register (SPIFLG)

The SPI flag register (SPIFLG) is shown in Figure 27-22 and described in Table 27-13.

Figure 27-22. SPI Flag Register (SPIFLG)

31							16
Reserved							
R-100h							
15					10	9	8
Reserved						TXINTFLG	RXINTFLG
R-0						R-0	R/WC-0
7	6	5	4	3	2	1	0
Reserved	OVRNINTFLG	Reserved	BITERRFLG	DESYNCFLG	PARERRFLG	TIMEOUTFLG	DLENERRFLG
R-0	R/W1C-0	R-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear bit; -*n* = value after reset

Table 27-13. SPI Flag Register (SPIFLG) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	4000h	Reads return default value and writes have no effect.
9	TXINTFLG		Transmitter empty interrupt flag. Serves as an interrupt flag indicating that the transmit buffer (TXBUF) is empty and a new data can be written to it. This flag is set when a data is copied to the shift register either directly or from the TXBUF register. This bit is cleared by one of following ways: • Writing a new data to either SPIDAT0 or SPIDAT1 • Writing a 0 to SPIGCR1.ENABLE
		0	Transmit buffer is now full. No interrupt pending for transmitter empty.
		1	Transmit buffer is empty. An interrupt is pending to fill the transmitter.
8	RXINTFLG		Receiver full interrupt flag. This flag is set when a word is received and copied into the buffer register (SPIBUF). This bit is cleared under the following ways: • Reading the SPIBUF register. During emulation mode, however, a read to the emulation register (SPIEMU) does not clear this flag bit. • Reading INTVEC1 register when there is a receive buffer full interrupt • Writing a 1 to this bit • Writing a 0 to SPIGCR1.ENABLE • System reset
		0	No new received data pending. Receive buffer is empty.
		1	A newly received data is ready to be read. Receive buffer is full.
			Note: Clearing RXINTFLG bit by writing a 1 before reading the SPIBUF sets the RXEMPTY bit of the SPIBUF register too. This way, one can ignore a received data. However, if the internal RXBUF is already full, the data from RXBUF will be copied to SPIBUF and the RXEMPTY bit will be cleared again. The SPIBUF contents should be read first if this situation needs to be avoided.
7	Reserved	0	Reads return zero and writes have no effect.
6	OVRNINTFLG		Receiver overrun flag. The bit is set when a receive operation completes before the previous character has been read from the receive buffer. The bit indicates that the last received character has been overwritten and therefore lost. This bit is cleared under the following conditions: • Reading INTVEC1 register when there is a receive buffer overrun interrupt • Writing a 1 to this bit
		0	Overrun condition did not occur.
		1	Overrun condition has occurred.
			Note: Reading SPIBUF register does not clear the OVRNINTFLG bit. If an overrun interrupt is detected, then the SPIBUF may need to be read twice to get to the overrun buffer. This is due to the fact that the overrun will always occur to the internal RXBUF. Each read to the SPIBUF will result in RXBUF contents (if it is full) getting copied to SPIBUF.
			Note: A special condition under which OVRNINTFLG flag gets set. If both SPIBUF and RXBUF are already full and while another buffer receive is underway, if any errors like TIMEOUT, BITERR and DLENERR occur, then OVRNINTFLG will be set to indicate that the status flags are getting overwritten by the new transfer. This overrun should be treated like a normal receiver overrun.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 27-13. SPI Flag Register (SPIFLG) Field Descriptions (continued)

Bit	Field	Value	Description
5	Reserved	0	Reads return zero and writes have no effect.
4	BITERRFLG		This bit is set when a mismatch of internal transmit data and transmitted data is detected. The SPI samples the signal of the transmit pin (master: SPIx_SIMO, slave: SPIx_SOMI) at the receive point (half clock cycle after transmit point). If the sampled value differs from the transmitted value a bit error is detected and the flag is set. A possible reason for a bit error can be a too high bit rate/capacitive load or another master/slave trying to transmit at the same time. This flag can be cleared by one of the following ways: • Write a 1 to this bit. • Set SPIGCR1.ENABLE bit to 0.
		0	No bit error occurred.
		1	A bit error occurred.
3	DESYNCFLG		Desynchronization of slave device. Desynchronization monitor is active in master mode only. The master monitors the $\overline{\text{SPIx_ENA}}$ signal coming from the slave device and sets the DESYNCFLG bit if the $\overline{\text{SPIx_ENA}}$ signal is not deasserted after the last bit is transmitted plus $t_{T2EDELAY}$. Desynchronization can occur if a slave device misses a clock edge coming from the master. This flag can be cleared by one of the following ways: • Write a 1 to this bit. • Set SPIGCR1.ENABLE bit to 0.
		0	No slave desynchronization detected.
		1	Slave is desynchronized
			Note: Inconsistency of DESYNCFLG in SPI. Due to the nature of this error, under some circumstances it is possible for a desynchronized error detected for the previous buffer to be visible in the current buffer. This is due to the fact that receive completion flag/interrupt will be generated when the buffer transfer is completed. But decence will be detected after the buffer transfer is completed. So, if CPU/DMA reads the received data quickly when an receive interrupt is detected, then the status flag may not reflect the correct decence condition.
2	PARERRFLG		Calculated parity differs from received parity bit. If the parity generator is enabled an even or odd parity bit is added at the end of a data word . During reception of the data word the parity generator calculates the reference parity and compares it to the received parity bit. In the event of a mismatch the PARERRFLG flag is set. This flag can be cleared by one of the following ways: • Write a 1 to this bit. • Set SPIGCR1.ENABLE bit to 0.
		0	No parity error detected.
		1	A parity error occurred.
1	TIMEOUTFLG		Time-out due to non-activation of $\overline{\text{SPIx_ENA}}$ signal. This flag is applicable only for the master mode. The SPI generates a time-out because the slave hasn't responded in time by activating the $\overline{\text{SPIx_ENA}}$ signal after the chip select signal has been activated. If a time-out condition is detected the corresponding chip select is deactivated immediately and the TIMEOUTFLG flag is set. This flag can be cleared by one of the following ways: • Write a 1 to this bit. • Set SPIGCR1.ENABLE bit to 0.
		0	No $\overline{\text{SPIx_ENA}}$ signal time-out occurred.
		1	An $\overline{\text{SPIx_ENA}}$ signal time-out occurred.
0	DLENERRFLG		Data length error flag. This flag can be cleared by one of the following ways: • Write a 1 to this bit. • Set SPIGCR1.ENABLE bit to 0.
		0	No data length error has occurred.
		1	A data length error has occurred.

Copyright © 2013–2016, Texas Instruments Incorporated

27.3.6 SPI Pin Control Register 0 (SPIPC0)

The SPI pin control register 0 (SPIPC0) is shown in Figure 27-23 and described in Table 27-14.

Figure 27-23. SPI Pin Control Register 0 (SPIPC0)

31	16
Reserved	
R-0	

15	12	11	10	9	8
Reserved		SOMIFUN	SIMOFUN	CLKFUN	ENAFUN
R-0		R/W-0	R/W-0	R/W-0	R/W-0

7	0
SCS0FUN[*n*](1)	
R/W-0	

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

(1) Not all devices support multiple slave chip select ($\overline{\text{SPIx_SCS[n]}}$) pins, see your device-specific data manual for supported pins. If the pins are not available, the corresponding bit is reserved and should be cleared to 0.

Table 27-14. SPI Pin Control Register 0 (SPIPC0) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reads return zero and writes have no effect.
11	SOMIFUN		Slave out, master in pin function. This bit determines whether the SPIx_SOMI pin is to be used as a general-purpose I/O pin or as a SPI functional pin.
		0	SPIx_SOMI pin is a GPIO pin.
		1	SPIx_SOMI pin is a SPI functional pin.
10	SIMOFUN		Slave in, master out pin function. This bit determines whether the SPIx_SIMO pin is to be used as a general-purpose I/O pin or as a SPI functional pin.
		0	SPIx_SIMO pin is a GPIO pin.
		1	SPIx_SIMO pin is a SPI functional pin.
9	CLKFUN		SPI clock pin function. This bit determines whether the SPIx_CLK pin is to be used as a general-purpose I/O pin, or as a SPI functional pin.
		0	SPIx_CLK pin is a GPIO pin.
		1	SPIx_CLK pin is a SPI functional pin.
8	ENAFUN		SPI enable pin function. This bit determines whether the $\overline{\text{SPIx_ENA}}$ pin is to be used as a general-purpose I/O pin, or as a SPI functional pin.
		0	$\overline{\text{SPIx_ENA}}$ pin is a GPIO pin.
		1	$\overline{\text{SPIx_ENA}}$ pin is a SPI functional pin.
7-0	SCS0FUN[*n*]		SPI chip select pin *n* function. This bit determines whether the $\overline{\text{SPIx_SCS[n]}}$ pin is to be used as a general-purpose I/O pin, or as a SPI functional pin. Not all devices support multiple slave chip select ($\overline{\text{SPIx_SCS[n]}}$) pins, see your device-specific data manual for supported pins. If the pins are not available, the corresponding bit is reserved and should be cleared to 0.
		0	$\overline{\text{SPIx_SCS[n]}}$ pin is a GPIO pin.
		1	$\overline{\text{SPIx_SCS[n]}}$ pin is a SPI functional pin.

Copyright © 2013–2016, Texas Instruments Incorporated

27.3.7 SPI Pin Control Register 1 (SPIPC1)

The SPI pin control register 1 (SPIPC1) is shown in Figure 27-24 and described in Table 27-15.

Figure 27-24. SPI Pin Control Register 1 (SPIPC1)

31	16
Reserved	
R-0	

15	12	11	10	9	8
Reserved		SOMIDIR	SIMODIR	CLKDIR	ENADIR
R-0		R/W-0	R/W-0	R/W-0	R/W-0

7	0
SCS0DIR[*n*](1)	
R/W-0	

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

(1) Not all devices support multiple slave chip select ($\overline{\text{SPIx_SCS[n]}}$) pins, see your device-specific data manual for supported pins. If the pins are not available, the corresponding bit is reserved and should be cleared to 0.

Table 27-15. SPI Pin Control Register 1 (SPIPC1) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reads return zero and writes have no effect.
11	SOMIDIR		SPIx_SOMI pin direction. Controls the direction of the SPIx_SOMI pin when it is used as a general-purpose I/O pin. If the SPIx_SOMI pin is used as a SPI functional pin, the I/O direction is determined by whether the SPI is configured as master or slave.
		0	SPIx_SOMI pin is an input.
		1	SPIx_SOMI pin is an output.
10	SIMODIR		SPIx_SIMO pin direction. Controls the direction of the SPIx_SIMO pin when it is used as a general-purpose I/O pin. If the SPIx_SIMO pin is used as a SPI functional pin, the I/O direction is determined by whether the SPI is configured as master or slave.
		0	SPIx_SIMO pin is an input.
		1	SPIx_SIMO pin is an output.
9	CLKDIR		SPIx_CLK pin direction. Controls the direction of the SPIx_CLK pin when it is used as a general-purpose I/O pin. If the SPIx_CLK pin is used as a SPI functional pin, the I/O direction is determined by whether the SPI is configured as master or slave.
		0	SPIx_CLK pin is an input.
		1	SPIx_CLK pin is an output.
8	ENADIR		$\overline{\text{SPIx_ENA}}$ pin direction. Controls the direction of the $\overline{\text{SPIx_ENA}}$ pin when it is used as a general-purpose I/O pin. If the $\overline{\text{SPIx_ENA}}$ pin is used as a SPI functional pin, then the I/O direction is determined by whether the SPI is configured as master or slave.
		0	$\overline{\text{SPIx_ENA}}$ pin is an input.
		1	$\overline{\text{SPIx_ENA}}$ pin is an output.
7-0	SCS0DIR[*n*]		$\overline{\text{SPIx_SCS[n]}}$ pin direction. Controls the direction of the $\overline{\text{SPIx_SCS[n]}}$ pin when it is used as a general-purpose I/O pin. If the $\overline{\text{SPIx_SCS[n]}}$ pin is used as a SPI functional pin, then the I/O direction is determined by whether the SPI is configured as master or slave. Not all devices support multiple slave chip select ($\overline{\text{SPIx_SCS[n]}}$) pins, see your device-specific data manual for supported pins. If the pins are not available, the corresponding bit is reserved and should be cleared to 0.
		0	$\overline{\text{SPIx_SCS[n]}}$ pin is an input.
		1	$\overline{\text{SPIx_SCS[n]}}$ pin is an output.

Copyright © 2013–2016, Texas Instruments Incorporated

27.3.8 SPI Pin Control Register 2 (SPIPC2)

The SPI pin control register 2 (SPIPC2) is shown in Figure 27-25 and described in Table 27-16.

Figure 27-25. SPI Pin Control Register 2 (SPIPC2)

31	16
Reserved	
R-0	

15	12	11	10	9	8
Reserved		SOMIDIN	SIMODIN	CLKDIN	ENADIN
R-0		R-U	R-U	R-U	R-U

7	0
SCS0DIN[*n*](1)	
R-U	

LEGEND: R = Read only; U = Undefined; -*n* = value after reset

(1) Not all devices support multiple slave chip select ($\overline{\text{SPIx_SCS[n]}}$) pins, see your device-specific data manual for supported pins. If the pins are not available, the corresponding bit is reserved and should be cleared to 0.

Table 27-16. SPI Pin Control Register 2 (SPIPC2) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reads return zero and writes have no effect.
11	SOMIDIN		SPIx_SOMI data in. This bit reflects the value of the SPIx_SOMI pin.
		0	Current value of SPIx_SOMI pin is logic 0.
		1	Current value of SPIx_SOMI pin is logic 1.
10	SIMODIN		SPIx_SIMO data in. This bit reflects the value of the SPIx_SIMO pin.
		0	Current value of SPIx_SIMO pin is logic 0.
		1	Current value of SPIx_SIMO pin is logic 1.
9	CLKDIN		Clock data in. This bit reflects the value of the SPIx_CLK pin.
		0	Current value of SPIx_CLK pin is logic 0.
		1	Current value of SPIx_CLK pin is logic 1.
8	ENADIN		$\overline{\text{SPIx_ENA}}$ data in. This bit reflects the value of the $\overline{\text{SPIx_ENA}}$ pin.
		0	Current value of $\overline{\text{SPIx_ENA}}$ pin is logic 0.
		1	Current value of $\overline{\text{SPIx_ENA}}$ pin is logic 1.
7-0	SCS0DIN[*n*]		$\overline{\text{SPIx_SCS[n]}}$ data in. This bit reflects the value of the $\overline{\text{SPIx_SCS[n]}}$ pin. Not all devices support multiple slave chip select ($\overline{\text{SPIx_SCS[n]}}$) pins, see your device-specific data manual for supported pins. If the pins are not available, the corresponding bit is reserved and should be cleared to 0.
		0	Current value of $\overline{\text{SPIx_SCS[n]}}$ pin is logic 0.
		1	Current value of $\overline{\text{SPIx_SCS[n]}}$ pin is logic 1.

Copyright © 2013–2016, Texas Instruments Incorporated

27.3.9 SPI Pin Control Register 3 (SPIPC3)

The SPI pin control register 3 (SPIPC3) is shown in Figure 27-26 and described in Table 27-17.

Figure 27-26. SPI Pin Control Register 3 (SPIPC3)

31	16
Reserved	
R-0	

15	12	11	10	9	8
Reserved		SOMIDOUT	SIMODOUT	CLKDOUT	ENADOUT
R-0		R/W-0	R/W-0	R/W-0	R/W-0

7	0
SCS0DOUT[*n*](1)	
R/W-0	

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

(1) Not all devices support multiple slave chip select ($\overline{\text{SPIx_SCS[n]}}$) pins, see your device-specific data manual for supported pins. If the pins are not available, the corresponding bit is reserved and should be cleared to 0.

Table 27-17. SPI Pin Control Register 3 (SPIPC3) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reads return zero and writes have no effect.
11	SOMIDOUT		SPIx_SOMI data out write. This bit is only active when the SPIx_SOMI pin is configured as a general-purpose I/O pin and configured as an output pin. The value of this bit indicates the value sent to the pin.
		0	Current value of SPIx_SOMI pin is logic 0.
		1	Current value of SPIx_SOMI pin is logic 1.
10	SIMODOUT		SPIx_SIMO data out write. This bit is only active when the SPIx_SIMO pin is configured as a general-purpose I/O pin and configured as an output pin. The value of this bit indicates the value sent to the pin.
		0	Current value of SPIx_SIMO pin is logic 0.
		1	Current value of SPIx_SIMO pin is logic 1.
9	CLKDOUT		SPIx_CLK data out write. This bit is only active when the SPIx_CLK pin is configured as a general-purpose I/O pin and configured as an output pin. The value of this bit indicates the value sent to the pin.
		0	Current value of SPIx_CLK pin is logic 0.
		1	Current value of SPIx_CLK pin is logic 1.
8	ENADOUT		$\overline{\text{SPIx_ENA}}$ data out write. Only active when the $\overline{\text{SPIx_ENA}}$ pin is configured as a general-purpose I/O pin and configured as an output pin. The value of this bit indicates the value sent to the pin.
		0	Current value of $\overline{\text{SPIx_ENA}}$ pin is logic 0.
		1	Current value of $\overline{\text{SPIx_ENA}}$ pin is logic 1.
7-0	SCS0DOUT[*n*]		$\overline{\text{SPIx_SCS[n]}}$ data out write. Only active when the $\overline{\text{SPIx_SCS[n]}}$ pin is configured as a general-purpose I/O pin and configured as an output pin. The value of this bit indicates the value sent to the pin *n*. Not all devices support multiple slave chip select ($\overline{\text{SPIx_SCS[n]}}$) pins, see your device-specific data manual for supported pins. If the pins are not available, the corresponding bit is reserved and should be cleared to 0.
		0	Current value of $\overline{\text{SPIx_SCS[n]}}$ pin is logic 0.
		1	Current value of $\overline{\text{SPIx_SCS[n]}}$ pin is logic 1.

Copyright © 2013–2016, Texas Instruments Incorporated

27.3.10 SPI Pin Control Register 4 (SPIPC4)

The SPI pin control register 4 (SPIPC4) is shown in Figure 27-27 and described in Table 27-18.

Figure 27-27. SPI Pin Control Register 4 (SPIPC4)

31	16
Reserved	
R-0	

15	12	11	10	9	8
Reserved		SOMISET	SIMOSET	CLKSET	ENASET
R-0		R/W-0	R/W-0	R/W-0	R/W-0

7	0
SCS0SET[*n*](1)	
R/W-0	

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

(1) Not all devices support multiple slave chip select ($\overline{\text{SPIx_SCS[n]}}$) pins, see your device-specific data manual for supported pins. If the pins are not available, the corresponding bit is reserved and should be cleared to 0.

Table 27-18. SPI Pin Control Register 4 (SPIPC4) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reads return zero and writes have no effect.
11	SOMISET		SPIx_SOMI data out set. This bit is only active when the SPIx_SOMI pin is configured as a general-purpose output pin. Reads return the value of the SPIx_SOMI pin.
		Write 0	No effect
		Write 1	SPIPC3.SOMIDOUT is set to 1.
10	SIMOSET		SPIx_SIMO data out set. This bit is only active when the SPIx_SIMO pin is configured as a general-purpose output pin. Reads return the value of the SPIx_SIMO pin.
		Write 0	No effect
		Write 1	SPIPC3.SIMODOUT is set to 1.
9	CLKSET		SPIx_CLK data out set. This bit is only active when the SPIx_CLK pin is configured as a general-purpose output pin. Reads return the value of the SPIx_CLK pin.
		Write 0	No effect
		Write 1	SPIPC3.CLKDOUT is set to 1.
8	ENASET		$\overline{\text{SPIx_ENA}}$ data out set. This bit is only active when the $\overline{\text{SPIx_ENA}}$ pin is configured as a general-purpose output pin. Reads return the value of the $\overline{\text{SPIx_ENA}}$ pin.
		Write 0	No effect.
		Write 1	SPIPC3.ENADOUT is set to 1.
7-0	SCS0SET[*n*]		$\overline{\text{SPIx_SCS[n]}}$ data out set. This bit is only active when the $\overline{\text{SPIx_SCS[n]}}$ pin is configured as a general-purpose output pin. Reads return the value of the $\overline{\text{SPIx_SCS[n]}}$ pin. Not all devices support multiple slave chip select ($\overline{\text{SPIx_SCS[n]}}$) pins, see your device-specific data manual for supported pins. If the pins are not available, the corresponding bit is reserved and should be cleared to 0.
		Write 0	No effect
		Write 1	SPIPC3.SCS0DOUT[*n*] is set to 1.

Copyright © 2013–2016, Texas Instruments Incorporated

27.3.11 SPI Pin Control Register 5 (SPIPC5)

The SPI pin control register 5 (SPIPC5) is shown in Figure 27-28 and described in Table 27-19.

Figure 27-28. SPI Pin Control Register 5 (SPIPC5)

31	16
Reserved	
R-0	

15	12	11	10	9	8
Reserved		SOMICLR	SIMOCLR	CLKCLR	ENACLR
R-0		R/W-0	R/W-0	R/W-0	R/W-0

7	0
SCS0CLR[[*n*] (1)	
R/W-0	

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

(1) Not all devices support multiple slave chip select ($\overline{\text{SPIx_SCS[n]}}$) pins, see your device-specific data manual for supported pins. If the pins are not available, the corresponding bit is reserved and should be cleared to 0.

Table 27-19. SPI Pin Control Register 5 (SPIPC5) Field Descriptions

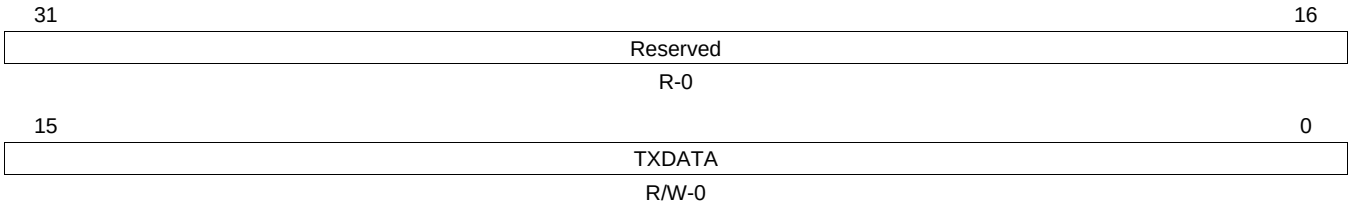
Bit	Field	Value	Description
31-12	Reserved	0	Reads return zero and writes have no effect.
11	SOMICLR		SPIx_SOMI data out clear. This bit is only active when the SPIx_SOMI pin is configured as a general-purpose output pin. Reads return the value of the SPIx_SOMI pin.
		Write 0	No effect.
		Write 1	SPIPC3.SOMIDOUT is cleared to 0.
10	SIMOCLR		SPIx_SIMO data out clear. This bit is only active when the SPIx_SIMO pin is configured as a general-purpose output pin. Reads return the value of the SPIx_SIMO pin.
		Write 0	No effect.
		Write 1	SPIPC3.SIMODOUT is cleared to 0.
9	CLKCLR		SPIx_CLK data out clear. This bit is only active when the SPIx_CLK pin is configured as a general-purpose output pin. Reads return the value of the SPIx_CLK pin.
		Write 0	No effect.
		Write 1	SPIPC3.CLKDOUT is cleared to 0.
8	ENACLR		$\overline{\text{SPIx_ENA}}$ data out clear. This bit is only active when the $\overline{\text{SPIx_ENA}}$ pin is configured as a general-purpose output pin. Reads return the value of the $\overline{\text{SPIx_ENA}}$ pin.
		Write 0	No effect.
		Write 1	SPIPC3.ENADOUT is cleared to 0.
7-0	SCS0CLR[*n*]		$\overline{\text{SPIx_SCS[n]}}$ data out clear. This bit is only active when the $\overline{\text{SPIx_SCS[n]}}$ pin is configured as a general-purpose output pin. Reads return the value of the $\overline{\text{SPIx_SCS[n]}}$ pin. Not all devices support multiple slave chip select ($\overline{\text{SPIx_SCS[n]}}$) pins, see your device-specific data manual for supported pins. If the pins are not available, the corresponding bit is reserved and should be cleared to 0.
		Write 0	No effect.
		Write 1	SPIPC3.SCS0DOUT[*n*] is cleared to 0.

Copyright © 2013–2016, Texas Instruments Incorporated

27.3.12 SPI Transmit Data Register 0 (SPIDAT0)

The SPI transmit data register 0 (SPIDAT0) is shown in Figure 27-29 and described in Table 27-20.

Figure 27-29. SPI Data Register 0 (SPIDAT0)



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

Table 27-20. SPI Data Register 0 (SPIDAT0) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return zero and writes have no effect.
15-0	TXDATA	0-FFFFh	SPI transmit data. When written, these bits will be copied to the shift register if it is empty. If the shift register is not empty, the TXBUF will hold the written values. SPIGCR1.ENABLE must be set to 1 before this register can be written to. Writing a 0 to the SPIGCR1.ENABLE forces the TXDATA field to 0. **Note:** Irrespective of the character length, the transmit data should be right-justified before writing to SPIDAT0 register. **Note:** The default data format control register for SPIDAT0 is SPIFMT0. However, it is possible to reprogram the DFSEL field of SPIDAT1 before using SPIDAT0, to select a different SPIFMT*n* register.

Copyright © 2013–2016, Texas Instruments Incorporated

27.3.13 SPI Transmit Data Register 1 (SPIDAT1)

The SPI transmit data register (SPIDAT1) is shown in Figure 27-30 and described in Table 27-21.

Figure 27-30. SPI Data Register 1 (SPIDAT1)

31	29	28	27	26	25	24
Reserved		CSHOLD	Reserved	WDEL	DFSEL	
R-0		R/W-0	R-0	R/W-0	R/W-0	

23	16
CSNR[*n*](1)	
R/W-0	

15	0
TXDATA	
R/W-0	

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

(1) Not all devices support multiple slave chip select ($\overline{\text{SPIx_SCS[n]}}$) pins, see your device-specific data manual for supported pins.

Table 27-21. SPI Data Register 1 (SPIDAT1) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Reads return zero and writes have no effect.
28	CSHOLD		Chip select hold mode. The CSHOLD bit is supported in master mode only. In slave mode, this bit is ignored. CSHOLD defines the behavior of the chip select line at the end of a data transfer.
		0	The chip select signal is deactivated at the end of a transfer after the T2CDELAY time has passed.
		1	The chip select signal is held active at the end of a transfer until a control field with new data and control information is loaded into SPIDAT1. If the new chip select hold information equals the previous one, the active chip select signal is extended until the end of transfer with CSHOLD cleared.
27	Reserved	0	Reads return zero and writes have no effect.
26	WDEL		Enable the delay counter at the end of the current transaction. The WDEL bit is supported in master mode only. In slave mode, this bit is ignored.
		0	No delay will be inserted. However, $\overline{\text{SPIx_SCS[n]}}$ pin will still be deactivated for at least 2 SPI module clock cycles if CSHOLD = 0.
		1	After a transaction, SPIFMT*n*.WDELAY of the selected data format will be loaded into the delay counter. No transaction will be performed until the SPIFMT*n*.WDELAY counter overflows. The $\overline{\text{SPIx_SCS[n]}}$ pin will be deactivated for at least (WDELAY + 2) × SPI module clock period.
25-24	DFSEL	0-3h	Data word format select
		0	Data word format 0 is selected
		1h	Data word format 1 is selected
		2h	Data word format 2 is selected
		3h	Data word format 3 is selected
			Note: Preselecting a Format Register. Writing to just the control field (using byte writes) does not initiate any SPI transfer in master mode. This feature can be used to set up SPIx_CLK phase or polarity before actually starting the transfer by just updating the DFSEL fields in the control field to select the required phase/polarity combination.
23-16	CSNR[*n*]		Chip select number. The CSNR field defines the state of the $\overline{\text{SPIx_SCS[n]}}$ pins during a master data transfer. The value of the CSNR field is driven directly on the $\overline{\text{SPIx_SCS[n]}}$ pins. Each bit in the CSNR field corresponds to an $\overline{\text{SPIx_SCS[n]}}$ pin, for example, CSNR[0] corresponds to $\overline{\text{SPIx_SCS[0]}}$ (see your device-specific data manual to determine how many SPI pins are available on your device).
			The state of the chip select pins when no transmissions are active is specified through the CSDEF field in the SPI default chip select register (SPIDEF). The chip select pins can remain in their active state by setting the CSHOLD bit to 1. When the SPI is configured in slave mode, this field must be written as 00h.
		0	$\overline{\text{SPIx_SCS[n]}}$ pin is driven low.
		1	$\overline{\text{SPIx_SCS[n]}}$ pin is driven high.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 27-21. SPI Data Register 1 (SPIDAT1) Field Descriptions (continued)

Bit	Field	Value	Description
15-0	TXDATA	0-FFFFh	Transfer data. When written, these bits will be copied to the shift register if it is empty. If the shift register is not empty, the TXBUF will hold the written values. SPIGCR1.ENABLE must be set to 1 before this register can be written to. Writing a 0 to the SPIGCR1.ENABLE forces the lower 16 bits of the SPIDAT1 to 0. **Note:** Irrespective of the character length, the transmit data should be right-justified before writing to SPIDAT1.

27.3.14 SPI Receive Buffer Register (SPIBUF)

The SPI receive buffer register (SPIBUF) is shown in Figure 27-31 and described in Table 27-22.

Figure 27-31. SPI Buffer Register (SPIBUF)

31	30	29	28	27	26	25	24
RXEMPTY	RXOVR	TXFULL	BITERR	DESYNC	PARERR	TIMEOUT	DLENERR
RS-1	RC-0	R-0	RC-0	RC-0	RC-0	RC-0	RC-0

23	16
Reserved	
R-0	

15	0
RXDATA	
R-0	

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

Table 27-22. SPI Buffer Register (SPIBUF) Field Descriptions

Bit	Field	Value	Description
31	RXEMPTY		Receive data buffer empty. When host reads the RXDATA field or the entire SPIBUF register this automatically sets the RXEMPTY flag. When a data transfer is completed, the received data is copied into SPIBUF, the RXEMPTY flag is cleared. This flag gets set to 1 under following conditions: • Reading the RXDATA field of the SPIBUF register. • Writing 1 to clear the RXINTFLG bit in the SPIFLG register.
		0	New data has been received and copied into the SPIBUF register.
		1	No data received since last reading of the SPIBUF register.
			Write-Clearing the SPIFLG.RXINTFLG bit before reading the SPIBUF register indicates the received data is being ignored. Conversely, SPIFLG.RXINTFLG can be cleared by reading the RXDATA field of the SPIBUF register or the entire SPIBUF register.
30	RXOVR		Receive data buffer overrun. When a data transfer is completed and the received data is copied into the RXBUF while it is already full, RXOVR is set. An overrun always occurs to the RXBUF, and SPIBUF contents never get overwritten until after it is read by the CPU/DMA. Reading SPIBUF register does not clear the RXOVR bit. If an overrun interrupt is detected, then the SPIBUF may need to be read twice to get to the overrun buffer. This is due to the fact that the overrun will always occur to the internal RXBUF. Each read to the SPIBUF will result in RXBUF contents (if it is full) getting copied to SPIBUF. **Note:** A special condition under which RXOVR flag gets set. If both SPIBUF and RXBUF are already full and while another buffer receive is underway, if any errors like TIMEOUT, BITERR and DLENERR occur, then RXOVR will be set to indicate that the status flags are getting overwritten by the new transfer. This overrun should be treated like a normal receiver overrun.
		0	No receive data overrun condition occurred since last time reading the data field.
		1	A receive data overrun condition occurred since last time reading the data field.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 27-22. SPI Buffer Register (SPIBUF) Field Descriptions (continued)

Bit	Field	Value	Description
29	TXFULL		Transmit data buffer full. This flag is a read-only flag. Writing into SPIDAT0 or SPIDAT1 field while the TX shift register is full will automatically set the TXFULL flag. Once the data is copied to the shift register, the TXFULL flag will be cleared. Writing to the SPIDAT0/SPIDAT1 register when both TXBUF and the TX shift register are empty does not set the TXFULL flag.
		0	The transmit buffer is empty; SPIDAT0/SPIDAT1 is ready to accept a new data.
		1	The transmit buffer is full; SPIDAT0/SPIDAT1 is not ready to accept new data.
28	BITERR		Bit error. There was a mismatch of internal transmit data and transmitted data. The SPI samples the signal of the transmit pin (master: SIMO, slave: SOMI) at the receive point (half clock cycle after transmit point). If the sampled value differs from the transmitted value, a bit error is detected and the flag BITERR is set. A possible reason for a bit error can be noise, a too-high bit rate/capacitive load, or another master/slave trying to transmit at the same time.
			Note: This flag is cleared to 0 when RXDATA portion of the SPIBUF register is read.
		0	No bit error occurred.
		1	A bit error occurred.
27	DESYNC		Desynchronization of slave device. This bit is active in master mode only. The master monitors the $\overline{\text{SPIx_ENA}}$ signal coming from the slave device and sets the DESYNC flag if $\overline{\text{SPIx_ENA}}$ is deactivated before the last reception point or after the last bit is transmitted plus $t_{T2EDELAY}$. If DESYNCENA is set, an interrupt is asserted. Desynchronization can occur if a slave device misses a clock edge coming from the master.
			Note: Possible inconsistency of DESYNC flag in SPI. Because of the nature of this error, under some circumstances it is possible for a desync error detected for the previous buffer to be visible in the current buffer. This is because the receive completion flag/interrupt will be generated when the buffer transfer is completed. But desync will be detected after the buffer transfer is completed. So, if CPU/DMA reads the received data quickly when an RXINT is detected, then the status flag may not reflect the correct desync condition.
			Note: This flag is cleared to 0 when the RXDATA portion of the SPIBUF register is read.
		0	No slave de-synchronization detected.
		1	A slave device is desynchronized.
26	PARERR		Parity error. The calculated parity differs from received parity bit. If the parity generator is enabled an even or odd parity bit is added at the end of a data word. During reception of the data word ,the parity generator calculates the reference parity and compares it to the received parity bit. If a mismatch is detected, the PARERR flag is set.
			Note: This flag is cleared to 0 when the RXDATA portion of the SPIBUF register is read.
		0	No parity error detected.
		1	A parity error occurred.
25	TIMEOUT		Time-out because of non-activation of $\overline{\text{SPIx_ENA}}$ pin. This bit is valid in master mode only. The SPI generates a time-out because the slave hasn't responded in time by activating the $\overline{\text{SPIx_ENA}}$ signal after the chip select signal has been activated. If a time-out condition is detected, the corresponding chip select is deactivated immediately and the TIMEOUT flag is set.
			Note: This flag is cleared to 0 when RXDATA portion of the SPIBUF register is read.
		0	No $\overline{\text{SPIx_ENA}}$ pin time-out occurred.
		1	An $\overline{\text{SPIx_ENA}}$ signal time-out occurred.
24	DLENERR		Data length error flag.
			Note: This flag is cleared to 0 when the RXDATA portion of the SPIBUF register is read.
		0	No data length error has occurred.
		1	A data length error has occurred.
23-16	Reserved	0	Reads return zero and writes have no effect.
15-0	RXDATA	0-FFFFh	SPI receive data. This is the received data, transferred from the receive shift-register at the end of a transfer completion. Irrespective of the programmed character length and the direction of shifting, the received data is stored right-justified in the register.

Copyright © 2013–2016, Texas Instruments Incorporated

27.3.15 SPI Emulation Register (SPIEMU)

The SPI emulation register (SPIEMU) is shown in Figure 27-32 and described in Table 27-23.

Figure 27-32. SPI Emulation Register (SPIEMU)

31	16
Reserved	
R-0	

15	0
RXDATA	
R-0	

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

Table 27-23. SPI Emulation Register (SPIEMU) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return zero and writes have no effect.
15-0	RXDATA	0-FFFFh	SPI receive data. SPI emulation is a mirror of the SPIBUF register. The only difference between SPIEMU and SPIBUF is that a read from SPIEMU does not clear any of the status flags.

Copyright © 2013–2016, Texas Instruments Incorporated

27.3.16 SPI Delay Register (SPIDELAY)

The SPI delay register (SPIDELAY) is shown in Figure 27-33 and described in Table 27-24.

Figure 27-33. SPI Delay Register (SPIDELAY)

31	24	23	16
C2TDELAY		T2CDELAY	
R/W-0		R/W-0	

15	8	7	0
T2EDELAY		C2EDELAY	
R/W-0		R/W-0	

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

Table 27-24. SPI Delay Register (SPIDELAY) Field Descriptions

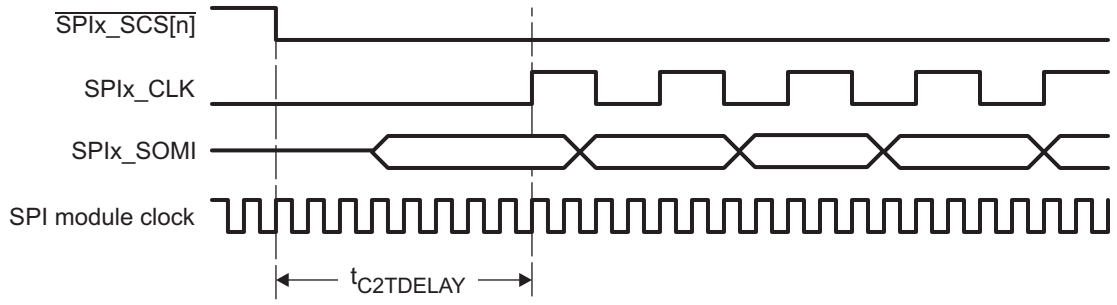
Bit	Field	Value	Description
31-24	C2TDELAY	0-FFh	Chip-select-active-to-transmit-start-delay. C2TDELAY is used in master mode only. It defines a setup time for the slave device that delays the data transmission from the chip select active edge by a multiple of SPI module clock cycles. C2TDELAY can be configured between 3 and 257 SPI module clock cycles. See Figure 27-34. The setup time value is calculated as follows: $t_{C2TDELAY}$ = (C2TDELAY + 2) × SPI module clock period **Note:** If C2TDELAY = 0, then $t_{C2TDELAY}$ = 0. Example: SPI module clock = 25 MHz -> SPI module clock period = 40 ns; C2TDELAY = 06h; > $t_{C2TDELAY}$ = 320 ns; When the chip select signal becomes active, the slave has to prepare for data transfer within 320 ns. **Note:** If phase = 1, the delay between $\overline{\text{SPIx_SCS[n]}}$ falling edge to the first edge of SPIx_CLK will have an additional 0.5 SPIx_CLK period delay. This delay is as per the SPI protocol.
23-16	T2CDELAY	0-FFh	Transmit-end-to-chip-select-inactive-delay. T2CDELAY is used in master mode only. It defines a hold time for the slave device that delays the chip select deactivation by a multiple of SPI module clock cycles after the last bit is transferred. T2CDELAY can be configured between 2 and 256 SPI module clock cycles. See Figure 27-35. The hold time value is calculated as follows: $t_{T2CDELAY}$ = (T2CDELAY + 1) × SPI module clock period **Note:** If T2CDELAY = 0, then $t_{T2CDELAY}$ = 0 Example: VBUSPCLK = 25 MHz -> VBUSPCLK period = 40 ns; T2CDELAY = 03h; > $t_{T2CDELAY}$ = 160 ns; After the last data bit (or parity bit) is being transferred the chip select signal is held active for 160 ns. **Note:** If phase = 0, then between the last edge of SPIx_CLK and rise-edge of $\overline{\text{SPIx_SCS[n]}}$ there will be an additional delay of 0.5 SPIx_CLK period. This is as per the SPI protocol. Both C2TDELAY and T2CDELAY counters will not have any dependancy on the $\overline{\text{SPIx_ENA}}$ pin value. Even if the $\overline{\text{SPIx_ENA}}$ pin is asserted by the slave, the master will continue to delay the start of SPIx_CLK until the C2TDELAY counter overflows. Similarly, even if the $\overline{\text{SPIx_ENA}}$ pin is deasserted by the slave, the master will continue to hold the $\overline{\text{SPIx_SCS[n]}}$ pins active until the T2CDELAY counter overflows. This way, it is assured that the setup/hold times of the $\overline{\text{SPIx_SCS[n]}}$ pins are determined by the delay timers alone. To achieve better throughput, it should be ensured that these two timers are kept at the minimum possible values.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 27-24. SPI Delay Register (SPIDELAY) Field Descriptions (continued)

Bit	Field	Value	Description
15-8	T2EDELAY	0-FFh	Transmit-data-finished-to-$\overline{\text{SPIx_ENA}}$-pin-inactive-time-out. T2EDELAY is used in master mode only. It defines a time-out value as a multiple of SPI clock before the $\overline{\text{SPIx_ENA}}$ signal has to become inactive and after the CS becomes inactive. The SPI clock depends on which data format is selected. If the slave device is missing one or more clock edges, it is becoming desynchronized. Although the master has finished the data transfer the slave is still waiting for the missed clock pulses and the $\overline{\text{SPIx_ENA}}$ signal is not disabled. The T2EDELAY defines a time-out value that triggers the DESYNC flag, if the $\overline{\text{SPIx_ENA}}$ signal is not deactivated in time. The DESYNC flag is set to indicate that the slave device did not deassert its $\overline{\text{SPIx_ENA}}$ pin in time to acknowledge that it has received all the bits of the sent character. The DESYNC flag is also set if the SPI detects a deassertion of the $\overline{\text{SPIx_ENA}}$ pin even before the end of the transmission. See Figure 27-36. The time-out value is calculated as follows: $t_{T2EDELAY}$ = T2EDELAY/SPIclock Example: SPIclock = 8 Mbit/s; T2EDELAY = 10h; > $t_{T2EDELAY}$ = 2 μs; The slave device has to disable the $\overline{\text{SPIx_ENA}}$ signal within 2 μs; otherwise, the DESYNC flag in SPIFLG is set and an interrupt is asserted if enabled.
7-0	C2EDELAY	0-FFh	Chip-select-active-to-$\overline{\text{SPIx_ENA}}$-signal-active-time-out. C2EDELAY is used only in master mode and it applies only if the addressed slave generates an $\overline{\text{SPIx_ENA}}$ signal as a hardware handshake response. C2EDELAY defines the maximum time between the SPI activates the chip select signal and the addressed slave has to respond by activating the $\overline{\text{SPIx_ENA}}$ signal. C2EDELAY defines a time-out value as a multiple of SPI clocks. See Figure 27-37. **Note:** If the slave device is not responding with the $\overline{\text{SPIx_ENA}}$ signal before the time-out value is reached, the TIMEOUT flag in SPIFLG is set and an interrupt is asserted if enabled. The timeout value is calculated as follows: $t_{C2EDELAY}$ = C2EDELAY/SPIclock Example: SPIclock = 8 Mbit/s; C2EDELAY = 30h; > $t_{C2EDELAY}$ = 6 μs; The slave device has to activate the $\overline{\text{SPIx_ENA}}$ signal within 6 μs after the SPI has activated the chip select signal ($\overline{\text{SPIx_SCS[n]}}$); otherwise, the TIMEOUT flag in SPIFLG is set and an interrupt is asserted if enabled.

Figure 27-34. Example: $t_{C2TDELAY}$ = 8 SPI Module Clock Cycles



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 27-35. Example: $t_{T2CDELAY}$ = 4 SPI Module Clock Cycles

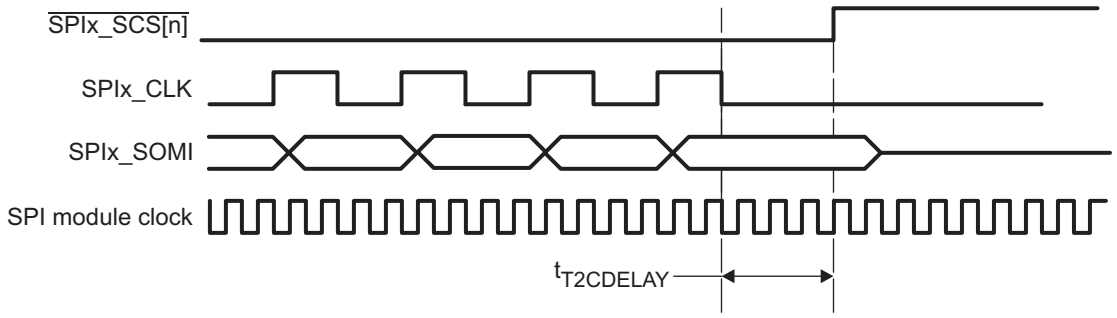


Figure 27-36. Transmit-Data-Finished-to-SPIx_ENA-Inactive-Timeout

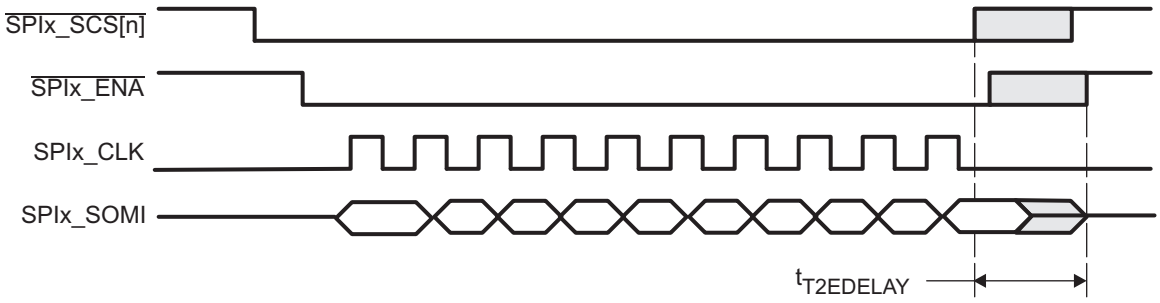
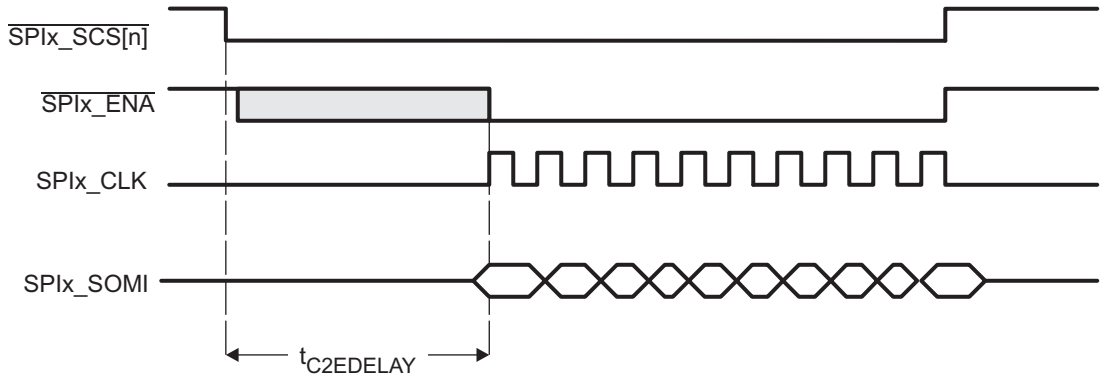


Figure 27-37. Chip-Select-Active-to-SPIx_ENA-Signal-Active-Timeout



Copyright © 2013–2016, Texas Instruments Incorporated

27.3.17 SPI Default Chip Select Register (SPIDEF)

The SPI default chip select register (SPIDEF) is shown in Figure 27-38 and described in Table 27-25.

Figure 27-38. SPI Default Chip Select Register (SPIDEF)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		CSDEF[*n*](1)	
R-0		R/W-FFh	

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

(1) Not all devices support multiple slave chip select ($\overline{\text{SPIx_SCS[n]}}$) pins, see your device-specific data manual for supported pins. If the pins are not available, the corresponding bit is reserved and should be cleared to 0.

Table 27-25. SPI Default Chip Select Register (SPIDEF) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return zero and writes have no effect.
7-0	CSDEF[*n*]		Chip select default pattern. The CSDEF field defines the state of the the $\overline{\text{SPIx_SCS[n]}}$ pins when no transmissions are performed. The value of the CSDEF field is driven directly on the $\overline{\text{SPIx_SCS[n]}}$ pins. Each bit in the CSDEF field corresponds to an $\overline{\text{SPIx_SCS[n]}}$ pin, for example, CSDEF[0] corresponds to $\overline{\text{SPIx_SCS[0]}}$ (see your device-specific data manual to determine how many SPI pins are available on your device). The state of the chip select pins during a transmission is specified through the CSNR field in the SPI transmit data register (SPIDAT1). The chip select pins can remain in their active state by setting the CSHOLD bit in SPIDAT1 to 1. In slave mode, the CSDEF field should be set to FFh.
		0	$\overline{\text{SPIx_SCS[n]}}$ pin is driven low.
		1	$\overline{\text{SPIx_SCS[n]}}$ pin is driven high.

Copyright © 2013–2016, Texas Instruments Incorporated

27.3.18 SPI Data Format Registers (SPIFMTn)

The SPI data format registers (SPIFMT0, SPIFMT1, SPIFMT2, and SPIFMT3) are shown in Figure 27-39 and described in Table 27-26.

Figure 27-39. SPI Data Format Register (SPIFMTn)

31	30	29					24
Reserved		WDELAY					
R-0		R/W-0					
23	22	21	20	19	18	17	16
PARPOL	PARENA	WAITENA	SHIFTDIR	Reserved	DISCSTIMERS	POLARITY	PHASE
R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0
15							8
PRESCALE							
R/W-0							
7		5	4				0
Reserved			CHARLEN				
R-0			R/W-0				

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

Table 27-26. SPI Data Format Register (SPIFMTn) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Reads return zero and writes have no effect.
29-24	WDELAY	0-3Fh	Delay in between transmissions. Idle time that will be applied at the end of the current transmission if the bit WDEL is set in the current buffer. The delay to be applied is equal to: WDELAY × $P_{SPI\ module\ clock}$ + 2 × $P_{SPI\ module\ clock}$ $P_{SPI\ module\ clock}$ -> Period of SPI module clock
23	PARPOL		Parity polarity: even or odd. PARPOL can be modified in privilege mode only.
		0	An even parity flag is added at the end of the transmit data stream.
		1	An odd parity flag is added at the end of the transmit data stream.
22	PARENA		Parity enable.
		0	No parity generation/ verification is performed.
		1	A parity is transmitted at the end of each transmit data stream. At the end of a transfer the parity generator compares the received parity bit with the locally calculated parity flag. If the parity bits do not match the PARERR flag is set in the corresponding control field. The parity type (even or odd) can be selected via the PARPOL bit.
21	WAITENA		The master waits for $\overline{\text{SPIx_ENA}}$ signal from slave. WAITENA is considered in master mode only. In slave mode this bit has no meaning. WAITENA enables a flexible SPI network where slaves with $\overline{\text{SPIx_ENA}}$ signal and slaves without $\overline{\text{SPIx_ENA}}$ signal can be mixed.
		0	The SPI does not wait for the $\overline{\text{SPIx_ENA}}$ signal from the slave and directly starts the transfer.
		1	Before the SPI starts the data transfer it waits for the $\overline{\text{SPIx_ENA}}$ signal to become low. If the $\overline{\text{SPIx_ENA}}$ signal is not pulled down by the addressed slave before the internal time-out counter (C2EDELAY) overflows, then the master aborts the transfer and sets the TIMEOUT error flag.
20	SHIFTDIR		Shift direction.
		0	Most significant bit is shifted out first.
		1	Least significant bit is shifted out first.
19	Reserved	0	Reads return zero and writes have no effect.
18	DISCSTIMERS		Disable chip select timers for this format register. The C2TDELAY and T2CDELAY timers are by default enabled for all the data format registers. Using this bit, these timers can be disabled for a particular data format if not required. When a master is handling multiple slaves, with varied set-up hold requirement, the application can selectively choose to include or not include the chip select delay timers for any slaves.
		0	Both C2TDELAY and T2CDELAY counts are inserted for the chip selects.
		1	No C2TDELAY or T2CDELAY is inserted in the chip select timings.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 27-26. SPI Data Format Register (SPIFMT*n*) Field Descriptions (continued)

Bit	Field	Value	Description
17	POLARITY		SPI clock polarity.
		0	SPI clock signal is low-inactive (before and after data transfer the clock signal is low).
		1	SPI clock signal is high-inactive (before and after data transfer the clock signal is high).
16	PHASE		SPI clock delay.
		0	SPI clock signal is not delayed versus the transmit/receive data stream. The first data bit is transmitted with the first clock edge and the first bit is received with the second (inverse) clock edge.
		1	SPI clock signal is delayed by a half SPI clock cycle versus the transmit/receive data stream. The first transmit bit has to output prior to the first clock edge. The master and slave receive the first bit with the first edge.
15-8	PRESCALE	2h-FFh	SPI prescaler. It determines the bit transfer rate if the SPI is the network master and is directly derived from the SPI module clock. If the SPI is configured as slave, PRESCALE needs to be configured to a valid value, but PRESCALE is ignored. The clock rate can be calculated as: SPI clock frequency = SPI module clock/(PRESCALE + 1) **Note:** PRESCALE values less than 2h are not supported.
7-5	Reserved	0	Reads return zero and writes have no effect.
4-0	CHARLEN	0-1Fh	SPI data word length. Legal values are 2h (data word length = 2 bit) to 10h (data word length = 16). Illegal values, such as 0 or 1Fh are not detected and their effect is indeterminate.

Copyright © 2013–2016, Texas Instruments Incorporated

27.3.19 SPI Interrupt Vector Register 1 (INTVEC1)

The SPI interrupt vector register 1 (INTVEC1) is shown in Figure 27-40 and described in Table 27-27.

Figure 27-40. SPI Interrupt Vector Register 1 (INTVEC1)

31	16
Reserved	
R-0	

15	6	5	1	0
Reserved		INTVECT1		Rsvd
R-0		R-0		R-0

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

Table 27-27. SPI Interrupt Vector Register 1 (INTVEC1) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reads return zero and writes have no effect.
5-1	INTVECT1	0-1Fh	Interrupt vector for interrupt line INT1. INTVECT1 returns the vector of the pending interrupt at interrupt line INT1. If more than one interrupt is pending, INTVECT1 always references the highest priority interrupt source first. The interrupts available for SPI in the descending order of their priorities are as given below. • Transmission error Interrupt • Receive buffer overrun interrupt • Receive buffer full interrupt • Transmit buffer empty interrupt The INTVECT1 field just reflects the status of SPIFLG in a vectorized format. So, any updates to SPIFLG will automatically reflect in the vector value in this register. Vectors for each of these interrupts will be reflected on the INTVECT1 bits, when they occur. Reading the vectors for the receive buffer overrun and receive buffer full interrupts will automatically clear the respective flags in the SPIFLG. Reading the vector register when transmitter empty is indicated does not clear the TXINTFLG in SPIFLG. Writing a new data to SPIDAT0/SPIDAT1 clears the transmitter empty interrupt. On reading the INTVECT1 bits, the vector of the next highest priority interrupt (if any) will be then reflected on the INTVECT1 bits. If two or more interrupts occur simultaneously, the vector for the highest priority interrupt will be reflected on the INTVECT1 bits. The following are the SPI interrupt vectors for line INT1:
		0	No interrupt pending
		1h-10h	Reserved
		11h	Error interrupt pending. Refer to lower halfword of SPIINT0 to determine more details about the type of error.
		12h	The pending interrupt is receive buffer full interrupt.
		13h	The pending interrupt is receive buffer overrun interrupt.
		14h	The pending interrupt is transmit buffer empty interrupt.
		15h-1Fh	Reserved
0	Reserved	0	Reads return zero and writes have no effect.

Copyright © 2013–2016, Texas Instruments Incorporated

64-Bit Timer Plus

This chapter describes the operation of the software-programmable 64-bit Timer Plus. See your device-specific data manual to determine how many Timer modules are available on your device.

Topic **Page**

28.1 Introduction **1261**
28.2 Registers **1279**

Copyright © 2013–2016, Texas Instruments Incorporated

28.1 Introduction

The 64-bit Timer Plus can be programmed in 64-bit mode, dual 32-bit unchained mode, or dual 32-bit chained mode. Some Timer Plus implementations have signal connections to internal device reset that can be used in watchdog timer mode. New features over previous timers include: external clock/event input, period reload, external event capture, and timer counter register read reset.

28.1.1 Purpose of the Peripheral

The timer can support four basic modes of operation: a 64-bit general-purpose (GP) timer, dual unchained 32-bit GP timers, dual chained 32-bit timers, or a watchdog timer. The GP timer modes can be used to generate periodic interrupts and DMA synchronization events. The watchdog timer mode is used to provide a recovery mechanism for the device in the event of a fault condition (such as a non-exiting code loop).

28.1.2 Features

The 64-bit timer consists of the following features -- some features may not be supported on all timer instantiations (see your device-specific data manual for supported features):

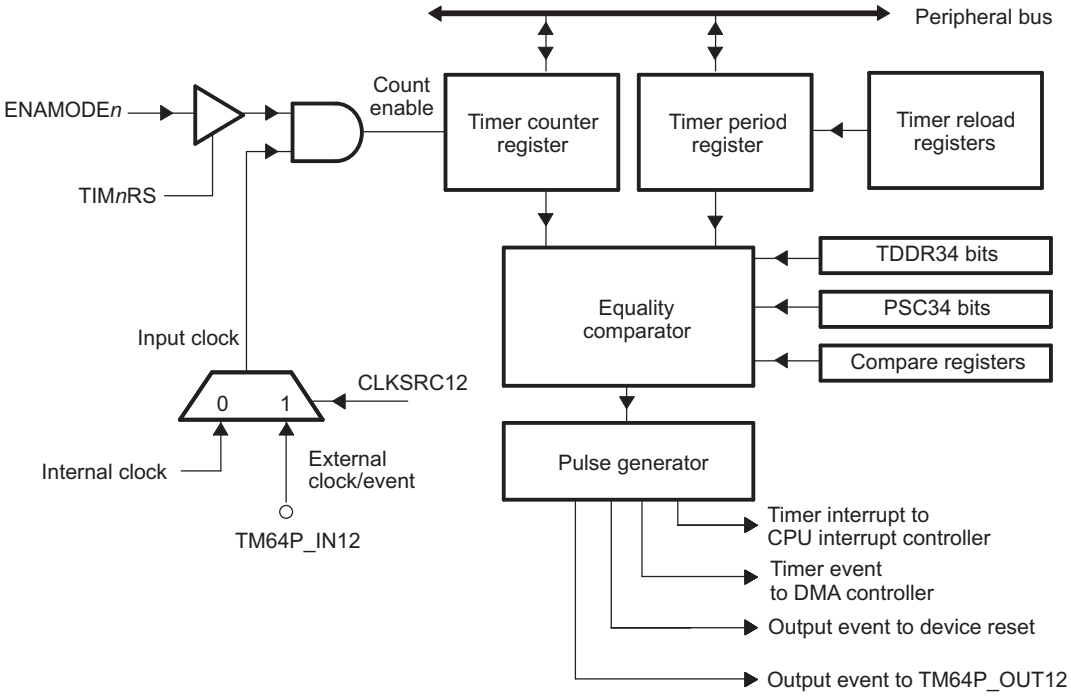
- 64-bit count-up counter
- Timer modes:
  - 64-bit general-purpose timer mode
  - Dual 32-bit unchained general-purpose timer mode
  - Dual 32-bit chained timer mode
  - Watchdog timer mode
- 2 possible clock sources:
  - Internal clock
  - External clock/event input via timer input pins
- 3 possible operation modes:
  - One-time operation (timer runs for one period then stops)
  - Continuous operation (timer automatically resets to zero after each period and continues to operate)
  - Continuous operation with period reload (timer automatically assumes the value of the reload registers after each period and continues to operate)
- Generates interrupts to CPU
- Generates sync events to DMA
- Generates output event to device reset (watchdog only)
- Generates output event to timer output pins (if pins are available)
- External event capture via timer input pins (if pins are available)

Copyright © 2013–2016, Texas Instruments Incorporated

28.1.3 Block Diagram

A block diagram of the timer is shown in Figure 28-1. Detailed information about the architecture and operation of the timers is in Section 28.1.5 and Section 28.1.6.

Figure 28-1. Timer Block Diagram



28.1.4 Industry Standard Compatibility Statement

This peripheral is not intended to conform to any specific industry standard.

Architecture

28.1.5 Architecture – General-Purpose Timer Mode

This section describes the timer in the general-purpose (GP) timer mode.

28.1.5.1 Backward Compatible Mode

The Timer Plus supports the following additional features over the other timers:

- External clock/event input
- Period reload
- External event capture mode
- Timer counter register read reset mode
- Timer counter capture registers
- Register for interrupt/DMA generation control and status

By default, period reload, external event capture mode, timer counter register read reset mode, timer counter capture registers, and interrupt/DMA/TM64P_OUT generation control and status are not available. To enable these features, you must set the PLUSEN bit in the timer global control register (TGCR). These features are described throughout the following sections. External clock/event input is always available, regardless of the state of the backward compatible bit.

Copyright © 2013–2016, Texas Instruments Incorporated

28.1.5.2 Clock Control

The timer can use an internal or external clock source for the counter period. The following sections explain how to select the clock source.

As shown in Table 28-1 and Figure 28-2, the timer clock source is selected using the clock source (CLKSRC12) bit in the timer control register (TCR). Two clock sources are available to drive the timer clock:

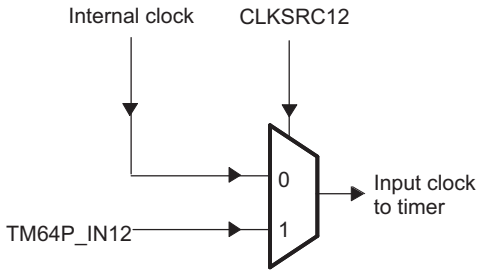
- internal clock by setting CLKSRC12 = 0.
- external clock on input pin TM64P_IN12 by setting CLKSRC12 = 1.

At reset, the clock source is the internal clock. Details on each of the clock source configuration options are included in the following sections.

Table 28-1. Timer Clock Source Selection

CLKSRC12	Input Clock
0	Internal clock (default)
1	External clock on timer input

Figure 28-2. Timer Clock Source Block Diagram



28.1.5.2.1 Using the Internal Clock Source to the Timer

The internal clock source to the timer is generated by the PLL controller. This clock source determines the speed of the timer since the timer counts up in units of source clock cycles. When determining the period and prescaler settings for the timer, choose the desired period in units of source clock cycles. For details on the generation of the on-chip clocks, see the *Phase-Locked Loop Controller (PLLC)* chapter.

The CLKSRC12 parameter in the timer control register (TCR) determines whether an internal or external clock is used as the clock source for the timer. If the timer is configured in 64-bit mode or 32-bit chained mode, CLKSRC12 controls the clock source for the entire timer. If the timer is configured in dual 32-bit unchained mode (TIMMODE = 01 in TGCR), CLKSRC12 controls the timer 1:2 side of the timer only.

To select the internal clock as the clock source for the timer, CLKSRC12 in TCR must be cleared to 0.

28.1.5.2.2 Using the External Clock Source to the Timer

An external clock source can be provided to clock the timer through the timer input pin TM64P_IN12. The CLKSRC12 parameter in the timer control register (TCR) determines whether an internal or external clock is used as the clock source for the timer. If the timer is configured in 64-bit mode or 32-bit chained mode, CLKSRC12 controls the clock source for the entire timer. If the timer is configured in dual 32-bit unchained mode (TIMMODE = 01 in TGCR), CLKSRC12 controls the timer 1:2 side of the timer only.

At reset, the clock source defaults to the internal clock. Details on each of the clock source configuration options are included in the following sections. To select the external clock as the clock source for the timer, CLKSRC12 in TCR must be set to 1. The external clock source frequency must be no greater than the timer peripheral reference clock (see your device-specific data manual).

Copyright © 2013–2016, Texas Instruments Incorporated

28.1.5.3 Signal Descriptions

As shown in Figure 28-2, the TM64P_IN12 pin may be used as input to the timer. This signal can be used to drive the clock/event count or be used as an external event input for event capture mode. Pin TM64P_OUT12 may be used as an output from the timer to generate a clock or pulse signal.

28.1.5.4 Timer Modes

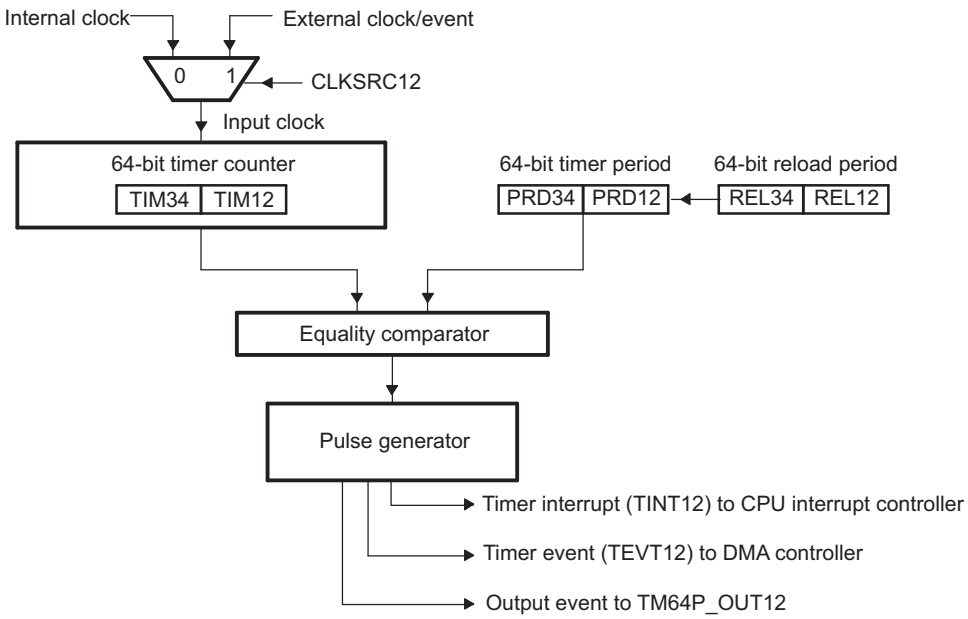
The following section describes the general-purpose (GP) timer modes.

28.1.5.4.1 64-Bit Timer Mode

The timer can be configured as a 64-bit timer by clearing the TIMMODE bit in the timer global control register (TGCR) to 0. At reset, 0 is the default setting for the TIMMODE bit.

In this mode, the timer operates as a single 64-bit up-counter (Figure 28-3). The counter registers (TIM12 and TIM34) form a 64-bit timer counter register and the period registers (PRD12 and PRD34) form a 64-bit timer period register. When the timer is enabled, the timer counter starts incrementing by 1 at every timer input clock cycle. When the timer counter matches the timer period, a maskable timer interrupt (TINT12) and a timer EDMA (TEVT12) are generated. When the timer is configured in continuous mode, the timer counter is reset to 0 on the cycle after the timer counter reaches the timer period. The timer can be stopped, restarted, reset, or disabled using control bits in TGCR.

Figure 28-3. 64-Bit Timer Mode Block Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

28.1.5.4.1.1 Enabling the 64-Bit Timer

The TIM12RS and TIM34RS bits in TGCR control whether the timer is in reset or capable of operating. For the timer to operate in 64-bit timer mode, the TIM12RS and TIM34RS bits must be set to 1.

The ENAMODE12 bit in the timer control register (TCR) controls whether the timer is disabled, enabled to run once, enabled to run continuously, or enabled to run continuously with period reload; the ENAMODE34 bit has no effect in 64-bit timer mode. When the timer is disabled (ENAMODE12 = 0), the timer does not run and maintains its current count value. When the timer is enabled for one time operation (ENAMODE12 = 1), it counts up until the counter value equals the period value and then stops. When the timer is enabled for continuous operation (ENAMODE12 = 2h), the counter counts up until it reaches the period value, then resets itself to zero and begins counting again. When the timer is enabled for continuous operation with period reload (ENAMODE12 = 3h), the counter counts up until it reaches the period value, then resets itself to zero, reloads the period registers (PRD12 and PRD34) with the value in the period reload registers (REL12 and REL34), and begins counting again.

Table 28-2 shows the bit values in TGCR to configure the 64-bit timer.

Table 28-2. 64-Bit Timer Configurations

64-Bit Timer Configuration	TGCR Bit		TCR Bit
	TIM12RS	TIM34RS	ENAMODE12
To place the 64-bit timer in reset	0	0	0
To disable the 64-bit timer (out of reset)	1h	1h	0
To enable the 64-bit timer for one-time operation	1h	1h	1h
To enable the 64-bit timer for continuous operation	1h	1h	2h
To enable the 64-bit timer for continuous operation with period reload	1h	1h	3h

Once the timer stops, if an external clock is used as the timer clock, the timer must remain disabled for at least one external clock period or the timer will not start counting again. When using the external clock, the count value is synchronized to the internal clock.

Note that when both the timer counter and timer period are cleared to 0, the timer can be enabled but the timer counter does not increment because the timer period is 0.

28.1.5.4.1.2 Reading the Counter Registers

When reading the timer count in 64-bit timer mode, the CPU must first read TIM12 followed by TIM34. When TIM12 is read, the timer copies TIM34 into a shadow register. When reading TIM34, the hardware logic returns the shadow register value. This ensures that the values read from the registers are not affected by the fact that the timer may continue to run as the registers are read. When reading the timers in 32-bit mode, TIM12 and TIM34 may be read in any order.

28.1.5.4.1.3 64-Bit Timer Configuration Procedure

To configure the GP timer to operate as a 64-bit timer, follow the steps below:

1. Select 64-bit mode (TIMMODE in TGCR).
2. Remove the timer from reset (TIM12RS and TIM34RS in TGCR).
3. Select the desired timer period (PRD12 and PRD34). Program with the desired timer period value - 1.
4. Enable the timer (ENAMODE12 in TCR).
5. If ENAMODE12 = 3h, write the desired timer period for the next timer cycle in the period reload registers (REL12 and REL34). Program with the desired timer period value - 1. This step can be done at any time before the current timer cycle ends.

Copyright © 2013–2016, Texas Instruments Incorporated

28.1.5.4.2 Dual 32-Bit Timer Modes

Each of the general-purpose timers can be configured as dual 32-bit timers by configuring the TIMMODE bit in the timer global control register (TGCR). In dual 32-bit timer mode, the two 32-bit timers can be operated independently (unchained mode) or in conjunction with each other (chained mode).

28.1.5.4.2.1 Chained Mode

The general-purpose timers can each be configured as a dual 32-bit chained timer by setting the TIMMODE bit to 3h in TGCR.

In the chained mode (Figure 28-4), one 32-bit timer (timer 3:4) is used as a 32-bit prescaler and the other 32-bit timer (timer 1:2) is used as a 32-bit timer. The 32-bit prescaler is used to clock the 32-bit timer. The 32-bit prescaler uses one counter register (TIM34) to form a 32-bit prescale counter register and one period register (PRD34) to form a 32-bit prescale period register.

When the timer is enabled, the prescale counter starts incrementing by 1 at every timer input clock cycle. One cycle after the prescale counter matches the prescale period, a clock signal is generated and the prescale counter register is reset to 0 (see the example in Figure 28-5).

The other 32-bit timer (timer 1:2) uses one counter register (TIM12) to form a 32-bit timer counter register and one period register (PRD12) to form a 32-bit timer period register. This timer is clocked by the output clock from the prescaler. The timer counter increments by 1 at every prescaler output clock cycle. When the timer counter matches the timer period, a maskable timer interrupt (TINT12) and a timer EDMA event (TEVT12) are generated. When the timer is configured in continuous mode, the timer counter is reset to 0 on the cycle after the timer counter reaches the timer period. The timer can be stopped, restarted, reset, or disabled using the TIM12RS and TIM34RS bits in TGCR. In the chained mode, the upper 16-bits of the timer control register (TCR) are not used.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 28-4. Dual 32-Bit Timers Chained Mode Block Diagram

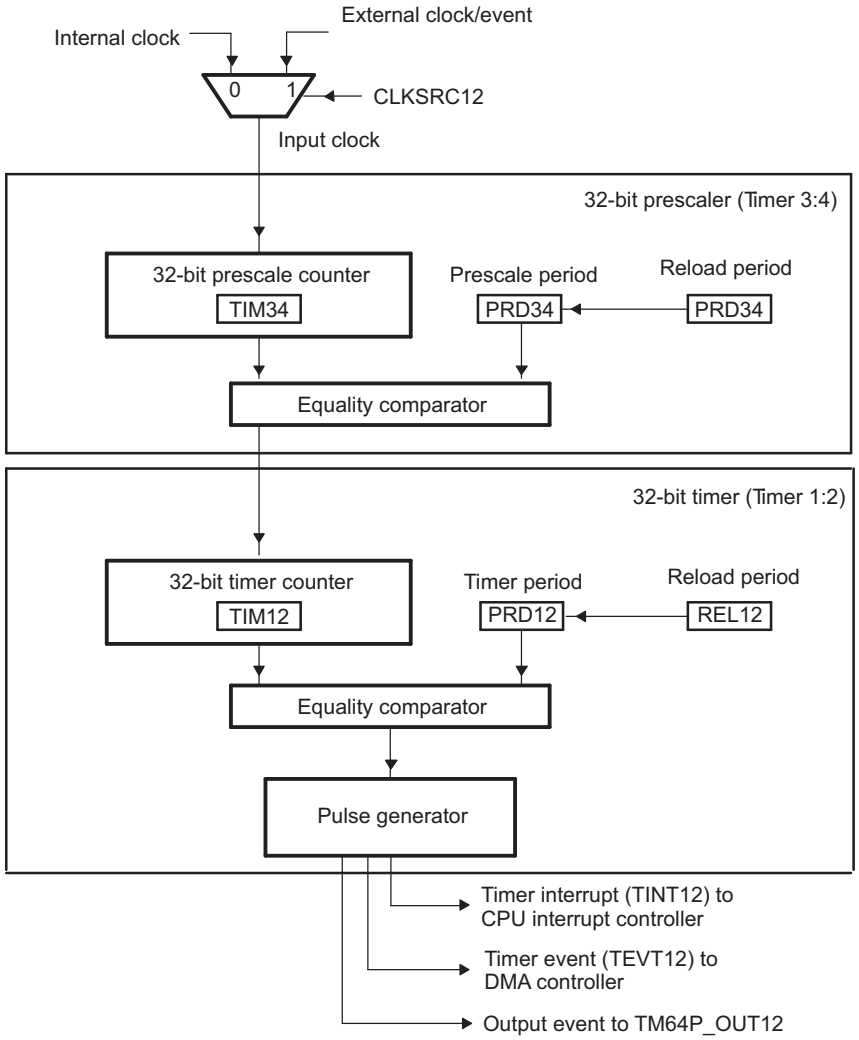
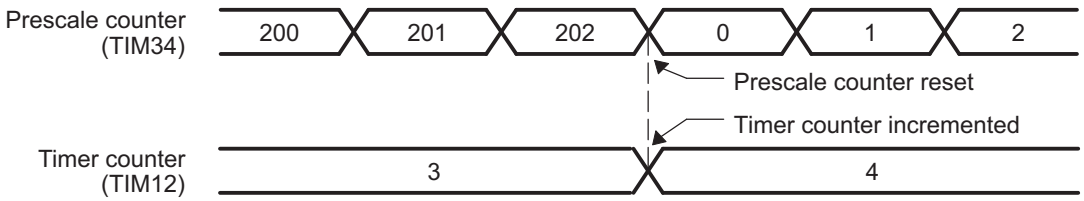


Figure 28-5. Dual 32-Bit Timers Chained Mode Example

32-bit prescaler settings: count = TIM34 = 200; period = PRD34 = 202
32-bit timer settings: count = TIM12 = 3; period = PRD12= 4



Copyright © 2013–2016, Texas Instruments Incorporated

28.1.5.4.2.1.1 Enabling the 32-Bit Timer Chained Mode

The TIM12RS and TIM34RS bits in TGCR determine whether the timer is in reset, or if it is capable of operating. The TIM12RS bit controls the reset of the timer 1:2 side of the timer and the TIM34RS bits control the reset of the timer 3:4 side of the timer. For the timer to operate, the TIM12RS and TIM34RS bits must be set to 1.

The ENAMODE12 bit in the timer control register (TCR) controls whether the timer is disabled, enabled to run once, enabled to run continuously, or enabled to run continuously with period reload; the ENAMODE34 bit has no effect in 32-bit timer chained mode. When the timer is disabled (ENAMODE12 = 0), the timer does not run and maintains its current count value. When the timer is enabled for one time operation (ENAMODE12 = 1), it counts up until the counter value equals the period value and then stops. When the timer is enabled for continuous operation (ENAMODE12 = 2h), the counter counts up until it reaches the period value, then resets itself to zero and begins counting again. When the timer is enabled for continuous operation with period reload (ENAMODE12 = 3h), the counter counts up until it reaches the period value, then resets itself to zero, reloads the period registers (PRD12 and PRD34) with the value in the period reload registers (REL12 and REL34), and begins counting again.

Table 28-3 shows the bit values in TGCR to configure the 32-bit timer in chained mode.

Table 28-3. 32-Bit Timer Chained Mode Configurations

	TGCR Bit		TCR Bit
32-Bit Timer Configuration	**TIM12RS**	**TIM34RS**	**ENAMODE12**
To place the 32-bit timer chained mode in reset	0	0	0
To disable the 32-bit timer chained mode (out of reset)	1h	1h	0
To enable the 32-bit timer chained mode for one-time operation	1h	1h	1h
To enable the 32-bit timer chained mode for continuous operation	1h	1h	2h
To enable the 32-bit timer chained mode for continuous operation with period reload (Timer 3 only)	1h	1h	3h

Once the timer stops, if an external clock is used as the timer clock, the timer must remain disabled for at least one external clock period or the timer will not start counting again. When using the external clock, the count value is synchronized to the internal clock.

Note that when both the timer counter and timer period are cleared to 0, the timer can be enabled but the timer counter does not increment because the timer period is 0.

28.1.5.4.2.1.2 32-Bit Timer Chained Mode Configuration Procedure

To configure the GP timer to operate as a dual 32-bit chained mode timer, follow the steps below:

1. Select 32-bit chained mode (TIMMODE in TGCR).
2. Remove the timer from reset (TIM12RS and TIM34RS in TGCR).
3. Select the desired timer period (PRD12). Program with the desired timer period value - 1.
4. Select the desired timer prescaler value (PRD34).
5. Enable the timer (ENAMODE12 in TCR).
6. If ENAMODE12 = 3h, write the desired timer period for the next timer cycle in the period reload registers (REL12 and REL34). Program with the desired timer period value - 1. This step can be done at any time before the current timer cycle ends.

Copyright © 2013–2016, Texas Instruments Incorporated

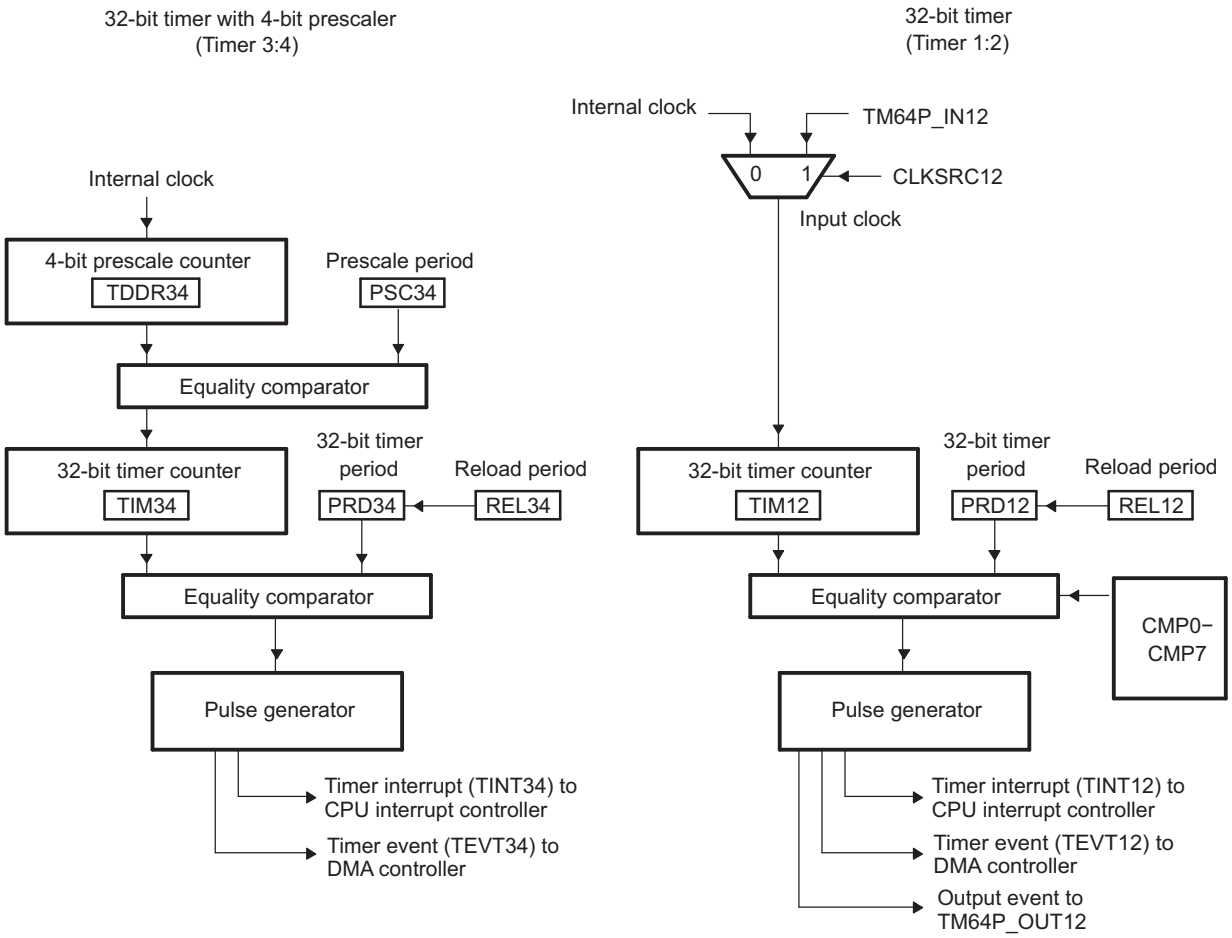
28.1.5.4.2.2 Unchained Mode

The general-purpose timers can be configured as a dual 32-bit unchained timers by setting the TIMMODE bit to 1 in TGCR.

In the unchained mode (Figure 28-6), the timer operates as two independent 32-bit timers. One 32-bit timer (timer 3:4) operates as a 32-bit timer being clocked by a 4-bit prescaler. The other 32-bit timer (timer 1:2) operates as a 32-bit timer with no prescaler.

Independent of the normal timer behavior, eight compare registers (CMP*n*) are compared against the value of the TIM12 register when the PLUSEN bit in TGCR is set. Upon a successful non-zero match, an interrupt and a DMA event are generated without affecting the TIM12 value, behavior, or associated counter registers. Note that some timer instantiations may not map the CMP interrupt and DMA events to the CPU and DMA engines (see your device-specific data manual for information).

Figure 28-6. Dual 32-Bit Timers Unchained Mode Block Diagram



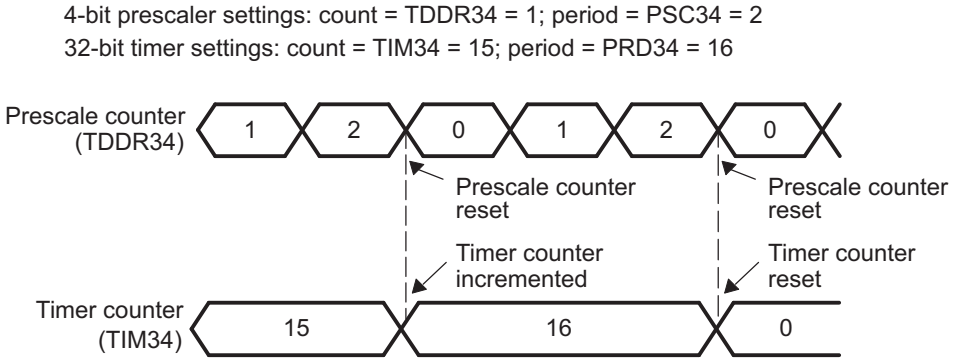
Copyright © 2013–2016, Texas Instruments Incorporated

28.1.5.4.2.2.1 32-Bit Timer With a 4-Bit Prescaler

In the unchained mode, the 4-bit prescale is clocked by the internal clock. The 4-bit prescaler uses the timer divide-down ratio (TDDR34) bit in TGCR to form a 4-bit prescale counter register and the prescale counter bits (PSC34) to form a 4-bit prescale period register (see Figure 28-6). When the timer is enabled, the prescale counter starts incrementing by 1 at every timer input clock cycle. One cycle after the prescale counter matches the prescale period, a clock signal is generated for the 32-bit timer.

The 32-bit timer uses TIM34 as a 32-bit timer counter register and PRD34 as a 32-bit timer period register. The 32-bit timer is clocked by the output clock from the 4-bit prescaler (see the example in Figure 28-7). The timer counter increments by 1 at every prescaler output clock cycle. When the timer counter matches the period, a maskable timer interrupt (TINT34) and a timer DMA event (TEVT34) are generated. When the timer is configured in continuous mode, the timer counter is reset to 0 on the cycle after the timer counter reaches the timer period. The timer can be stopped, restarted, reset, or disabled using the TIM34RS bit in TGCR. For timer 3:4, the lower 16 bits of the timer control register (TCR) have no control.

Figure 28-7. Dual 32-Bit Timers Unchained Mode Example



28.1.5.4.2.2.2 32-Bit Timer with No Prescaler

The other 32-bit timer (timer 1:2) uses TIM12 as the 32-bit counter register and PRD12 as a 32-bit timer period register (see Figure 28-6). When the timer is enabled, the timer counter increments by 1 at every timer input clock cycle. When the timer counter matches the timer period, a maskable timer interrupt (TINT12), a timer DMA event (TEVT12), and a timer output event on TM64P_OUT12 are generated. When the timer is configured in continuous mode, the timer counter is reset to 0 on the cycle after the timer counter reaches the timer period. The timer can be stopped, restarted, reset, or disabled using the TIM12RS bit in TGCR. For timer 1:2, the upper 16 bit of the timer control register (TCR) have no control.

Copyright © 2013–2016, Texas Instruments Incorporated

28.1.5.4.2.2.3 Enabling the 32-Bit Unchained Mode Timer

The TIM12RS and TIM34RS bits in TGCR determine whether the timer is in reset, or if it is capable of operating. The TIM12RS bit controls the reset of the timer 1:2 side of the timer and the TIM34RS bit controls the reset of the timer 3:4 side of the timer. For the timer to operate, the TIM12RS and/or TIM34RS bits must be set to 1.

The ENAMODE*n* bit in the timer control register (TCR) controls whether the timer is disabled, enabled to run once, or enabled to run continuously.

- When the timer is disabled (ENAMODE*n* = 0), the timer does not run and maintains its current count value.
- When the timer is enabled for one time operation (ENAMODE*n* = 1), it counts up until the counter value equals the period value and then stops.
- When the timer is enabled for continuous operation (ENAMODE*n* = 2h), the counter counts up until it reaches the period value, then resets itself to zero and begins counting again.
- When the timer is enabled for continuous operation with period reload (ENAMODE*n* = 3h), the counter counts up until it reaches the period value, then resets itself to zero, reloads the period registers (PRD12 and/or PRD34) with the value in the period reload registers (REL12 and/or REL34), and begins counting again.

Table 28-4 shows the bit values in TGCR to configure the 32-bit timer in unchained mode.

Once the timer stops, if an external clock is used as the timer clock, the timer must remain disabled for at least one external clock period or the timer will not start counting again. When using the external clock, the count value is synchronized to the internal clock.

Note that when both the timer counter and timer period are cleared to 0, the timer can be enabled but the timer counter does not increment because the timer period is 0.

Table 28-4. 32-Bit Timer Unchained Mode Configurations

	TGCR Bit		TCR Bit	
32-Bit Timer Configuration	**TIM12RS**	**TIM34RS**	**ENAMODE12**	**ENAMODE34**
To place the 32-bit timer unchained mode with 4-bit prescaler in reset	x	0	x	0
To disable the 32-bit timer unchained mode with 4-bit prescaler (out of reset)	x	1h	x	0
To enable the 32-bit timer unchained mode with 4-bit prescaler for one-time operation	x	1h	x	1h
To enable the 32-bit timer unchained mode with 4-bit prescaler for continuous operation	x	1h	x	2h
To enable the 32-bit timer unchained mode with 4-bit prescaler for continuous operation with period reload	x	1h	x	3h
To place the 32-bit timer unchained mode with no prescaler in reset	0	x	0	x
To disable the 32-bit timer unchained mode with no prescaler (out of reset)	1h	x	0	x
To enable the 32-bit timer unchained mode with no prescaler for one-time operation	1h	x	1h	x
To enable the 32-bit timer unchained mode with no prescaler for continuous operation	1h	x	2h	x
To enable the 32-bit timer unchained mode with no prescaler for continuous operation with period reload	1h	x	3h	x

Copyright © 2013–2016, Texas Instruments Incorporated

28.1.5.4.2.2.4 32-Bit Timer Unchained Mode Configuration Procedure

To configure timer 1:2, follow the steps below:

1. Select 32-bit unchained mode (TIMMODE in TGCR).
2. Remove the timer 1:2 from reset (TIM12RS in TGCR).
3. Select the desired timer period for timer 1:2 (PRD12). Program with the desired timer period value - 1.
4. Select the desired clock source for timer 1:2 (CLKSRC12 in TCR).
5. Enable timer 1:2 (ENAMODE12 in TCR).
6. If ENAMODE12 = 3h, write the desired timer period for the next timer cycle in the period reload register (REL12). Program with the desired timer period value - 1. This step can be done at any time before the current timer cycle ends.

To configure timer 3:4, follow the steps below:

1. Select 32-bit unchained mode (TIMMODE in TGCR).
2. Remove the timer 3:4 from reset (TIM34RS in TGCR).
3. Select the desired timer period for timer 3:4 (PRD34). Program with the desired timer period value - 1.
4. Select the desired prescaler value for timer 3:4 (PSC34 in TGCR).
5. Enable timer 3:4 (ENAMODE34 in TCR).
6. If ENAMODE34 = 3h, write the desired timer period for the next timer cycle in the period reload register (REL34). Program with the desired timer period value - 1. This step can be done at any time before the current timer cycle ends.

28.1.5.4.2.2.5 Event Capture Mode

When the PLUSEN bit in the timer global control register (TGCR) is set, Event Capture Mode is available for TIM12 when the timer is configured in 32-bit unchained mode. When Event Capture Mode is enabled, the timer cycle is restarted when an external input event occurs on pin TM64P_IN12. In particular, when an external input event occurs, the timer stops counting, generates output events (TINT12, TEVT12, and TM64P_OUT12), copies values from the timer counter register TIM12 to the timer capture register CAP12, reloads the timer period register PRD12 if in continuous mode with period reload (ENAMODE = 3h), and then restarts counting in continuous mode. Event Capture Mode is available only when the timer clock source is the internal timer (CLKSRC = 0) and the timer is in continuous mode (ENAMODE = 2h or 3h).

Capture mode is enabled using the Capture mode enable bit CAPMODE12 in the timer control register (TCR). The type of input event is selected by the capture event mode bit CAPEVTMODE12 in the timer control register (TCR). All of the following input event types are available:

- Rising edge of input signal
- Falling edge of input signal
- Rising or falling edge of input signal

28.1.5.4.2.2.6 Timer Counter Register Read Reset Mode

Read Reset Mode is available when the PLUSEN bit in the timer global control register (TGCR) is set and the timer is configured in 32-bit unchained mode. When Read Reset Mode is enabled, the timer cycle will restart when the timer counter registers are read (TIM12 and/or TIM34). In particular, when the timer registers are read, the timer stops counting, copies values from the timer counter registers (TIM12 and/or TIM34) to the timer capture registers (CAP12 and/or CAP34), reloads the timer period registers (PRD12 and/or PRD34) if in continuous mode with period reload (ENAMODE = 3h), and then restarts counting in continuous mode. Timer output events (TINT*n*, TEVT*n*, and TM64P_OUT*n*) are not generated during this process. Read Reset Mode is enabled using the read reset mode enable bit (READRSTMODE) in the timer control register (TCR).

Copyright © 2013–2016, Texas Instruments Incorporated

28.1.5.4.3 Timer Capture Registers

When the timer has a timeout due to a normal expiration of timer, external input event in Event Capture Mode, or read of timer counter registers in Read Reset Mode, the values of the timer counter registers (TIM12 and TIM34) are copied onto the timer counter capture registers (CAP12 and CAP34). Note that the value in TDDR is not captured when a read of TIM34 happens.

28.1.5.4.4 Counter and Period Registers Used in GP Timer Modes

Table 28-5 summarizes how the counter registers (TIM*n*) and period registers (PRD*n*) are used in each GP timer mode.

Table 28-5. Counter and Period Registers Used in GP Timer Modes

Timer Mode	Counter Registers	Period Registers
64-bit general-purpose	TIM34:TIM12	PRD34:PRD12
Dual 32-bit chained:		
Prescaler (Timer 3:4)	TIM34	PRD34
Timer (Timer 1:2)	TIM12	PRD12
Dual 32-bit unchained:		
Timer (Timer 1:2)	TIM12	PRD12
Timer with prescaler (Timer 3:4)	TDDR34 bits and TIM34	PSC34 bits and PRD34

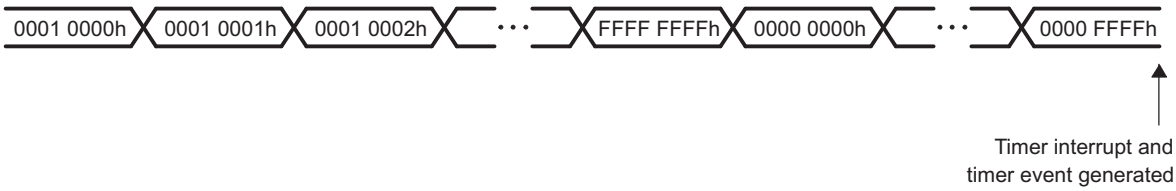
28.1.5.5 Timer Operation Boundary Conditions

The following boundary conditions affect the timer operation.

28.1.5.5.1 Timer Counter Overflow

Timer counter overflow can happen when the timer counter register is set to a value greater than the value in the timer period register. The counter reaches its maximum value (FFFF FFFFh or FFFF FFFF FFFF FFFFh), rolls over to 0, and continues counting until it reaches the timer period. An example is in Figure 28-8.

Figure 28-8. 32-Bit Timer Counter Overflow Example



28.1.5.5.2 Writing to Registers of an Active Timer

Writes to most timer registers are not allowed when the timer is active, except for setting the timer period reload registers (REL12 and REL34) and stopping and resetting the timers. In the 64-bit and dual 32-bit timer modes, registers that are protected by hardware are:

- TIM12
- TIM34
- PRD12
- PRD34
- TCR (except the ENAMODE bit)
- TGCR (except the TIM12RS and TIM34RS bits)

Copyright © 2013–2016, Texas Instruments Incorporated

28.1.5.6 General-Purpose Timer Power Management

The timer can be placed in reduced power modes to conserve power during periods of low activity. The power management of the peripheral is controlled by the processor Power and Sleep Controller (PSC). The PSC acts as a master controller for power management for all of the peripherals on the device. For detailed information on power management procedures using the PSC, see the *Power and Sleep Controller (PSC)* chapter. The timer can be placed in an idle mode to conserve power when it is not being used.

28.1.6 Architecture – Watchdog Timer Mode

This section describes the use of the timer as a watchdog timer. In order to fully function in watchdog timer mode, the timer must be internally connected to the device hardware reset signal. For information on which timer instantiation can function as a watchdog timer, see your device-specific data manual.

28.1.6.1 Watchdog Timer

As a 64-bit watchdog timer, the peripheral can be used to prevent system lockup when the software becomes trapped in loops with no controlled exit.

After a hardware reset, the watchdog timer is disabled. The timer then can be configured as a watchdog timer using the timer mode (TIMMODE) bit in the timer global control register (TGCR) and the watchdog timer enable (WDEN) bit in the watchdog timer control register (WDTCR). In the watchdog timer mode, the timer requires a special service sequence to be executed periodically. Without this periodic servicing, the timer counter increments until it matches the timer period and causes a watchdog timeout event.

When the timeout event occurs, the watchdog timer resets the entire processor.

28.1.6.2 Watchdog Timer Mode Restrictions

The watchdog timer mode has the following restrictions:

- No external clock source
- No one-time enabling

28.1.6.3 Watchdog Timer Mode Operation

The watchdog timer mode is selected and enabled when:

- TIMMODE = 2h in TGCR
- WDEN = 1 in WDTCR

Figure 28-9 shows the timer when it is used in the watchdog timer mode. The counter registers (TIM12 and TIM34) form a 64-bit timer counter register and the period registers (PRD12 and PRD34) form a 64-bit period register. When the timer counter matches the timer period, the timer generates a watchdog timeout event which resets the entire processor.

To activate the watchdog timer, a certain sequence of events must be followed, as shown in the state diagram of Figure 28-10.

Once the watchdog timer is activated, it can be disabled only by a watchdog timeout event or by a hardware reset. A special key sequence is required to prevent the watchdog timer from being accidentally serviced while the software is trapped in a loop or by some other software failure.

To prevent a watchdog timeout event, the timer has to be serviced periodically by writing A5C6h followed by DA7Eh to the watchdog timer service key (WDKEY) bits in WDTCR before the timer finishes counting up. Both A5C6h and DA7Eh are allowed to be written to the WDKEY bits, but only the correct sequence of A5C6h followed by DA7Eh to the WDKEY bits services the watchdog timer. Any other writes to the WDKEY bits triggers the watchdog timeout event immediately.

When the watchdog timer is in the Timeout state, the watchdog timer is disabled, the WDEN bit is cleared to 0, and the timer is reset.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 28-9. Watchdog Timer Mode Block Diagram

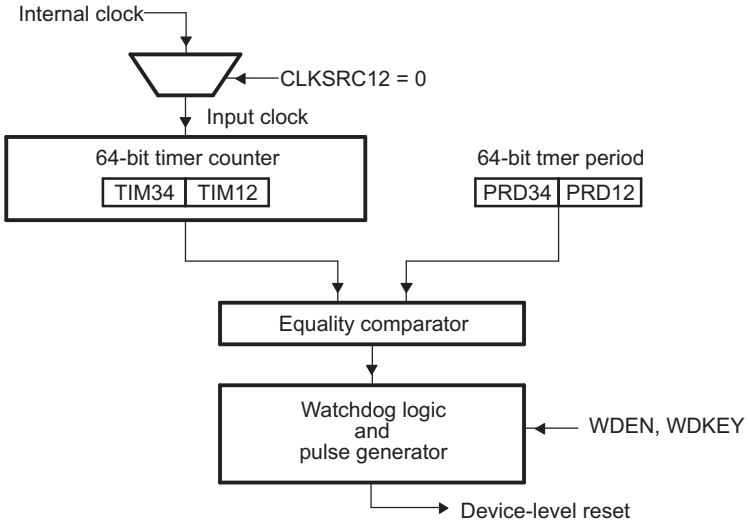
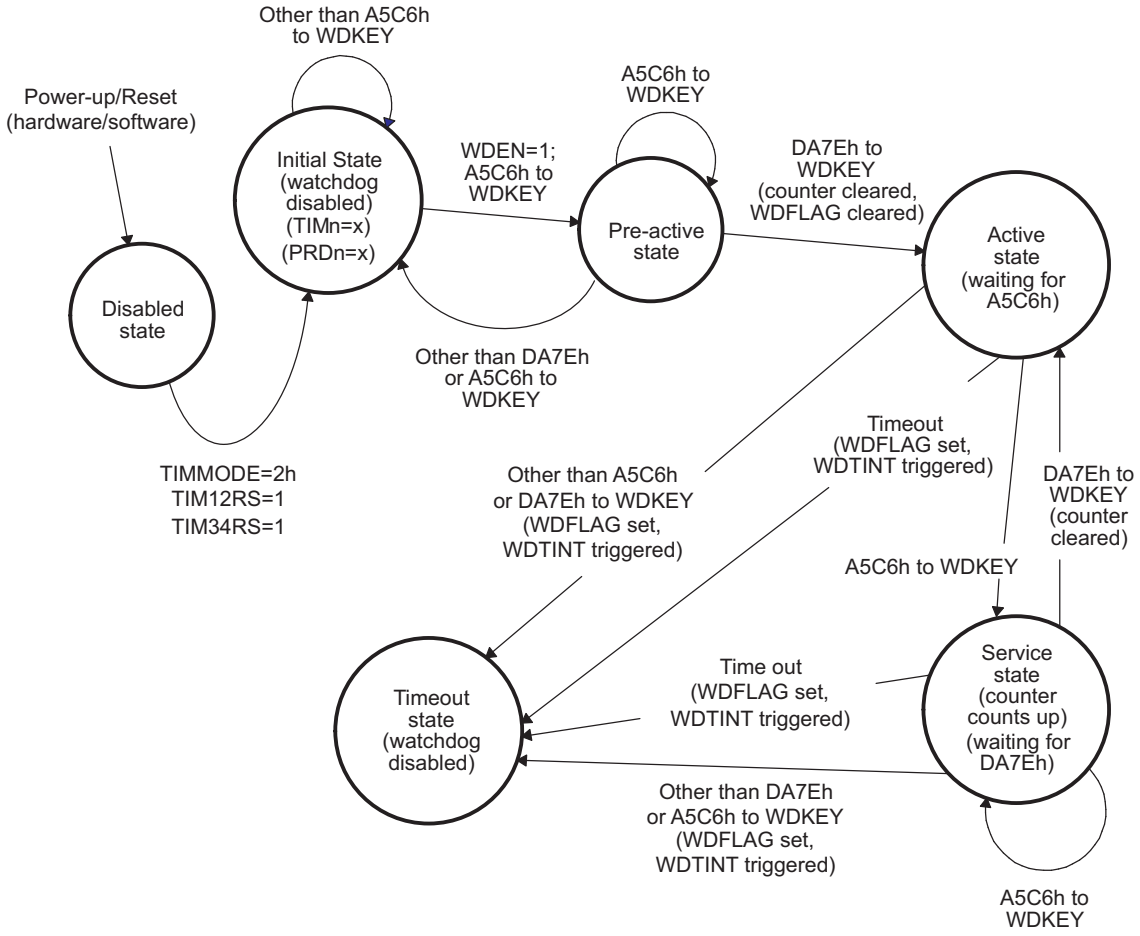


Figure 28-10. Watchdog Timer Operation State Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

After a hardware reset, the watchdog timer is disabled; however, reads or writes to the watchdog timer registers are allowed. Once the WDEN bit is set (enabling the watchdog timer) and A5C6h is written to the WDKEY bits, the watchdog timer enters the Pre-active state. In the Pre-active state:

- A write to WDTCR is allowed only when the write comes with the correct key (A5C6h or DA7Eh) to the WDKEY bits.
- A write of DA7Eh to the WDKEY bits when the WDEN bit is set to 1 resets the counters and activates the watchdog timer.

The watchdog timer must be configured before the watchdog timer enters the Active state. The WDEN bit must be set to 1 before writing DA7Eh to the WDKEY bits in the Pre-active state. Every time the watchdog timer is serviced by the correct WDKEY sequence, the watchdog timer counter is automatically reset.

28.1.6.4 Watchdog Timer Register Write Protection

Once the watchdog timer enters the Pre-active state (see Figure 28-10), writes to TIM12, TIM34, PRD12, PRD34, and WDTCR are write protected (except for the WDKEY field). While the watchdog timer is in the Timeout state, writing to the WDEN bit has no effect.

Once the watchdog timer enters its Initial state (see Figure 28-10), do not write to TGCR.

28.1.6.5 Watchdog Timer Power Management

The watchdog timer cannot be placed in power-down mode.

28.1.7 Reset Considerations

The timer has two reset sources: hardware reset and the timer reset (TIM12RS and TIM34RS) bits in the timer global control register (TGCR).

28.1.7.1 Software Reset Considerations

When the TIM12RS bit in TGCR is cleared to 0, the TIM12 register is held with the current value.

When the TIM34RS bit in TGCR is cleared to 0, the TIM34 register is held with the current value.

28.1.7.2 Hardware Reset Considerations

When a hardware reset is asserted, all timer registers are set to their default values.

28.1.8 Interrupt Support

Each of the timers can send either one of two separate interrupt events (TINT*n*) to the CPU, depending on the operating mode of the timer. The timer interrupts are generated when the count value in the counter register reaches the value specified in the period register.

When the PLUSEN bit in the timer global control register (TGCR) is set, matches between TIM12 and CMP*n* in dual 32-bit unchained mode will also generate interrupts. Setting the PLUSEN bit also enables additional features for control, status, and generation of interrupts. See Section 28.1.11 for more information.

28.1.9 DMA Event Support

Each of the timers can send either one of two separate timer events (TEVT*n*) to the DMA engine, depending on the operating mode of the timer. The timer events are generated when the count value in the counters register reaches the value specified in the period register.

When the PLUSEN bit in the timer global control register (TGCR) is set, matches between TIM12 and CMP*n* in dual 32-bit unchained mode will also generate DMA events. Setting the PLUSEN bit also enables additional features for control, status, and generation of dma events are enabled. See Section 28.1.11 for more information.

Copyright © 2013–2016, Texas Instruments Incorporated

28.1.10 TM64P_OUT Event Support

The timer can generate an output pulse (Figure 28-11) or clock (Figure 28-12) signals on the TM64P_OUT12 pin. The output signal is generated when the count value in the counter registers reaches the value specified in the period registers (TSTAT12 drives the TM64P_OUT12 pin). Table 28-6 gives equations for various TSTAT12 timing parameters in pulse and clock modes.

The output mode is selected with the clock/pulse bit (CP*n*) in the timer control register (TCR). In pulse mode, the PWID12 bit in TCR sets the pulse width between 1 to 4 timer clock periods. The TM64P_OUT12 pin may be inverted using the INVOUTP12 bit in TCR.

Figure 28-11. Timer Operation in Pulse Mode (CP*n* = 0)

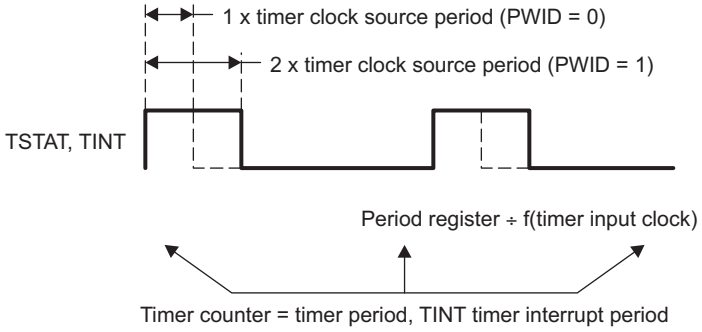


Figure 28-12. Timer Operation in Clock Mode (CP*n* = 1)

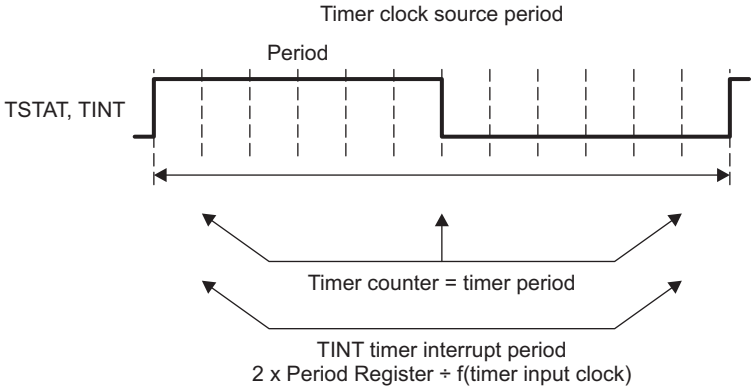


Table 28-6. TSTAT Parameters in Pulse and Clock Modes

Mode	Frequency	Period	Width High	Width Low
Pulse	$\frac{f\ (clock\ source)}{timer\ period\ register}$	$\frac{timer\ period\ register}{f\ (clock\ source)}$	$\frac{(PWID + 1)}{f\ (clock\ source)}$	$\frac{timer\ period\ register - (PWID + 1)}{f\ (clock\ source)}$
Clock	$\frac{f\ (clock\ source)}{2 \times timer\ period\ register}$	$\frac{2 \times timer\ period\ register}{f\ (clock\ source)}$	$\frac{timer\ period\ register}{f\ (clock\ source)}$	$\frac{timer\ period\ register}{f\ (clock\ source)}$

Copyright © 2013–2016, Texas Instruments Incorporated

External Timer Pin GPIO Mode

The external timer pins (TM64P_IN12 and TM64P_OUT12) can be individually configured to function as general-purpose input/output (GPIO) pins. In GPIO mode, the pins are able to detect and drive arbitrary data. The pins are also able to source external interrupt events. Some timer instantiations may not have external pins, see your device-specific data manual for pin information.

The GPIO interrupt and GPIO enable register (GPINTGPEN) enables the GPIO mode and associated interrupts. The GPIO data and GPIO direction register (GPDATGPDIR) determines if GPIO-enabled pins are used as input or output pins; and it is the means by which data is read-from or written-to the GPIO pins.

Normal timer counting modes cannot be used when the GPIO mode is enabled -- TIM12RS in the timer global control register (TGCR) cannot be brought out of reset when either GPENO12 or GPENI12 in GPINTGPEN is asserted.

28.1.11 Interrupt/DMA Event Generation Control and Status

When the PLUSEN bit in the timer global control register (TGCR) is set, the timer supports additional features for control and status of interrupt and DMA event generation. Interrupt/DMA events are generated when the count value in the timer counter registers reaches the value specified in the timer period registers and interrupt/DMA events are also generated when the Event Capture Mode is enabled and an external event occurs.

To generate events in the case when the value in the timer counter registers equals the value specified in the timer period registers, set the period compare interrupt enable bit (PRDINTEN*n*) in the interrupt control and status register (INTCTLSTAT). The event status for this case is reflected in the period compare interrupt status bit (PRDINTSTAT*n*), which is also in INTCTLSTAT. The PRDINTSTAT*n* bit is cleared by writing a 1 to the bit.

Similarly, to generate events in Event Capture Mode, set the event interrupt enable bit (EVTINTEN*n*) in INTCTLSTAT. The event status for this case is reflected in the external interrupt status bit (EVTINTSTAT*n*) in INTCTLSTAT. The EVTINTSTAT*n* bit is cleared by writing a 1 to the bit.

28.1.12 Power Management

The general-purpose timers can be placed in reduced power modes to conserve power during periods of low activity. The power management of the peripheral is controlled by the processor Power and Sleep Controller (PSC). The PSC acts as a master controller for power management for all of the peripherals on the device. For detailed information on power management procedures using the PSC, see the *Power and Sleep Controller (PSC)* chapter.

28.1.13 Emulation Considerations

Each timer has an emulation management register (EMUMGT). As shown in Table 28-7, the FREE and SOFT bits of EMUMGT determine how the timer responds to an emulation suspend event. An emulation suspend event corresponds to any type of emulator access to the CPU, such as a hardware or software breakpoint or a probe point.

Note that during emulation, the timer count values will increment once every timer peripheral clock (not CPU clock). So when single-steeping though code, the timer values will not update on every CPU clock cycle.

The timer can respond to emulation events from the CPU based on the configuration of the Emulation Suspend Source Register (SUSPSRC) in the System Configuration Module. See the *System Configuration (SYSCFG) Module* chapter for information on SUSPSRC and how it is configured.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 28-7. Timer Emulation Modes Selection

FREE	SOFT	Emulation Mode
0	0	The timer stops immediately.
0	1	The timer stops when the timer counter value increments and reaches the value in the timer period register.
1	x	The timer runs free regardless of SOFT bit status.

28.2 Registers

Table 28-8 lists the memory-mapped registers for the 64-bit Timer Plus. See your device-specific data manual for the memory address of these registers. All other register offset addresses not listed in Table 28-8 should be considered as reserved locations and the register contents should not be modified.

Table 28-8. Timer Registers

Offset	Acronym	Register Description	Section
0h	REVID	Revision ID Register	Section 28.2.1
4h	EMUMGT	Emulation Management Register	Section 28.2.2
8h	GPINTGPEN	GPIO Interrupt and GPIO Enable Register	Section 28.2.3
Ch	GPDATGPDIR	GPIO Data and GPIO Direction Register	Section 28.2.4
10h	TIM12	Timer Counter Register 12	Section 28.2.5
14h	TIM34	Timer Counter Register 34	Section 28.2.5
18h	PRD12	Timer Period Register 12	Section 28.2.6
1Ch	PRD34	Timer Period Register 34	Section 28.2.6
20h	TCR	Timer Control Register	Section 28.2.7
24h	TGCR	Timer Global Control Register	Section 28.2.8
28h	WDTCR	Watchdog Timer Control Register	Section 28.2.9
34h	REL12	Timer Reload Register 12	Section 28.2.10
38h	REL34	Timer Reload Register 34	Section 28.2.11
3Ch	CAP12	Timer Capture Register 12	Section 28.2.12
40h	CAP34	Timer Capture Register 34	Section 28.2.13
44h	INTCTLSTAT	Timer Interrupt Control and Status Register	Section 28.2.14
60h	CMP0	Compare Register 0	Timer Compare Registers (CMP0-CMP7)
64h	CMP1	Compare Register 1	Timer Compare Registers (CMP0-CMP7)
68h	CMP2	Compare Register 2	Timer Compare Registers (CMP0-CMP7)
6Ch	CMP3	Compare Register 3	Timer Compare Registers (CMP0-CMP7)
70h	CMP4	Compare Register 4	Timer Compare Registers (CMP0-CMP7)
74h	CMP5	Compare Register 5	Timer Compare Registers (CMP0-CMP7)
78h	CMP6	Compare Register 6	Timer Compare Registers (CMP0-CMP7)

Copyright © 2013–2016, Texas Instruments Incorporated

Table 28-8. Timer Registers (continued)

Offset	Acronym	Register Description	Section
7Ch	CMP7	Compare Register 7	Timer Compare Registers (CMP0-CMP7)

Copyright © 2013–2016, Texas Instruments Incorporated

28.2.1 Revision ID Register (REVID)

The revision ID register (REVID) contains the peripheral revision. The REVID is shown in Figure 28-13 and described in Table 28-9.

Figure 28-13. Revision ID Register (REVID)

31	0
REV	
R-4472 020Ch	

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

Table 28-9. Revision ID Register (REVID) Field Descriptions

Bit	Field	Value	Description
31-0	REV	4472 020Ch	Revision ID of the Timer.

28.2.2 Emulation Management Register (EMUMGT)

The emulation management register (EMUMGT) is shown in Figure 28-14 and described in Table 28-10.

Figure 28-14. Emulation Management Register (EMUMGT)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		SOFT	FREE
R-0		R/W-0	R/W-0

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

Table 28-10. Emulation Management Register (EMUMGT) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	SOFT		Determines emulation mode functionality of the timer. When the FREE bit is cleared to 0, the SOFT bit selects the timer mode.
		0	The timer stops immediately.
		1	The timer stops when the counter increments and reaches the value in the timer period register (PRD*n*).
0	FREE		Determines emulation mode functionality of the timer. When the FREE bit is cleared to 0, the SOFT bit selects the timer mode.
		0	The SOFT bit selects the timer mode.
		1	The timer runs free regardless of the SOFT bit.

Copyright © 2013–2016, Texas Instruments Incorporated

28.2.3 GPIO Interrupt Control and Enable Register (GPINTGPEN)

The GPIO interrupt control and enable register (GPINTGPEN) is shown in Figure 28-15 and described in Table 28-11.

Figure 28-15. GPIO Interrupt Control and Enable Register (GPINTGPEN)

31	24
Reserved	
R/W-0	

23	18	17	16
Reserved		GPENO12	GPENI12
R/W-0		R/W-0	R/W-0

15	8
Reserved	
R/W-0	

7	6	5	4	3	2	1	0
Reserved		GPINT12INVO	GPINT12INVI	Reserved		GPINT12ENO	GPINT12ENI
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 28-11. GPIO Interrupt Control and Enable Register (GPINTGPEN) Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17	GPENO12		Enable TM64P_OUT12 to function in GPIO mode.
		0	TM64P_OUT12 is used as a TIMER output pin.
		1	TM64P_OUT12 is used as a GPIO pin.
16	GPENI12		Enable TM64P_IN12 to function in GPIO mode
		0	TM64P_IN12 is used as a TIMER input pin.
		1	TM64P_IN12 is used as a GPIO pin.
15-6	Reserved	0	Reserved
5	GPINT12INVO		Invert interrupt/event signal from TM64P_OUT12 when GPINT12ENO = 1.
		0	Rising signal edge on TM64P_OUT12 generates the interrupt/event.
		1	Falling signal edge on TM64P_OUT12 generates the interrupt/event.
4	GPINT12INVI		Invert interrupt/event signal for TM64P_IN12 when GPINT12ENI = 1.
		0	Rising signal edge on TM64P_IN12 generates the interrupt/event.
		1	Falling signal edge on TM64P_IN12 generates the interrupt/event.
3-2	Reserved	0	Reserved
1	GPINT12ENO		Enable TM64P_OUT12 to source interrupts/events in GPIO mode.
		0	Timer interrupts/events are sourced in TIMER mode.
		1	Timer interrupts/events are sourced externally from TM64P_OUT12.
0	GPINT12ENI		Enable TM64P_IN12 to source interrupts/events in GPIO mode.
		0	Timer interrupts/events are sourced in TIMER mode.
		1	Timer interrupts/events are sourced externally from TM64P_IN12.

Copyright © 2013–2016, Texas Instruments Incorporated

28.2.4 GPIO Data and Direction Register (GPDATGPDIR)

The GPIO data and direction register (GPDATGPDIR) is shown in Figure 28-16 and described in Table 28-12.

Figure 28-16. GPIO Data and Direction Register (GPDATGPDIR)

31	18	17	16
Reserved		GPDIRO12	GPDIRI12
R/W-0		R/W-0	R/W-0

15	2	1	0
Reserved		GPDATO12	GPDATI12
R/W-0		R/W-0	R/W-0

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

Table 28-12. GPIO Data and Direction Register (GPDATGPDIR) Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17	GPDIRO12		Select direction of TM64P_OUT12 in GPIO mode.
		0	TM64P_OUT12 functions as an input pin in GPIO mode.
		1	TM64P_OUT12 functions as an output pin in GPIO mode (TM64P_OUT12 cannot capture GPIO interrupt events when configured as output).
16	GPDIRI12		Select direction of TM64P_IN12 in GPIO mode.
		0	TM64P_IN12 functions as an input pin in GPIO mode.
		1	TM64P_IN12 functions as an output pin in GPIO mode (TM64P_IN12 cannot capture GPIO interrupt events when configured as output).
15-2	Reserved	0	Reserved
1	GPDATO12		Data on TM64P_OUT12 in GPIO mode. Only valid when GPENO12 = 1.
			When GPDIRO12 = 0 (input):
		0	TM64P_OUT12 is detected logic low.
		1	TM64P_OUT12 is detected logic high.
			When GPDIRO12 = 1 (output):
		0	TM64P_OUT12 is driven logic low.
		1	TM64P_OUT12 is driven logic high.
0	GPDATI12		Data on TM64P_IN12 in GPIO mode. Only valid when GPENI12 = 1.
			When GPDIRI12 = 0 (input):
		0	TM64P_IN12 is detected logic low.
		1	TM64P_IN12 is detected logic high.
			When GPDIRI12 = 1 (output):
		0	TM64P_IN12 is driven logic low.
		1	TM64P_IN12 is driven logic high.

Copyright © 2013–2016, Texas Instruments Incorporated

28.2.5 Timer Counter Registers (TIM12 and TIM34)

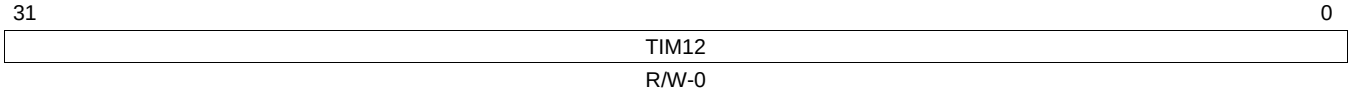
The timer counter register is a 64-bit wide register. This 64-bit register is divided into two 32-bit registers, TIM12 and TIM34.

In the dual 32-bit timer mode, the 64-bit register is divided with TIM12 acting as one 32-bit counter and TIM34 acting as another. These two registers can be configured as chained or unchained.

28.2.5.1 Timer Counter Register 12 (TIM12)

The timer counter register 12 (TIM12) is shown in Figure 28-17 and described in Table 28-13

Figure 28-17. Timer Counter Register 12 (TIM12)



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

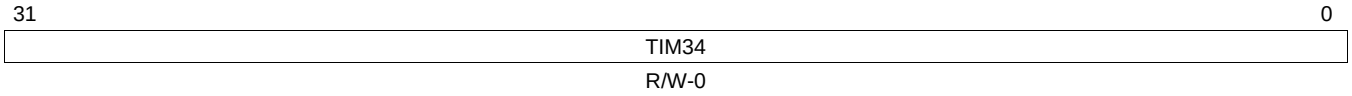
Table 28-13. Timer Counter Register 12 (TIM12) Field Descriptions

Bit	Field	Value	Description
31-0	TIM12	0-FFFF FFFFh	TIM12 count bits. This 32-bit value is the current count of the main counter.

28.2.5.2 Timer Counter Register 34 (TIM34)

The timer counter register 34 (TIM34) is shown in Figure 28-18 and described in Table 28-14.

Figure 28-18. Timer Counter Register 34 (TIM34)



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

Table 28-14. Timer Counter Register 34 (TIM34) Field Descriptions

Bit	Field	Value	Description
31-0	TIM34	0-FFFF FFFFh	TIM34 count bits. This 32-bit value is the current count of the main counter.

Copyright © 2013–2016, Texas Instruments Incorporated

28.2.6 Timer Period Registers (PRD12 and PRD34)

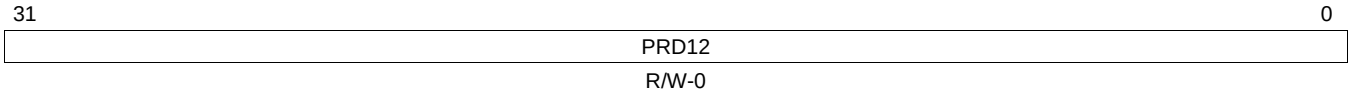
The timer period register is a 64-bit wide register. This 64-bit register is divided into two 32-bit registers, PRD12 and PRD34.

Similar to TIM*n* in the dual 32-bit timer mode, PRD*n* can be divided into 2 registers: for timer 1:2, PRD12 and for timer 3:4, PRD34. These two registers can be used in conjunction with the two timer counter registers TIM12 and TIM34.

28.2.6.1 Timer Period Register 12 (PRD12)

The timer period register 12 (PRD12) is shown in Figure 28-19 and described in Table 28-15.

Figure 28-19. Timer Period Register 12 (PRD12)



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

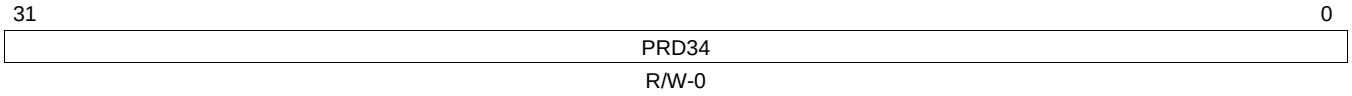
Table 28-15. Timer Period Register (PRD12) Field Descriptions

Bit	Field	Value	Description
31-0	PRD12	0-FFFF FFFFh	PRD12 period bits. This 32-bit value is the number of timer input clock cycles to count.

28.2.6.2 Timer Period Register 34 (PRD34)

The timer period register 34 (PRD34) is shown in Figure 28-20 and described in Table 28-16.

Figure 28-20. Timer Period Register 34 (PRD34)



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

Table 28-16. Timer Period Register (PRD34) Field Descriptions

Bit	Field	Value	Description
31-0	PRD34	0-FFFF FFFFh	PRD34 period bits. This 32-bit value is the number of timer input clock cycles to count.

Copyright © 2013–2016, Texas Instruments Incorporated

28.2.7 Timer Control Register (TCR)

The timer control register (TCR) is shown in Figure 28-21 and described in Table 28-17.

Figure 28-21. Timer Control Register (TCR)

31	27	26	25	24
Reserved		READRSTMODE34	Reserved	
R/W-0		R/W-0	R/W-0	

23	22	21	16
ENAMODE34		Reserved	
R/W-0		R/W-0	

15	14	13	12	11	10	9	8
Reserved		CAPVTMODE12		CAPMODE12	READRSTMODE12	TIEN12	CLKSRC12
R-0		R/W-0		R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
ENAMODE12		PWID12		CP12	INVINP12	INVOUTP12	TSTAT12
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 28-17. Timer Control Register (TCR) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reserved
26	READRSTMODE34		Read reset mode enable bit. Determines the effect of a timer counter read on TIM34. Read reset mode is only available in dual 32-bit unchained. Output events (interrupt/EDMA/other) are not generated when read reset occurs.
		0	There is no effect when timer counter register TIM34 is read.
		1	Timer counter is reset when timer counter register TIM34 is read.
25-24	Reserved	0	Reserved
23-22	ENAMODE34	0-3h	Enabling mode: determines the enabling modes fo the timer.
		0	The timer is disabled (not counting) and maintains current value.
		1h	The timer is enabled one time. The timer stops after the counter reaches the period.
		2h	The timer is enabled continuously, TIM34 increments until the timer counter matches the period, resets the timer counter to 0 on the cycle after matching and continues.
		3h	The timer is enabled continuously with period reload, TIM*n* increments until the timer counter matches the period, resets the timer counter to 0 on the cycle after matching, reloads the period register with the values in the reload registers (REL*n*), and continues counting.
21-14	Reserved	0	Reserved
13-12	CAPEVTMODE12	0-3h	Capture event mode. Uses these bits to specify the type of event for Capture mode.
		0	Event occurs on timer input rising edge.
		1h	Event occurs on time input falling edge.
		2h	Event occurs on both rising and falling edges.
		3h	Reserved
11	CAPMODE12		Capture mode enable bit. Determines if external event can reset timer. Capture mode is only available in dual 32-bit unchained mode and when CLKSRC = 0 and ENAMODE = 2h or 3h. Output events (interrupt/EDMA/other) are generated when capture mode event occurs.
		0	Timer is not in capture mode.
		1	Timer is in capture mode. External event can reset timer.
10	READRSTMODE12		Read reset mode enable bit. Determines the effect of a timer counter read on TIM12. Read reset mode is only available in dual 32-bit unchained. Output events (interrupt/EDMA/other) are not generated when read reset occurs.
		0	There is no effect when timer counter register TIM12 is read.
		1	Timer counter is reset when timer counter register TIM12 is read.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 28-17. Timer Control Register (TCR) Field Descriptions (continued)

Bit	Field	Value	Description
9	TIEN12		Timer input gate enable bit. Allows timer input pin TM64P_IN12 to gate the internal timer clock source (CLKSRC = 0). Timer starts counting when TM64P_IN12 transitions from low to high. Timer stops counting when TM64P_IN12 transitions from high to low.
		0	Timer clock is not gated by TM64P_IN12.
		1	Timer clock is gated by TM64P_IN12.
8	CLKSRC12		CLKSRC determines the selected clock source for the timer.
		0	Internal clock
		1	External clock on TM64P_IN12
7-6	ENAMODE12	0-3h	Enabling mode: determines the enabling modes fo the timer.
		0	The timer is disabled (not counting) and maintains current value.
		1h	The timer is enabled one time. The timer stops after the counter reaches the period.
		2h	The timer is enabled continuously, TIM*n* increments until the timer counter matches the period, resets the timer counter to 0 on the cycle after matching and continues.
		3h	The timer is enabled continuously with period reload, TIM*n* increments until the timer counter matches the period, resets the timer counter to 0 on the cycle after matching, reloads the period register with the values in the reload registers (REL*n*), and continues counting.
5-4	PWID12	0-3h	Pulse width - Determines the pulse width on the TSTAT12 bit (and the TM64P_OUT12 pin) when the clock/pulse mode is set to pulse.
		0	TSTAT12 stays active for one timer clock cycle when the timer counter reaches the period.
		1h	TSTAT12 stays active for two timer clock cycles when the timer counter reaches the period.
		2h	TSTAT12 stays active for three timer clock cycles when the timer counter reaches the period.
		3h	TSTAT12 stays active for four timer clock cycles when the timer counter reaches the period.
3	CP12		Clock/Pulse bit - Determines whether the TM64P_OUT12 output event should behave as a 50% duty-cycle clock or a signal pulse.
		0	Pulse Mode. TM64P_OUT12 goes active after the timer counter reaches the period. The pulse width is determined by PWID12.
		1	Clock Mode. TM64P_OUT12 will behave as a 50% duty cycle signal. It toggles high-to-low or low-to-high when the timer counter reaches zero.
2	INVINP12		Invert TM64P_IN12. Only affects operation if CLKSRC = 1.
		0	Uninverted TM64P_IN12 signal drives timer.
		1	Inverted TM64P_IN12 signal drives timer.
1	INVOUTP12		Invert TM64P_OUT12.
		0	TM64P_OUT12 signal is not inverted.
		1	TM64P_OUT12 signal is inverted.
0	TSTAT12		Timer status. Drives the value of timer output TM64P_OUT12 when it is configured to function as timer output.
		0	TM64P_OUT12 signal is not asserted.
		1	TM64P_OUT12 signal is asserted.

Copyright © 2013–2016, Texas Instruments Incorporated

28.2.8 Timer Global Control Register (TGCR)

The timer global control register (TGCR) is shown in Figure 28-22 and described in Table 28-18.

Figure 28-22. Timer Global Control Register (TGCR)

31							16
Reserved							
R-0							
15		12	11				8
TDDR34			PSC34				
R/W-0			R/W-0				
7	5	4	3	2	1	0	
Reserved		PLUSEN	TIMMODE		TIM34RS	TIM12RS	
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 28-18. Timer Global Control Register (TGCR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-12	TDDR34	0-Fh	Timer linear divide-down ratio specifies the timer divide-down ratio for timer 3:4. When the timer is enabled, TDDR34 increments every timer clock. The TIM34 counter increments on the cycle after TDDR34 matches PSC34. TDDR34 resets to 0 and continues. When TIM34 matches PRD34, timer 3:4 stops, if timer 3:4 is enabled one time; TIM34 resets to 0 on the cycle after matching PRD34 and timer 3:4 continues, if timer 3:4 is enabled continuously.
11-8	PSC34	0-Fh	TIM34 pre-scalar counter specifies the count for timer 3:4.
7-5	Reserved	0	Reserved
4	PLUSEN		Enable new timer plus features.
		0	Enable backward compatibility. New timer features are unavailable.
		1	Disable backward compatibility. New timer features are available.
3-2	TIMMODE	0-3h	TIMMODE determines the timer mode.
		0	The timer is in 64-bit GP timer mode.
		1h	The timer is in dual 32-bit timer unchained mode.
		2h	The timer is in 64-bit watchdog timer mode.
		3h	The timer is in dual 32-bit timer, chained mode.
1	TIM34RS		Timer 3:4 reset.
		0	Timer 3:4 is in reset.
		1	Timer 3:4 is not in reset. Timer 3:4 can be used as a 32-bit timer. Note that for the timer to function properly in 64-bit timer mode, both TIM34RS and TIM12RS must be set to 1. Changing this bit does not affect the timer, if the timer is in the watchdog active state.
0	TIM12RS		Timer 1:2 reset.
		0	Timer 1:2 is in reset.
		1	Timer 1:2 is not in reset. Timer 1:2 can be used as a 32-bit timer. Note that for the timer to function properly in 64-bit timer mode, both TIM34RS and TIM12RS must be set to 1. Changing this bit does not affect the timer, if the timer is in the watchdog active state.

Copyright © 2013–2016, Texas Instruments Incorporated

28.2.9 Watchdog Timer Control Register (WDTCR)

The watchdog timer control register (WDTCR) is shown in Figure 28-23 and described in Table 28-19.

Figure 28-23. Watchdog Timer Control Register (WDTCR)

31	16
WDKEY	
R/W-0	

15	14	13	12	11	0
WDFLAG	WDEN	Reserved		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 28-19. Watchdog Timer Control Register (WDTCR) Field Descriptions

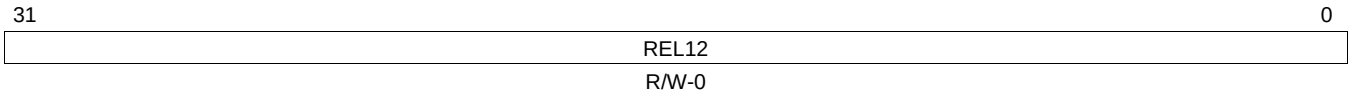
Bit	Field	Value	Description
31-16	WDKEY	0-FFFFh	16-bit watchdog timer service key. Only the sequence of an A5C6h followed by a DA7Eh services the watchdog. Not applicable in regular timer mode.
15	WDFLAG		Watchdog flag bit. WDFLAG can be cleared by enabling the watchdog timer, by device reset, or being written with 1. It is set by a watchdog time-out.
		0	No watchdog time-out occurred.
		1	Watchdog time-out occurred.
14	WDEN		Watchdog timer enable bit.
		0	Disable watchdog timer
		1	Enable watchdog timer
13-12	Reserved	0	Reserved. This bit field must be written as 00b.
11-0	Reserved	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

28.2.10 Timer Reload Register 12 (REL12)

The timer reload register 12 (REL12) is shown in Figure 28-24 and described in Table 28-20.

Figure 28-24. Timer Reload Register 12 (REL12)



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

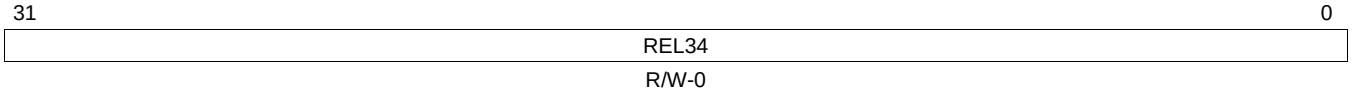
Table 28-20. Timer Reload Register 12 (REL12) Field Descriptions

Bit	Field	Value	Description
31- 0	REL12	0-FFFF FFFFh	Period reload bits.

28.2.11 Timer Reload Register 34 (REL34)

The timer reload register 34 (REL34) is shown in Figure 28-25 and described in Table 28-21.

Figure 28-25. Timer Reload Register 34 (REL34)



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

Table 28-21. Timer Reload Register 34 (REL34) Field Descriptions

Bit	Field	Value	Description
31- 0	REL34	0-FFFF FFFFh	Period reload bits.

Copyright © 2013–2016, Texas Instruments Incorporated

28.2.12 Timer Capture Register 12 (CAP12)

The timer capture register 12 (CAP12) is shown in Figure 28-26 and described in Table 28-22.

Figure 28-26. Timer Capture Register 12 (CAP12)

31	0
CAP12	
R/W-0	

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

Table 28-22. Timer Capture Register 12 (CAP12) Field Descriptions

Bit	Field	Value	Description
31- 0	CAP12	0-FFFF FFFFh	Captured timer counter bits.

28.2.13 Timer Capture Register 34 (CAP34)

The timer capture register 34 (CAP34) is shown in Figure 28-27 and described in Table 28-23.

Figure 28-27. Timer Capture Register 34 (CAP34)

31	0
CAP34	
R/W-0	

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

Table 28-23. Timer Capture Register 34 (CAP34) Field Descriptions

Bit	Field	Value	Description
31- 0	CAP34	0-FFFF FFFFh	Captured timer counter bits.

Copyright © 2013–2016, Texas Instruments Incorporated

28.2.14 Timer Interrupt Control and Status Register (INTCTLSTAT)

The timer interrupt control and status register (INTCTLSTAT) is shown in Figure 28-28 and described in Table 28-24.

Figure 28-28. Timer Interrupt Control and Status Register (INTCTLSTAT)

31	24
Reserved	
R-0	

23	20	19	18	17	16
Reserved		EVTINTSTAT34	EVTINTEN34	PRDINTSTAT34	PRDINTEN34
R-0		R/W1C-0	R/W-0	R/W1C-0	R/W-0

15	8
Reserved	
R-0	

7	4	3	2	1	0
Reserved		EVTINTSTAT12	EVTINTEN12	PRDINTSTAT12	PRDINTEN12
R-0		R/W1C-0	R/W-0	R/W1C-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear bit; -*n* = value after reset

Table 28-24. Timer Interrupt Control and Status Register (INTCTLSTAT) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reserved
19	EVTINTSTAT34		Interrupt status which reflects the condition that an external event caused a timeout when timer is in capture mode. Write a 1 to clear this bit.
		0	Interrupt has not occurred.
		1	Interrupt has occurred.
18	EVTINTEN34		Enables the interrupt generation when timer is in capture mode.
		0	Disable interrupt when in event capture mode.
		1	Enable interrupt when in event capture mode.
17	PRDINTSTAT34		Interrupt status which reflects the condition that timer counter matched the period register when timer is enabled. Write a 1 to clear this bit.
		0	Interrupt has not occurred.
		1	Interrupt has occurred.
16	PRDINTEN34		Enable interrupt generation when timer is enabled in 64-bit/32-bit chained/unchained/watchdog modes.
		0	Disable interrupt
		1	Enable interrupt
15-4	Reserved	0	Reserved
3	EVTINTSTAT12		Interrupt status which reflects the condition that an external event caused a timeout when timer is in capture mode. Write a 1 to clear this bit.
		0	Interrupt has not occurred.
		1	Interrupt has occurred.
2	EVTINTEN12		Enables the interrupt generation when timer is in capture mode.
		0	Disable interrupt when in event capture mode.
		1	Enable interrupt when in event capture mode.
1	PRDINTSTAT12		Interrupt status which reflects the condition that timer counter matched the period register when timer is enabled. Write a 1 to clear this bit.
		0	Interrupt has not occurred.
		1	Interrupt has occurred.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 28-24. Timer Interrupt Control and Status Register (INTCTLSTAT) Field Descriptions (continued)

Bit	Field	Value	Description
0	PRDINTEN12		Enable interrupt generation when timer is enabled in 64-bit/32-bit chained/unchained/watchdog modes.
		0	Disable interrupt
		1	Enable interrupt

Timer Compare Registers (CMP0-CMP7)

The timer compare register (CMP*n*) is shown in Figure 28-29 and described in Table 28-25.

Figure 28-29. Timer Compare Register (CMP*n*)

31	0
CMP*n*	
R/W-0	

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

Table 28-25. Timer Compare Register (CMP*n*) Field Descriptions

Bit	Field	Value	Description
31-0	CMP*n*	0-FFFF FFFFh	Timer compare register. When PLUSEN = 1 in the timer global control register (TGCR) and the timer is configured in 32-bit unchained mode, TIM12 is compared to all 8 compare registers (CMP0-CMP7). When CMP*n* matches TIM12, a timer CMP*n* interrupt and DMA event are generated. A CMP*n* match will not affect the TIM12 count or behavior.

Copyright © 2013–2016, Texas Instruments Incorporated

Universal Asynchronous Receiver/Transmitter (UART)

This chapter describes the universal asynchronous receiver/transmitter (UART) peripheral. See your device-specific data manual to determine how many UARTs are available on your device.

Topic **Page**

29.1 Introduction **1295**
29.2 Peripheral Architecture **1297**
29.3 Registers **1308**

Copyright © 2013–2016, Texas Instruments Incorporated

29.1 Introduction

29.1.1 Purpose of the Peripheral

The UART peripheral is based on the industry standard TL16C550 asynchronous communications element, which in turn is a functional upgrade of the TL16C450. Functionally similar to the TL16C450 on power up (single character or TL16C450 mode), the UART can be placed in an alternate FIFO (TL16C550) mode. This relieves the CPU of excessive software overhead by buffering received and transmitted characters. The receiver and transmitter FIFOs store up to 16 bytes including three additional bits of error status per byte for the receiver FIFO.

The UART performs serial-to-parallel conversions on data received from a peripheral device and parallel-to-serial conversion on data received from the CPU. The CPU can read the UART status at any time. The UART includes control capability and a processor interrupt system that can be tailored to minimize software management of the communications link.

The UART includes a programmable baud generator capable of dividing the UART input clock by divisors from 1 to 65535 and producing a 16× reference clock or a 13× reference clock for the internal transmitter and receiver logic. For detailed timing and electrical specifications for the UART, see your device-specific data manual.

29.1.2 Features

Check your device-specific data manual to see the list of features that are supported and that are not supported by the UART.

29.1.3 Functional Block Diagram

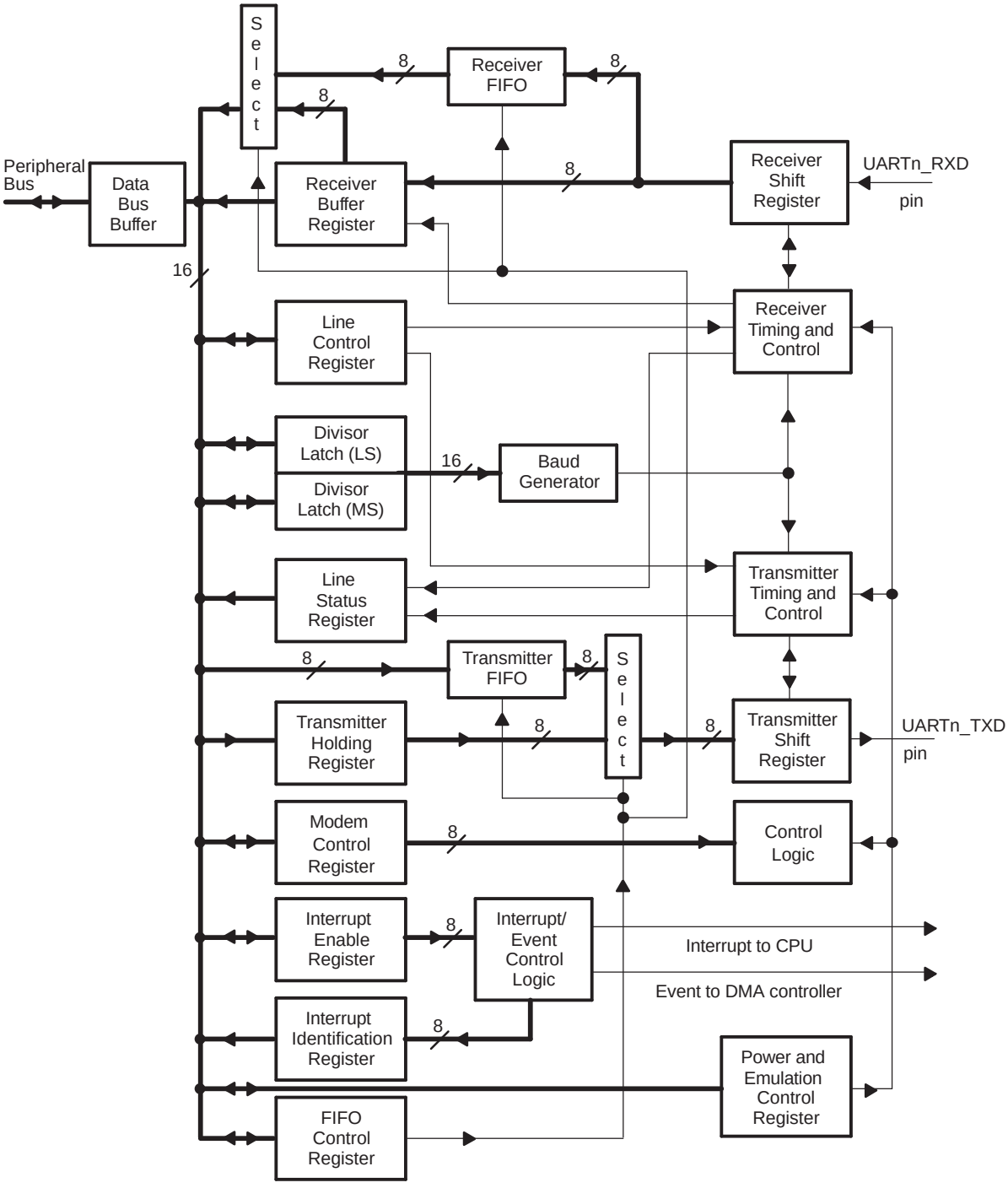
A functional block diagram of the UART is shown in Figure 29-1.

29.1.4 Industry Standard(s) Compliance Statement

The UART peripheral is based on the industry standard TL16C550 asynchronous communications element, which is a functional upgrade of the TL16C450. The information in this chapter assumes you are familiar with these standards.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 29-1. UART Block Diagram



NOTE: The value *n* indicates the applicable UART; that is, UART0, UART1, and so on.

Copyright © 2013–2016, Texas Instruments Incorporated

29.2 Peripheral Architecture

29.2.1 Clock Generation and Control

The UART bit clock is derived from an input clock to the UART. See your device-specific data manual to check the maximum data rate supported by the UART.

Figure 29-2 is a conceptual clock generation diagram for the UART. The processor clock generator receives a signal from an external clock source and produces a UART input clock with a programmed frequency. The UART contains a programmable baud generator that takes an input clock and divides it by a divisor in the range between 1 and (2^{16} - 1) to produce a baud clock (BCLK). The frequency of BCLK is sixteen times (16×) the baud rate (each received or transmitted bit lasts 16 BCLK cycles) or thirteen times (13×) the baud rate (each received or transmitted bit lasts 13 BCLK cycles). When the UART is receiving, the bit is sampled in the 8th BCLK cycle for 16× over sampling mode and on the 6th BCLK cycle for 13× over-sampling mode. The 16× or 13× reference clock is selected by configuring the OSM_SEL bit in the mode definition register (MDR). The formula to calculate the divisor is:

$$\text{Divisor} = \frac{\text{UART input clock frequency}}{\text{Desired baud rate} \times 16} \quad [\text{MDR.OSM_SEL} = 0]$$

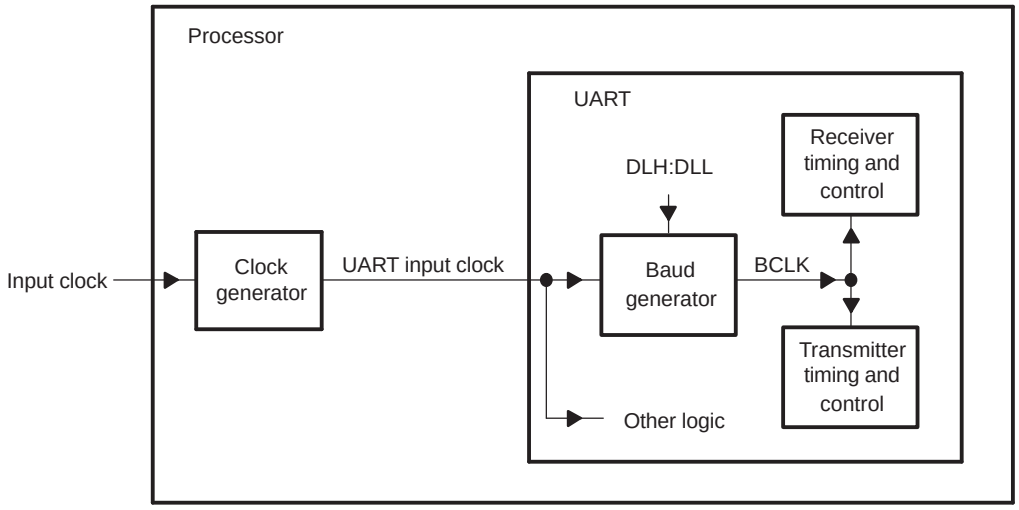
$$\text{Divisor} = \frac{\text{UART input clock frequency}}{\text{Desired baud rate} \times 13} \quad [\text{MDR.OSM_SEL} = 1]$$

Two 8-bit register fields (DLH and DLL), called divisor latches, hold this 16-bit divisor. DLH holds the most significant bits of the divisor, and DLL holds the least significant bits of the divisor. For information about these register fields, see Section 29.3. These divisor latches must be loaded during initialization of the UART in order to ensure desired operation of the baud generator. Writing to the divisor latches results in two wait states being inserted during the write access while the baud generator is loaded with the new value.

Figure 29-3 summarizes the relationship between the transferred data bit, BCLK, and the UART input clock. Note that the timing relationship depicted in Figure 29-3 shows that each bit lasts for 16 BCLK cycles . This is in case of 16x over-sampling mode. For 13× over-sampling mode each bit lasts for 13 BCLK cycles .

Example baud rates and divisor values relative to a 150-MHz UART input clock and 16× over-sampling mode are shown in Table 29-1.

Figure 29-2. UART Clock Generation Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 29-3. Relationships Between Data Bit, BCLK, and UART Input Clock

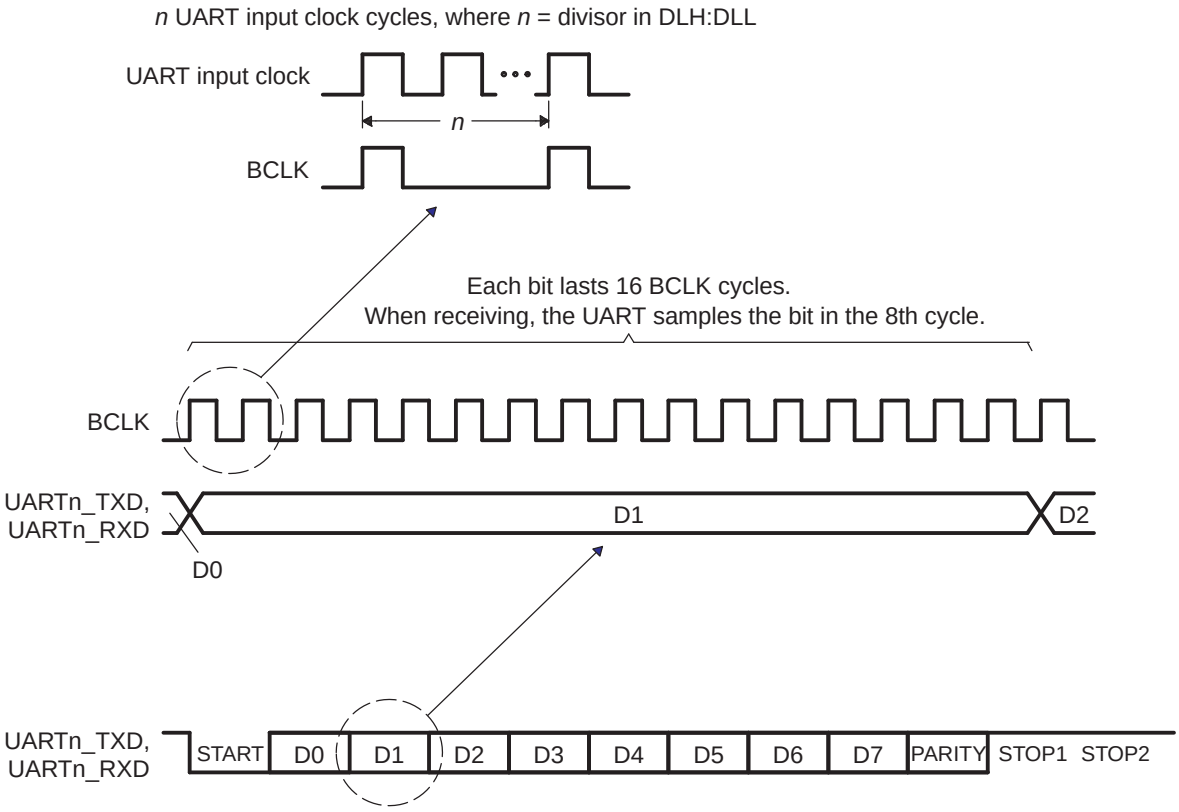


Table 29-1. Baud Rate Examples for 150-MHZ UART Input Clock and 16× Over-sampling Mode

Baud Rate	Divisor Value	Actual Baud Rate	Error (%)
2400	3906	2400.154	0.01
4800	1953	4800.372	0.01
9600	977	9595.701	-0.04
19200	488	19211.066	0.06
38400	244	38422.131	0.06
56000	167	56137.725	0.25
128000	73	129807.7	0.33
3000000	3	3125000	4.00

Table 29-2. Baud Rate Examples for 150-MHZ UART Input Clock and 13× Over-sampling Mode

Baud Rate	Divisor Value	Actual Baud Rate	Error (%)
2400	4808	2399	-0.01
4800	2404	4799.646	-0.01
9600	1202	9599.386	-0.01
19200	601	19198.771	-0.01
38400	300	38461.538	0.16
56000	206	56011.949	0.02
128000	90	128205.128	0.16
3000000	4	2884615.385	-4.00

Copyright © 2013–2016, Texas Instruments Incorporated

29.2.2 Signal Descriptions

The UARTs utilize a minimal number of signal connections to interface with external devices. The UART signal descriptions are included in Table 29-3. Note that the number of UARTs and their supported features vary on each device, see your device-specific data manual for more details.

Table 29-3. UART Signal Descriptions

Signal Name[1]	Signal Type	Function
UARTn_TXD	Output	Serial data transmit
UARTn_RXD	Input	Serial data receive
$\overline{\text{UARTn_CTS}}$ [2]	Input	Clear-to-Send handshaking signal
$\overline{\text{UARTn_RTS}}$ [2]	Output	Request-to-Send handshaking signal

(1) The value *n* indicates the applicable UART; that is, UART0, UART1, etc.
(2) This signal is not supported in all UARTs. See your device-specific data manual to check if it is supported.

29.2.3 Pin Multiplexing

Extensive pin multiplexing is used to accommodate the largest number of peripheral functions in the smallest possible package. Pin multiplexing is controlled using a combination of hardware configuration at device reset and software programmable register settings. See your device-specific data manual to determine how pin multiplexing affects the UART.

29.2.4 Protocol Description

29.2.4.1 Transmission

The UART transmitter section includes a transmitter hold register (THR) and a transmitter shift register (TSR). When the UART is in the FIFO mode, THR is a 16-byte FIFO. Transmitter section control is a function of the UART line control register (LCR). Based on the settings chosen in LCR, the UART transmitter sends the following to the receiving device:

- 1 START bit
- 5, 6, 7, or 8 data bits
- 1 PARITY bit (optional)
- 1, 1.5, or 2 STOP bits

29.2.4.2 Reception

The UART receiver section includes a receiver shift register (RSR) and a receiver buffer register (RBR). When the UART is in the FIFO mode, RBR is a 16-byte FIFO. Receiver section control is a function of the UART line control register (LCR). Based on the settings chosen in LCR, the UART receiver accepts the following from the transmitting device:

- 1 START bit
- 5, 6, 7, or 8 data bits
- 1 PARITY bit (optional)
- 1 STOP bit (any other STOP bits transferred with the above data are not detected)

Copyright © 2013–2016, Texas Instruments Incorporated

29.2.4.3 Data Format

The UART transmits in the following format:

1 START bit + data bits (5, 6, 7, 8) + 1 PARITY bit (optional) + STOP bit (1, 1.5, 2)

It transmits 1 START bit; 5, 6, 7, or 8 data bits, depending on the data width selection; 1 PARITY bit, if parity is selected; and 1, 1.5, or 2 STOP bits, depending on the STOP bit selection.

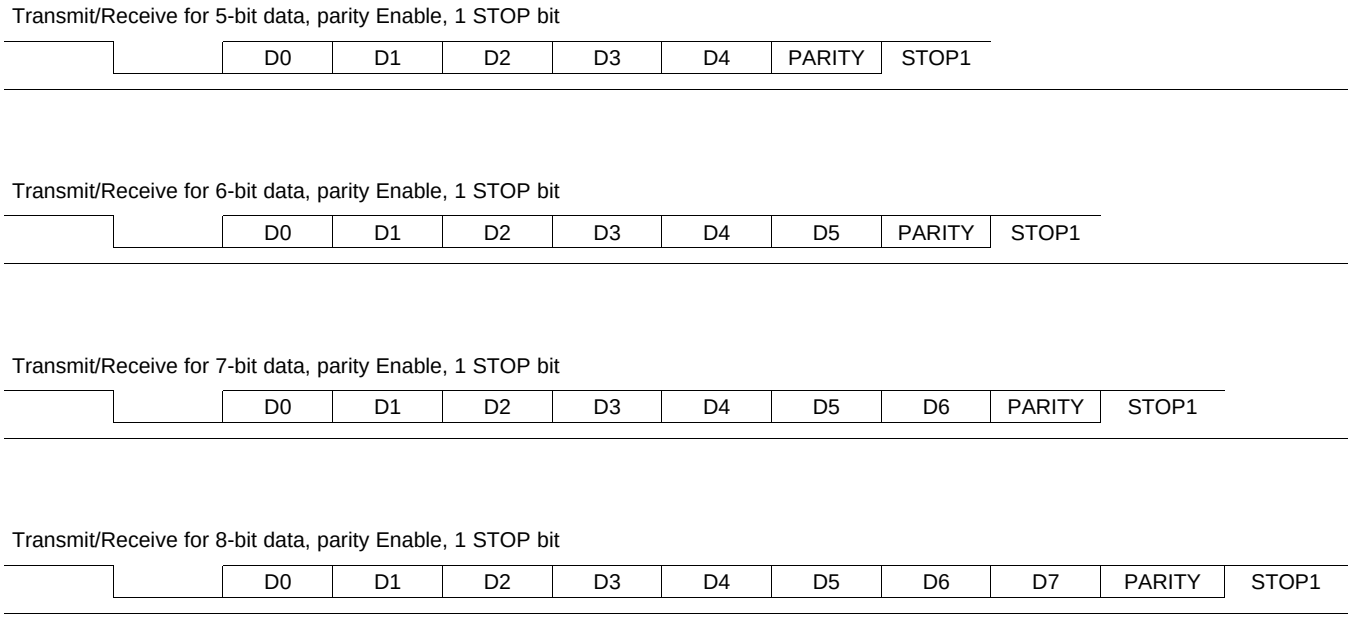
The UART receives in the following format:

1 START bit + data bits (5, 6, 7, 8) + 1 PARITY bit (optional) + 1 STOP bit

It receives 1 START bit; 5, 6, 7, or 8 data bits, depending on the data width selection; 1 PARITY bit, if parity is selected; and 1 STOP bit.

The protocol formats are shown in Figure 29-4.

Figure 29-4. UART Protocol Formats



Copyright © 2013–2016, Texas Instruments Incorporated

29.2.5 Operation

29.2.5.1 Transmission

The UART transmitter section includes a transmitter hold register (THR) and a transmitter shift register (TSR). When the UART is in the FIFO mode, THR is a 16-byte FIFO. Transmitter section control is a function of the UART line control register (LCR). Based on the settings chosen in LCR, the UART transmitter sends the following to the receiving device:

- 1 START bit
- 5, 6, 7, or 8 data bits
- 1 PARITY bit (optional)
- 1, 1.5, or 2 STOP bits

THR receives data from the internal data bus, and when TSR is ready, the UART moves the data from THR to TSR. The UART serializes the data in TSR and transmits the data on the UARTn_TXD pin.

In the non-FIFO mode, if THR is empty and the THR empty (THRE) interrupt is enabled in the interrupt enable register (IER), an interrupt is generated. This interrupt is cleared when a character is loaded into THR or the interrupt identification register (IIR) is read. In the FIFO mode, the interrupt is generated when the transmitter FIFO is empty, and it is cleared when at least one byte is loaded into the FIFO or IIR is read.

29.2.5.2 Reception

The UART receiver section includes a receiver shift register (RSR) and a receiver buffer register (RBR). When the UART is in the FIFO mode, RBR is a 16-byte FIFO. Timing is supplied by the 16× receiver clock. Receiver section control is a function of the UART line control register (LCR). Based on the settings chosen in LCR, the UART receiver accepts the following from the transmitting device:

- 1 START bit
- 5, 6, 7, or 8 data bits
- 1 PARITY bit (optional)
- 1 STOP bit (any other STOP bits transferred with the above data are not detected)

RSR receives the data bits from the UARTn_RXD pin. Then RSR concatenates the data bits and moves the resulting value into RBR (or the receiver FIFO). The UART also stores three bits of error status information next to each received character, to record a parity error, framing error, or break.

In the non-FIFO mode, when a character is placed in RBR and the receiver data-ready interrupt is enabled in the interrupt enable register (IER), an interrupt is generated. This interrupt is cleared when the character is read from RBR. In the FIFO mode, the interrupt is generated when the FIFO is filled to the trigger level selected in the FIFO control register (FCR), and it is cleared when the FIFO contents drop below the trigger level.

Copyright © 2013–2016, Texas Instruments Incorporated

29.2.5.3 FIFO Modes

The following two modes can be used for servicing the receiver and transmitter FIFOs:

- FIFO interrupt mode. The FIFO is enabled and the associated interrupts are enabled. Interrupts are sent to the CPU to indicate when specific events occur.
- FIFO poll mode. The FIFO is enabled but the associated interrupts are disabled. The CPU polls status bits to detect specific events.

Because the receiver FIFO and the transmitter FIFO are controlled separately, either one or both can be placed into the interrupt mode or the poll mode.

29.2.5.3.1 FIFO Interrupt Mode

When the receiver FIFO is enabled in the FIFO control register (FCR) and the receiver interrupts are enabled in the interrupt enable register (IER), the interrupt mode is selected for the receiver FIFO. The following are important points about the receiver interrupts:

- The receiver data-ready interrupt is issued to the CPU when the FIFO has reached the trigger level that is programmed in FCR. It is cleared when the CPU or the DMA controller reads enough characters from the FIFO such that the FIFO drops below its programmed trigger level.
- The receiver line status interrupt is generated in response to an overrun error, a parity error, a framing error, or a break. This interrupt has higher priority than the receiver data-ready interrupt. For details, see Section 29.2.8.
- The data-ready (DR) bit in the line status register (LSR) indicates the presence or absence of characters in the receiver FIFO. The DR bit is set when a character is transferred from the receiver shift register (RSR) to the empty receiver FIFO. The DR bit remains set until the FIFO is empty again.
- A receiver time-out interrupt occurs if all of the following conditions exist:
  - At least one character is in the FIFO,
  - The most recent character was received more than four continuous character times ago. A character time is the time allotted for 1 START bit, *n* data bits, 1 PARITY bit, and 1 STOP bit, where *n* depends on the word length selected with the WLS bits in the line control register (LCR). See Table 29-4.
  - The most recent read of the FIFO has occurred more than four continuous character times before.
- Character times are calculated by using the baud rate.
- When a receiver time-out interrupt has occurred, it is cleared and the time-out timer is cleared when the CPU or the EDMA controller reads one character from the receiver FIFO. The interrupt is also cleared if a new character is received in the FIFO or if the URRST bit is cleared in the power and emulation management register (PWREMU_MGMT).
- If a receiver time-out interrupt has not occurred, the time-out timer is cleared after a new character is received or after the CPU or EDMA reads the receiver FIFO.

When the transmitter FIFO is enabled in FCR and the transmitter holding register empty (THRE) interrupt is enabled in IER, the interrupt mode is selected for the transmitter FIFO. The THRE interrupt occurs when the transmitter FIFO is empty. It is cleared when the transmitter hold register (THR) is loaded (1 to 16 characters may be written to the transmitter FIFO while servicing this interrupt) or the interrupt identification register (IIR) is read.

Table 29-4. Character Time for Word Lengths

Word Length (*n*)	Character Time	Four Character Times
5	Time for 8 bits	Time for 32 bits
6	Time for 9 bits	Time for 36 bits
7	Time for 10 bits	Time for 40 bits
8	Time for 11 bits	Time for 44 bits

Copyright © 2013–2016, Texas Instruments Incorporated

29.2.5.3.2 FIFO Poll Mode

When the receiver FIFO is enabled in the FIFO control register (FCR) and the receiver interrupts are disabled in the interrupt enable register (IER), the poll mode is selected for the receiver FIFO. Similarly, when the transmitter FIFO is enabled and the transmitter interrupts are disabled, the transmitted FIFO is in the poll mode. In the poll mode, the CPU detects events by checking bits in the line status register (LSR):

- The RXFIFOE bit indicates whether there are any errors in the receiver FIFO.
- The TEMT bit indicates that both the transmitter holding register (THR) and the transmitter shift register (TSR) are empty.
- The THRE bit indicates when THR is empty.
- The BI (break), FE (framing error), PE (parity error), and OE (overrun error) bits specify which error or errors have occurred.
- The DR (data-ready) bit is set as long as there is at least one byte in the receiver FIFO.

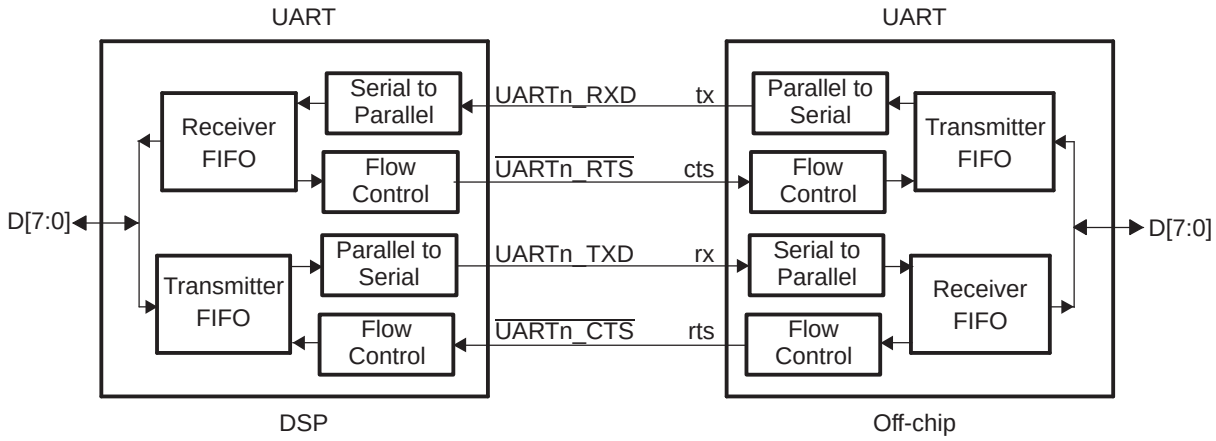
Also, in the FIFO poll mode:

- The interrupt identification register (IIR) is not affected by any events because the interrupts are disabled.
- The UART does not indicate when the receiver FIFO trigger level is reached or when a receiver time-out occurs.

29.2.5.4 Autoflow Control

The UART can employ autoflow control by connecting the $\overline{\text{UARTn_CTS}}$ and $\overline{\text{UARTn_RTS}}$ signals. Note that all UARTs do not support autoflow control, see your device-specific data manual for supported features. The $\overline{\text{UARTn_CTS}}$ input must be active before the transmitter FIFO can transmit data. The $\overline{\text{UARTn_RTS}}$ becomes active when the receiver needs more data and notifies the sending device. When $\overline{\text{UARTn_RTS}}$ is connected to $\overline{\text{UARTn_CTS}}$, data transmission does not occur unless the receiver FIFO has space for the data. Therefore, when two UARTs are connected as shown in Figure 29-5 with autoflow enabled, overrun errors are eliminated.

Figure 29-5. UART Interface Using Autoflow Diagram

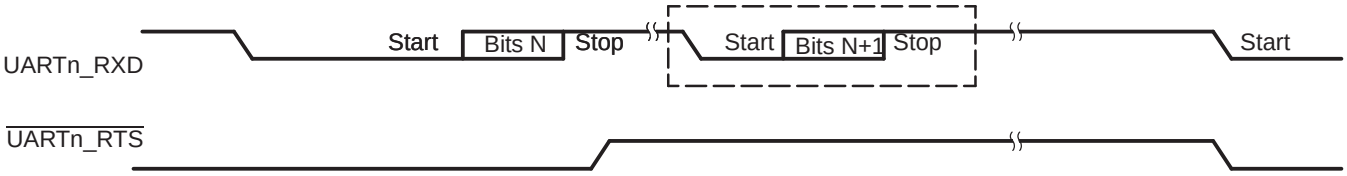


Copyright © 2013–2016, Texas Instruments Incorporated

29.2.5.4.1 UARTn_RTS Behavior

UARTn_RTS data flow control originates in the receiver block (see Figure 29-1). When the receiver FIFO level reaches a trigger level of 1, 4, 8, or 14 (see Figure 29-6), UARTn_RTS is deasserted. The sending UART may send an additional byte after the trigger level is reached (assuming the sending UART has another byte to send), because it may not recognize the deassertion of UARTn_RTS until after it has begun sending the additional byte. For trigger level 1, 4, and 8, UARTn_RTS is automatically reasserted once the receiver FIFO is emptied. For trigger level 14, UARTn_RTS is automatically reasserted once the receiver FIFO drops below the trigger level.

Figure 29-6. Autoflow Functional Timing Waveforms for UARTn_RTS



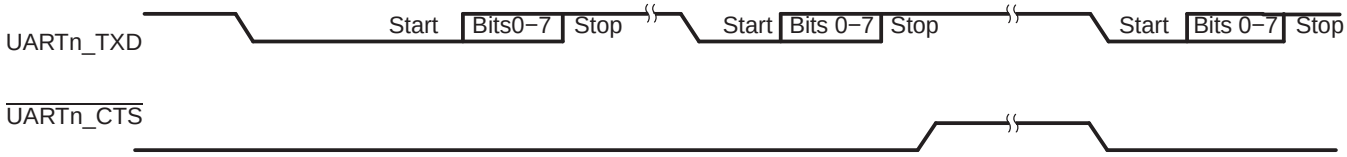
(1) N = Receiver FIFO trigger level.

(2) The two blocks in dashed lines cover the case where an additional byte is sent.

29.2.5.4.2 UARTn_CTS Behavior

The transmitter checks UARTn_CTS before sending the next data byte. If UARTn_CTS is active, the transmitter sends the next byte. To stop the transmitter from sending the following byte, UARTn_CTS must be released before the middle of the last STOP bit that is currently being sent (see Figure 29-7). When flow control is enabled, UARTn_CTS level changes do not trigger interrupts because the device automatically controls its own transmitter. Without autoflow control, the transmitter sends any data present in the transmitter FIFO and a receiver overrun error may result.

Figure 29-7. Autoflow Functional Timing Waveforms for UARTn_CTS



(1) When UARTn_CTS is active (low), the transmitter keeps sending serial data out.

(2) When UARTn_CTS goes high before the middle of the last STOP bit of the current byte, the transmitter finishes sending the current byte but it does not send the next byte.

(3) When UARTn_CTS goes from high to low, the transmitter begins sending data again.

29.2.5.5 Loopback Control

The UART can be placed in the diagnostic mode using the LOOP bit in the modem control register (MCR), which internally connects the UART output back to the UART input. In this mode, the transmit and receive data paths, the transmitter and receiver interrupts, and the modem control interrupts can be verified without connecting to another UART.

Copyright © 2013–2016, Texas Instruments Incorporated

29.2.6 Reset Considerations

29.2.6.1 Software Reset Considerations

Two bits in the power and emulation management register (PWREMU_MGMT) control resetting the parts of the UART:

- The UTRST bit controls resetting the transmitter only. If UTRST = 1, the transmitter is active; if UTRST = 0, the transmitter is in reset.
- The URRST bit controls resetting the receiver only. If URRST = 1, the receiver is active; if URRST = 0, the receiver is in reset.

In each case, putting the receiver and/or transmitter in reset will reset the state machine of the affected portion but does not affect the UART registers.

29.2.6.2 Hardware Reset Considerations

When the processor RESET pin is asserted, the entire processor is reset and is held in the reset state until the RESET pin is released. As part of a device reset, the UART state machine is reset and the UART registers are forced to their default states. The default states of the registers are shown in Section 29.3.

29.2.7 Initialization

The following steps are required to initialize the UART:

1. Perform the necessary device pin multiplexing setup (see your device-specific data manual).
2. Set the desired baud rate by writing the appropriate clock divisor values to the divisor latch registers (DLL and DLH).
3. If the FIFOs will be used, select the desired trigger level and enable the FIFOs by writing the appropriate values to the FIFO control register (FCR). The FIFOEN bit in FCR must be set first, before the other bits in FCR are configured.
4. Choose the desired protocol settings by writing the appropriate values to the line control register (LCR).
5. If autoflow control is desired, write appropriate values to the modem control register (MCR). Note that all UARTs do not support autoflow control, see your device-specific data manual for supported features.
6. Choose the desired response to emulation suspend events by configuring the FREE bit and enable the UART by setting the UTRST and URRST bits in the power and emulation management register (PWREMU_MGMT).

29.2.8 Interrupt Support

29.2.8.1 Interrupt Events and Requests

The UART generates the interrupt requests described in Table 29-5. All requests are multiplexed through an arbiter to a single UART interrupt request to the CPU, as shown in Figure 29-8. Each of the interrupt requests has an enable bit in the interrupt enable register (IER) and is recorded in the interrupt identification register (IIR).

If an interrupt occurs and the corresponding enable bit is set to 1, the interrupt request is recorded in IIR and is forwarded to the CPU. If an interrupt occurs and the corresponding enable bit is cleared to 0, the interrupt request is blocked. The interrupt request is neither recorded in IIR nor forwarded to the CPU.

29.2.8.2 Interrupt Multiplexing

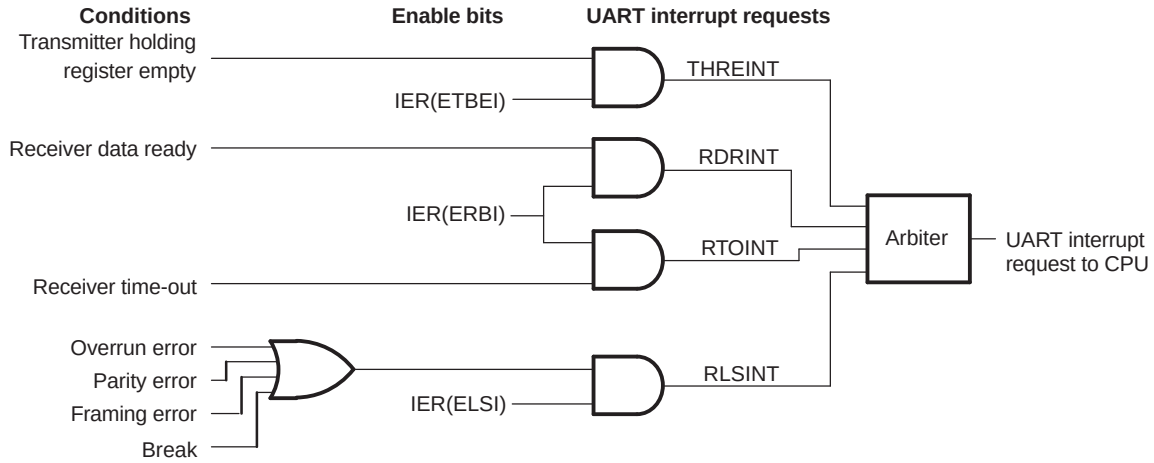
The UARTs have dedicated interrupt signals to the CPU and the interrupts are not multiplexed with any other interrupt source.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 29-5. UART Interrupt Requests Descriptions

UART Interrupt Request	Interrupt Source	Comment
THREINT	THR-empty condition: The transmitter holding register (THR) or the transmitter FIFO is empty. All of the data has been copied from THR to the transmitter shift register (TSR).	If THREINT is enabled in IER, by setting the ETBEI bit, it is recorded in IIR. As an alternative to using THREINT, the CPU can poll the THRE bit in the line status register (LSR).
RDAINT	Receive data available in non-FIFO mode or trigger level reached in the FIFO mode.	If RDAINT is enabled in IER, by setting the ERBI bit, it is recorded in IIR. As an alternative to using RDAINT, the CPU can poll the DR bit in the line status register (LSR). In the FIFO mode, this is not a functionally equivalent alternative because the DR bit does not respond to the FIFO trigger level. The DR bit only indicates the presence or absence of unread characters.
RTOINT	Receiver time-out condition (in the FIFO mode only): No characters have been removed from or input to the receiver FIFO during the last four character times (see Table 29-4), and there is at least one character in the receiver FIFO during this time.	The receiver time-out interrupt prevents the UART from waiting indefinitely, in the case when the receiver FIFO level is below the trigger level and thus does not generate a receiver data-ready interrupt. If RTOINT is enabled in IER, by setting the ERBI bit, it is recorded in IIR. There is no status bit to reflect the occurrence of a time-out condition.
RLSINT	Receiver line status condition: An overrun error, parity error, framing error, or break has occurred.	If RLSINT is enabled in IER, by setting the ELSI bit, it is recorded in IIR. As an alternative to using RLSINT, the CPU can poll the following bits in the line status register (LSR): overrun error indicator (OE), parity error indicator (PE), framing error indicator (FE), and break indicator (BI).

Figure 29-8. UART Interrupt Request Enable Paths



Copyright © 2013–2016, Texas Instruments Incorporated

29.2.9 DMA Event Support

In the FIFO mode, the UART generates the following two DMA events:

- **Receive event (URXEVT):** The trigger level for the receiver FIFO (1, 4, 8, or 14 characters) is set with the RXFIFTL bit in the FIFO control register (FCR). Every time the trigger level is reached or a receiver time-out occurs, the UART sends a receive event to the EDMA controller. In response, the EDMA controller reads the data from the receiver FIFO by way of the receiver buffer register (RBR). Note that the receive event is not asserted if the data at the top of the receiver FIFO is erroneous even if the trigger level has been reached.
- **Transmit event (UTXEVT):** When the transmitter FIFO is empty (when the last byte in the transmitter FIFO has been copied to the transmitter shift register), the UART sends an UTXEVT signal to the EDMA controller. In response, the EDMA controller refills the transmitter FIFO by way of the transmitter holding register (THR). The UTXEVT signal is also sent to the DMA controller when the UART is taken out of reset using the UTRST bit in the power and emulation management register (PWREMU_MGMT).

Activity in DMA channels can be synchronized to these events. In the non-FIFO mode, the UART generates no DMA events. Any DMA channel synchronized to either of these events must be enabled at the time the UART event is generated. Otherwise, the DMA channel will miss the event and, unless the UART generates a new event, no data transfer will occur.

29.2.10 Power Management

The UART peripheral can be placed in reduced-power modes to conserve power during periods of low activity. The power management of the UART peripheral is controlled by the processor Power and Sleep Controller (PSC). The PSC acts as a master controller for power management for all of the peripherals on the device. For detailed information on power management procedures using the PSC, see the *Power and Sleep Controller (PSC)* chapter.

29.2.11 Emulation Considerations

The FREE bit in the power and emulation management register (PWREMU_MGMT) determines how the UART responds to an emulation suspend event such as an emulator halt or breakpoint. If FREE = 0 and a transmission is in progress, the UART halts after completing the one-word transmission; if FREE = 0 and a transmission is not in progress, the UART halts immediately. If FREE = 1, the UART does not halt and continues operating normally.

Note also that most emulator accesses are transparent to UART operation. Emulator read operations do not affect any register contents, status bits, or operating states, with the exception of the interrupt identification register (IIR). Emulator writes, however, may affect register contents and may affect UART operation, depending on what register is accessed and what value is written.

The UART registers can be read from or written to during emulation suspend events, even if the UART activity has stopped.

29.2.12 Exception Processing

29.2.12.1 Divisor Latch Not Programmed

Since the processor reset signal has no effect on the divisor latch, the divisor latch will have an unknown value after power up. If the divisor latch is not programmed after power up, the baud clock (BCLK) will not operate and will instead be set to a constant logic 1 state.

The divisor latch values should always be reinitialized following a processor reset.

29.2.12.2 Changing Operating Mode During Busy Serial Communication

Since the serial link characteristics are based on how the control registers are programmed, the UART will expect the control registers to be static while it is busy engaging in a serial communication. Therefore, changing the control registers while the module is still busy communicating with another serial device will most likely cause an error condition and should be avoided.

Copyright © 2013–2016, Texas Instruments Incorporated

29.3 Registers

The system programmer has access to and control over any of the UART registers that are listed in Table 29-6. These registers, which control UART operations, receive data, and transmit data, are available at 32-bit addresses in the device memory map. See your device-specific data manual for the memory address of these registers.

- RBR, THR, and DLL share one address. When the DLAB bit in LCR is 0, reading from the address gives the content of RBR, and writing to the address modifies THR. When DLAB = 1, all accesses at the address read or modify DLL. DLL can also be accessed with address offset 20h.
- IER and DLH share one address. When DLAB = 0, all accesses read or modify IER. When DLAB = 1, all accesses read or modify DLH. DLH can also be accessed with address offset 24h.
- IIR and FCR share one address. Regardless of the value of the DLAB bit, reading from the address gives the content of IIR, and writing modifies FCR.

Table 29-6. UART Registers

Offset	Acronym	Register Description	Section
0h	RBR	Receiver Buffer Register (read only)	Section 29.3.1
0h	THR	Transmitter Holding Register (write only)	Section 29.3.2
4h	IER	Interrupt Enable Register	Section 29.3.3
8h	IIR	Interrupt Identification Register (read only)	Section 29.3.4
8h	FCR	FIFO Control Register (write only)	Section 29.3.5
Ch	LCR	Line Control Register	Section 29.3.6
10h	MCR	Modem Control Register	Section 29.3.7
14h	LSR	Line Status Register	Section 29.3.8
18h	MSR	Modem Status Register	Section 29.3.9
1Ch	SCR	Scratch Pad Register	Section 29.3.10
20h	DLL	Divisor LSB Latch	Section 29.3.11
24h	DLH	Divisor MSB Latch	Section 29.3.11
28h	REVID1	Revision Identification Register 1	Section 29.3.12
2Ch	REVID2	Revision Identification Register 2	Section 29.3.12
30h	PWREMU_MGMT	Power and Emulation Management Register	Section 29.3.13
34h	MDR	Mode Definition Register	Section 29.3.14

Copyright © 2013–2016, Texas Instruments Incorporated

29.3.1 Receiver Buffer Register (RBR)

The receiver buffer register (RBR) is shown in Figure 29-9 and described in Table 29-7.

The UART receiver section consists of a receiver shift register (RSR) and a receiver buffer register (RBR). When the UART is in the FIFO mode, RBR is a 16-byte FIFO. Timing is supplied by the 16x receiver clock or 13x receiver clock by programming OSM_SEL bit field of MDR register. Receiver section control is a function of the line control register (LCR).

RSR receives serial data from the UARTn_RXD pin. Then RSR concatenates the data and moves it into RBR (or the receiver FIFO). In the non-FIFO mode, when a character is placed in RBR and the receiver data-ready interrupt is enabled (DR = 1 in IER), an interrupt is generated. This interrupt is cleared when the character is read from RBR. In the FIFO mode, the interrupt is generated when the FIFO is filled to the trigger level selected in the FIFO control register (FCR), and it is cleared when the FIFO contents drop below the trigger level.

Access considerations:

RBR, THR, and DLL share one address. To read RBR, write 0 to the DLAB bit in LCR, and read from the shared address. When DLAB = 0, writing to the shared address modifies THR. When DLAB = 1, all accesses at the shared address read or modify DLL.

DLL also has a dedicated address. If you use the dedicated address, you can keep DLAB = 0, so that RBR and THR are always selected at the shared address.

Figure 29-9. Receiver Buffer Register (RBR)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		DATA	
R-0		R-0	

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

Table 29-7. Receiver Buffer Register (RBR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	DATA	0-FFh	Received data

Copyright © 2013–2016, Texas Instruments Incorporated

29.3.2 Transmitter Holding Register (THR)

The transmitter holding register (THR) is shown in Figure 29-10 and described in Table 29-8.

The UART transmitter section consists of a transmitter hold register (THR) and a transmitter shift register (TSR). When the UART is in the FIFO mode, THR is a 16-byte FIFO. Transmitter section control is a function of the line control register (LCR).

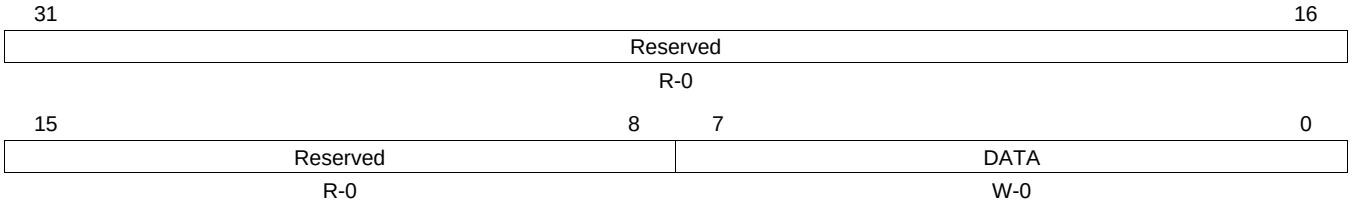
THR receives data from the internal data bus and when TSR is idle, the UART moves the data from THR to TSR. The UART serializes the data in TSR and transmits the data on the TX pin. In the non-FIFO mode, if THR is empty and the THR empty (THRE) interrupt is enabled (ETBEI = 1 in IER), an interrupt is generated. This interrupt is cleared when a character is loaded into THR or the interrupt identification register (IIR) is read. In the FIFO mode, the interrupt is generated when the transmitter FIFO is empty, and it is cleared when at least one byte is loaded into the FIFO or IIR is read.

Access considerations:

RBR, THR, and DLL share one address. To load THR, write 0 to the DLAB bit of LCR, and write to the shared address. When DLAB = 0, reading from the shared address gives the content of RBR. When DLAB = 1, all accesses at the address read or modify DLL.

DLL also has a dedicated address. If you use the dedicated address, you can keep DLAB = 0, so that RBR and THR are always selected at the shared address.

Figure 29-10. Transmitter Holding Register (THR)



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

Table 29-8. Transmitter Holding Register (THR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	DATA	0-FFh	Data to transmit

Copyright © 2013–2016, Texas Instruments Incorporated

29.3.3 Interrupt Enable Register (IER)

The interrupt enable register (IER) is used to individually enable or disable each type of interrupt request that can be generated by the UART. Each interrupt request that is enabled in IER is forwarded to the CPU. IER is shown in Figure 29-11 and described in Table 29-9.

Access considerations:

IER and DLH share one address. To read or modify IER, write 0 to the DLAB bit in LCR. When DLAB = 1, all accesses at the shared address read or modify DLH.

DLH also has a dedicated address. If you use the dedicated address, you can keep DLAB = 0, so that IER is always selected at the shared address.

Figure 29-11. Interrupt Enable Register (IER)

31	16
Reserved	
R-0	

15	4	3	2	1	0
Reserved		Rsvd	ELSI	ETBEI	ERBI
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 29-9. Interrupt Enable Register (IER) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	EDSSI	0	Enable Modem Status Interrupt
2	ELSI		Receiver line status interrupt enable.
		0	Receiver line status interrupt is disabled.
		1	Receiver line status interrupt is enabled.
1	ETBEI		Transmitter holding register empty interrupt enable.
		0	Transmitter holding register empty interrupt is disabled.
		1	Transmitter holding register empty interrupt is enabled.
0	ERBI		Receiver data available interrupt and character timeout indication interrupt enable.
		0	Receiver data available interrupt and character timeout indication interrupt is disabled.
		1	Receiver data available interrupt and character timeout indication interrupt is enabled.

Copyright © 2013–2016, Texas Instruments Incorporated

29.3.4 Interrupt Identification Register (IIR)

The interrupt identification register (IIR) is a read-only register at the same address as the FIFO control register (FCR), which is a write-only register. When an interrupt is generated and enabled in the interrupt enable register (IER), IIR indicates that an interrupt is pending in the IPEND bit and encodes the type of interrupt in the INTID bits. Reading IIR clears any THR empty (THRE) interrupts that are pending.

IIR is shown in Figure 29-12 and described in Figure 29-12.

The UART has an on-chip interrupt generation and prioritization capability that permits flexible communication with the CPU. The UART provides three priority levels of interrupts:

- Priority 1 - Receiver line status (highest priority)
- Priority 2 - Receiver data ready or receiver timeout
- Priority 3 - Transmitter holding register empty

The FIFOEN bit in IIR can be checked to determine whether the UART is in the FIFO mode or the non-FIFO mode.

Access consideration:

IIR and FCR share one address. Regardless of the value of the DLAB bit in LCR, reading from the address gives the content of IIR, and writing to the address modifies FCR.

Figure 29-12. Interrupt Identification Register (IIR)

31	16
Reserved	
R-0	

15	8	7	6	5	4	3	1	0
Reserved		FIFOEN		Reserved		INTID		IPEND
R-0		R-0		R-0		R-0		R-1

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

Table 29-10. Interrupt Identification Register (IIR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-6	FIFOEN	0-3h	FIFOs enabled.
		0	Non-FIFO mode
		1h-2h	Reserved
		3h	FIFOs are enabled. FIFOEN bit in the FIFO control register (FCR) is set to 1.
5-4	Reserved	0	Reserved
3-1	INTID	0-7h	Interrupt type. See Table 29-11.
		0	Reserved
		1h	Transmitter holding register empty (priority 3)
		2h	Receiver data available (priority 2)
		3h	Receiver line status (priority 1, highest)
		4h-5h	Reserved
		6h	Character timeout indication (priority 2)
		7h	Reserved
0	IPEND		Interrupt pending. When any UART interrupt is generated and is enabled in IER, IPEND is forced to 0. IPEND remains 0 until all pending interrupts are cleared or until a hardware reset occurs. If no interrupts are enabled, IPEND is never forced to 0.
		0	Interrupts pending.
		1	No interrupts pending.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 29-11. Interrupt Identification and Interrupt Clearing Information

Priority Level	IIR Bits 3	2	1	0	Interrupt Type	Interrupt Source	Event That Clears Interrupt
None	0	0	0	1	None	None	None
1	0	1	1	0	Receiver line status	Overrun error, parity error, framing error, or break is detected.	For an overrun error, reading the line status register (LSR) clears the interrupt. For a parity error, framing error, or break, the interrupt is cleared only after all the erroneous data have been read.
2	0	1	0	0	Receiver data-ready	Non-FIFO mode: Receiver data is ready.	Non-FIFO mode: The receiver buffer register (RBR) is read.
						FIFO mode: Trigger level reached. If four character times (see Table 29-4) pass with no access of the FIFO, the interrupt is asserted again.	FIFO mode: The FIFO drops below the trigger level.(1)
2	1	1	0	0	Receiver time-out	FIFO mode only: No characters have been removed from or input to the receiver FIFO during the last four character times (see Table 29-4), and there is at least one character in the receiver FIFO during this time.	One of the following events: • A character is read from the receiver FIFO.(1) • A new character arrives in the receiver FIFO. • The URRST bit in the power and emulation management register (PWREMU_MGMT) is loaded with 0.
3	0	0	1	0	Transmitter holding register empty	Non-FIFO mode: Transmitter holding register (THR) is empty. FIFO mode: Transmitter FIFO is empty.	A character is written to the transmitter holding register (THR) or the interrupt identification register (IIR) is read.

(1) In the FIFO mode, the receiver data-ready interrupt or receiver time-out interrupt is cleared by the CPU or by the DMA controller, whichever reads from the receiver FIFO first.

29.3.5 FIFO Control Register (FCR)

The FIFO control register (FCR) is a write-only register at the same address as the interrupt identification register (IIR), which is a read-only register. Use FCR to enable and clear the FIFOs and to select the receiver FIFO trigger level FCR is shown in Figure 29-13 and described in Table 29-12. The FIFOEN bit must be set to 1 before other FCR bits are written to or the FCR bits are not programmed.

Access consideration:

IIR and FCR share one address. Regardless of the value of the DLAB bit, reading from the address gives the content of IIR, and writing to the address modifies FCR.

CAUTION

For proper communication between the UART and the EDMA controller, the DMAMODE1 bit must be set to 1. Always write a 1 to the DMAMODE1 bit, and after a hardware reset, change the DMAMODE1 bit from 0 to 1.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 29-13. FIFO Control Register (FCR)

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
RXFIFTL		Reserved		DMAMODE1[(1)]	TXCLR	RXCLR	FIFOEN
W-0		R-0		W-0	W1C-0	W1C-0	W-0

LEGEND: R = Read only; W = Write only; W1C = Write 1 to clear (writing 0 has no effect); -*n* = value after reset

(1) Always write 1 to the DMAMODE1 bit. After a hardware reset, change the DMAMODE1 bit from 0 to 1. DMAMODE = 1 is required for proper communication between the UART and the DMA controller.

Table 29-12. FIFO Control Register (FCR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-6	RXFIFTL	0-3h	Receiver FIFO trigger level. RXFIFTL sets the trigger level for the receiver FIFO. When the trigger level is reached, a receiver data-ready interrupt is generated (if the interrupt request is enabled). Once the FIFO drops below the trigger level, the interrupt is cleared.
		0	1 byte
		1h	4 bytes
		2h	8 bytes
		3h	14 bytes
5-4	Reserved	0	Reserved
3	DMAMODE1		DMA MODE1 enable if FIFOs are enabled. Always write 1 to DMAMODE1. After a hardware reset, change DMAMODE1 from 0 to 1. DMAMOD1 = 1 is a requirement for proper communication between the UART and the EDMA controller.
		0	DMA MODE1 is disabled.
		1	DMA MODE1 is enabled.
2	TXCLR		Transmitter FIFO clear. Write a 1 to TXCLR to clear the bit.
		0	No effect.
		1	Clears transmitter FIFO and resets the transmitter FIFO counter. The shift register is not cleared.
1	RXCLR		Receiver FIFO clear. Write a 1 to RXCLR to clear the bit.
		0	No effect.
		1	Clears receiver FIFO and resets the receiver FIFO counter. The shift register is not cleared.
0	FIFOEN		Transmitter and receiver FIFOs mode enable. FIFOEN must be set before other FCR bits are written to or the FCR bits are not programmed. Clearing this bit clears the FIFO counters.
		0	Non-FIFO mode. The transmitter and receiver FIFOs are disabled, and the FIFO pointers are cleared.
		1	FIFO mode. The transmitter and receiver FIFOs are enabled.

Copyright © 2013–2016, Texas Instruments Incorporated

29.3.6 Line Control Register (LCR)

The line control register (LCR) is shown in Figure 29-14 and described in Table 29-13.

The system programmer controls the format of the asynchronous data communication exchange by using LCR. In addition, the programmer can retrieve, inspect, and modify the content of LCR; this eliminates the need for separate storage of the line characteristics in system memory.

Figure 29-14. Line Control Register (LCR)

31								16
Reserved								
R-0								
15	8	7	6	5	4	3	2	1 0
Reserved		DLAB	BC	SP	EPS	PEN	STB	WLS
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 29-13. Line Control Register (LCR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	DLAB		Divisor latch access bit. The divisor latch registers (DLL and DLH) can be accessed at dedicated addresses or at addresses shared by RBR, THR, and IER. Using the shared addresses requires toggling DLAB to change which registers are selected. If you use the dedicated addresses, you can keep DLAB = 0.
		0	Allows access to the receiver buffer register (RBR), the transmitter holding register (THR), and the interrupt enable register (IER) selected. At the address shared by RBR, THR, and DLL, the CPU can read from RBR and write to THR. At the address shared by IER and DLH, the CPU can read from and write to IER.
		1	Allows access to the divisor latches of the baud generator during a read or write operation (DLL and DLH). At the address shared by RBR, THR, and DLL, the CPU can read from and write to DLL. At the address shared by IER and DLH, the CPU can read from and write to DLH.
6	BC		Break control.
		0	Break condition is disabled.
		1	Break condition is transmitted to the receiving UART. A break condition is a condition where the UARTn_TXD signal is forced to the spacing (cleared) state.
5	SP		Stick parity. The SP bit works in conjunction with the EPS and PEN bits. The relationship between the SP, EPS, and PEN bits is summarized in Table 29-14.
		0	Stick parity is disabled.
		1	Stick parity is enabled. • When odd parity is selected (EPS = 0), the PARITY bit is transmitted and checked as set. • When even parity is selected (EPS = 1), the PARITY bit is transmitted and checked as cleared.
4	EPS		Even parity select. Selects the parity when parity is enabled (PEN = 1). The EPS bit works in conjunction with the SP and PEN bits. The relationship between the SP, EPS, and PEN bits is summarized in Table 29-14.
		0	Odd parity is selected (an odd number of logic 1s is transmitted or checked in the data and PARITY bits).
		1	Even parity is selected (an even number of logic 1s is transmitted or checked in the data and PARITY bits).
3	PEN		Parity enable. The PEN bit works in conjunction with the SP and EPS bits. The relationship between the SP, EPS, and PEN bits is summarized in Table 29-14.
		0	No PARITY bit is transmitted or checked.
		1	Parity bit is generated in transmitted data and is checked in received data between the last data word bit and the first STOP bit.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 29-13. Line Control Register (LCR) Field Descriptions (continued)

Bit	Field	Value	Description
2	STB		Number of STOP bits generated. STB specifies 1, 1.5, or 2 STOP bits in each transmitted character. When STB = 1, the WLS bit determines the number of STOP bits. The receiver clocks only the first STOP bit, regardless of the number of STOP bits selected. The number of STOP bits generated is summarized in Table 29-15.
		0	1 STOP bit is generated.
		1	WLS bit determines the number of STOP bits: • When WLS = 0, 1.5 STOP bits are generated. • When WLS = 1h, 2h, or 3h, 2 STOP bits are generated.
1-0	WLS	0-3h	Word length select. Number of bits in each transmitted or received serial character. When STB = 1, the WLS bit determines the number of STOP bits.
		0	5 bits
		1h	6 bits
		2h	7 bits
		3h	8 bits

Table 29-14. Relationship Between ST, EPS, and PEN Bits in LCR

ST Bit	EPS Bit	PEN Bit	Parity Option
x	x	0	Parity disabled: No PARITY bit is transmitted or checked
0	0	1	Odd parity selected: Odd number of logic 1s
0	1	1	Even parity selected: Even number of logic 1s
1	0	1	Stick parity selected with PARITY bit transmitted and checked as set
1	1	1	Stick parity selected with PARITY bit transmitted and checked as cleared

Table 29-15. Number of STOP Bits Generated

STB Bit	WLS Bits	Word Length Selected with WLS Bits	Number of STOP Bits Generated	Baud Clock (BCLK) Cycles
0	x	Any word length	1	16
1	0h	5 bits	1.5	24
1	1h	6 bits	2	32
1	2h	7 bits	2	32
1	3h	8 bits	2	32

Copyright © 2013–2016, Texas Instruments Incorporated

29.3.7 Modem Control Register (MCR)

The modem control register (MCR) is shown in Figure 29-15 and described in Table 29-16. The modem control register provides the ability to enable/disable the autoflow functions, and enable/disable the loopback function for diagnostic purposes.

Figure 29-15. Modem Control Register (MCR)

31							16
Reserved							
R-0							
15		**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved			AFE[1]	LOOP	OUT2	OUT1	RTS[1]	Rsvd
R-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

(1) All UARTs do not support this feature, see your device-specific data manual for supported features. If this feature is not available, this bit is reserved and should be cleared to 0.

Table 29-16. Modem Control Register (MCR) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reserved
5	AFE		Autoflow control enable. Autoflow control allows the $\overline{\text{UARTn_RTS}}$ and $\overline{\text{UARTn_CTS}}$ signals to provide handshaking between UARTs during data transfer. When AFE = 1, the RTS bit determines the autoflow control enabled. Note that all UARTs do not support this feature, see your device-specific data manual for supported features. If this feature is not available, this bit is reserved and should be cleared to 0.
		0	Autoflow control is disabled.
		1	Autoflow control is enabled: • When RTS = 0, $\overline{\text{UARTn_CTS}}$ is only enabled. • When RTS = 1, $\overline{\text{UARTn_RTS}}$ and $\overline{\text{UARTn_CTS}}$ are enabled.
4	LOOP		Loop back mode enable. LOOP is used for the diagnostic testing using the loop back feature.
		0	Loop back mode is disabled.
		1	Loop back mode is enabled. When LOOP is set, the following occur: • The UARTn_TXD signal is set high. • The UARTn_RXD pin is disconnected • The output of the transmitter shift register (TSR) is lopped back in to the receiver shift register (RSR) input.
3	OUT2	0	OUT2 Control Bit
2	OUT1	0	OUT1 Control Bit
1	RTS		RTS control. When AFE = 1, the RTS bit determines the autoflow control enabled. Note that all UARTs do not support this feature, see your device-specific data manual for supported features. If this feature is not available, this bit is reserved and should be cleared to 0.
		0	$\overline{\text{UARTn_RTS}}$ is disabled, $\overline{\text{UARTn_CTS}}$ is only enabled.
		1	$\overline{\text{UARTn_RTS}}$ and $\overline{\text{UARTn_CTS}}$ are enabled.
0	Reserved	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

29.3.8 Line Status Register (LSR)

The line status register (LSR) is shown in Figure 29-16 and described in Table 29-17. LSR provides information to the CPU concerning the status of data transfers. LSR is intended for read operations only; do not write to this register. Bits 1 through 4 record the error conditions that produce a receiver line status interrupt.

Figure 29-16. Line Status Register (LSR)

31									16
Reserved									
R-0									
15	8	7	6	5	4	3	2	1	0
Reserved		RXFIFOE	TEMT	THRE	BI	FE	PE	OE	DR
R-0		R-0	R-1	R-1	R-0	R-0	R-0	R-0	R-0

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

Table 29-17. Line Status Register (LSR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	RXFIFOE		Receiver FIFO error.
			In non-FIFO mode:
		0	There has been no error, or RXFIFOE was cleared because the CPU read the erroneous character from the receiver buffer register (RBR).
		1	There is a parity error, framing error, or break indicator in the receiver buffer register (RBR).
			In FIFO mode:
		0	There has been no error, or RXFIFOE was cleared because the CPU read the erroneous character from the receiver FIFO and there are no more errors in the receiver FIFO.
		1	At least one parity error, framing error, or break indicator in the receiver FIFO.
6	TEMT		Transmitter empty (TEMT) indicator.
			In non-FIFO mode:
		0	Either the transmitter holding register (THR) or the transmitter shift register (TSR) contains a data character.
		1	Both the transmitter holding register (THR) and the transmitter shift register (TSR) are empty.
			In FIFO mode:
		0	Either the transmitter FIFO or the transmitter shift register (TSR) contains a data character.
		1	Both the transmitter FIFO and the transmitter shift register (TSR) are empty.
5	THRE		Transmitter holding register empty (THRE) indicator. If the THRE bit is set and the corresponding interrupt enable bit is set (ETBEI = 1 in IER), an interrupt request is generated.
			In non-FIFO mode:
		0	Transmitter holding register (THR) is not empty. THR has been loaded by the CPU.
		1	Transmitter holding register (THR) is empty (ready to accept a new character). The content of THR has been transferred to the transmitter shift register (TSR).
			In FIFO mode:
		0	Transmitter FIFO is not empty. At least one character has been written to the transmitter FIFO. You can write to the transmitter FIFO if it is not full.
		1	Transmitter FIFO is empty. The last character in the FIFO has been transferred to the transmitter shift register (TSR).

Copyright © 2013–2016, Texas Instruments Incorporated

Table 29-17. Line Status Register (LSR) Field Descriptions (continued)

Bit	Field	Value	Description
4	BI		Break indicator. The BI bit is set whenever the receive data input (UARTn_RXD) was held low for longer than a full-word transmission time. A full-word transmission time is defined as the total time to transmit the START, data, PARITY, and STOP bits. If the BI bit is set and the corresponding interrupt enable bit is set (ELSI = 1 in IER), an interrupt request is generated.
			In non-FIFO mode:
		0	No break has been detected, or the BI bit was cleared because the CPU read the erroneous character from the receiver buffer register (RBR).
		1	A break has been detected with the character in the receiver buffer register (RBR).
			In FIFO mode:
		0	No break has been detected, or the BI bit was cleared because the CPU read the erroneous character from the receiver FIFO and the next character to be read from the FIFO has no break indicator.
		1	A break has been detected with the character at the top of the receiver FIFO.
3	FE		Framing error (FE) indicator. A framing error occurs when the received character does not have a valid STOP bit. In response to a framing error, the UART sets the FE bit and waits until the signal on the RX pin goes high. Once the RX signal goes high, the receiver is ready to detect a new START bit and receive new data. If the FE bit is set and the corresponding interrupt enable bit is set (ELSI = 1 in IER), an interrupt request is generated.
			In non-FIFO mode:
		0	No framing error has been detected, or the FE bit was cleared because the CPU read the erroneous data from the receiver buffer register (RBR).
		1	A framing error has been detected with the character in the receiver buffer register (RBR).
			In FIFO mode:
		0	No framing error has been detected, or the FE bit was cleared because the CPU read the erroneous data from the receiver FIFO and the next character to be read from the FIFO has no framing error.
		1	A framing error has been detected with the character at the top of the receiver FIFO.
2	PE		Parity error (PE) indicator. A parity error occurs when the parity of the received character does not match the parity selected with the EPS bit in the line control register (LCR). If the PE bit is set and the corresponding interrupt enable bit is set (ELSI = 1 in IER), an interrupt request is generated.
			In non-FIFO mode:
		0	No parity error has been detected, or the PE bit was cleared because the CPU read the erroneous data from the receiver buffer register (RBR).
		1	A parity error has been detected with the character in the receiver buffer register (RBR).
			In FIFO mode:
		0	No parity error has been detected, or the PE bit was cleared because the CPU read the erroneous data from the receiver FIFO and the next character to be read from the FIFO has no parity error.
		1	A parity error has been detected with the character at the top of the receiver FIFO.
1	OE		Overrun error (OE) indicator. An overrun error in the non-FIFO mode is different from an overrun error in the FIFO mode. If the OE bit is set and the corresponding interrupt enable bit is set (ELSI = 1 in IER), an interrupt request is generated.
			In non-FIFO mode:
		0	No overrun error has been detected, or the OE bit was cleared because the CPU read the content of the line status register (LSR).
		1	Overrun error has been detected. Before the character in the receiver buffer register (RBR) could be read, it was overwritten by the next character arriving in RBR.
			In FIFO mode:
		0	No overrun error has been detected, or the OE bit was cleared because the CPU read the content of the line status register (LSR).
		1	Overrun error has been detected. If data continues to fill the FIFO beyond the trigger level, an overrun error occurs only after the FIFO is full and the next character has been completely received in the shift register. An overrun error is indicated to the CPU as soon as it happens. The new character overwrites the character in the shift register, but it is not transferred to the FIFO.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 29-17. Line Status Register (LSR) Field Descriptions (continued)

Bit	Field	Value	Description
0	DR		Data-ready (DR) indicator for the receiver. If the DR bit is set and the corresponding interrupt enable bit is set (ERBI = 1 in IER), an interrupt request is generated.
			In non-FIFO mode:
		0	Data is not ready, or the DR bit was cleared because the character was read from the receiver buffer register (RBR).
		1	Data is ready. A complete incoming character has been received and transferred into the receiver buffer register (RBR).
			In FIFO mode:
		0	Data is not ready, or the DR bit was cleared because all of the characters in the receiver FIFO have been read.
		1	Data is ready. There is at least one unread character in the receiver FIFO. If the FIFO is empty, the DR bit is set as soon as a complete incoming character has been received and transferred into the FIFO. The DR bit remains set until the FIFO is empty again.

Copyright © 2013–2016, Texas Instruments Incorporated

29.3.9 Modem Status Register (MSR)

The Modem status register (MSR) is shown in Figure 29-17 and described in Table 29-18. MSR provides information to the CPU concerning the status of modem control signals. MSR is intended for read operations only; do not write to this register.

Figure 29-17. Modem Status Register (MSR)

31								16
Reserved								
R-0								
15	8	7	6	5	4	3	2	1	0
Reserved		CD	RI	DSR	CTS	DCD	TERI	DDSR	DCTS
R-0		R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

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

Table 29-18. Modem Status Register (MSR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	CD	0	Complement of the Carrier Detect input. When the UART is in the diagnostic test mode (loopback mode MCR[4] = 1), this bit is equal to the MCR bit 3 (OUT2).
6	RI	0	Complement of the Ring Indicator input. When the UART is in the diagnostic test mode (loopback mode MCR[4] = 1), this bit is equal to the MCR bit 2 (OUT1).
5	DSR	0	Complement of the Data Set Ready input. When the UART is in the diagnostic test mode (loopback mode MCR[4] = 1), this bit is equal to the MCR bit 0 (DTR).
4	CTS	0	Complement of the Clear To Send input. When the UART is in the diagnostic test mode (loopback mode MCR[4] = 1), this bit is equal to the MCR bit 1 (RTS).
3	DCD	0	Change in DCD indicator bit. DCD indicates that the DCD input has changed state since the last time it was read by the CPU. When DCD is set and the modem status interrupt is enabled, a modem status interrupt is generated.
2	TERI	0	Trailing edge of RI (TERI) indicator bit. TERI indicates that the RI input has changed from a low to a high. When TERI is set and the modem status interrupt is enabled, a modem status interrupt is generated.
1	DDSR	0	Change in DSR indicator bit. DDSR indicates that the DSR input has changed state since the last time it was read by the CPU. When DDSR is set and the modem status interrupt is enabled, a modem status interrupt is generated.
0	DCTS	0	Change in CTS indicator bit. DCTS indicates that the CTS input has changed state since the last time it was read by the CPU. When DCTS is set (autoflow control is not enabled and the modem status interrupt is enabled), a modem status interrupt is generated. When autoflow control is enabled, no interrupt is generated.

Copyright © 2013–2016, Texas Instruments Incorporated

29.3.10 Scratch Pad Register (SCR)

The Scratch Pad register (SCR) is shown in Figure 29-18 and described in Table 29-19. SCR is intended for programmer's use as a scratch pad. It temporarily holds the programmer's data without affecting UART operation.

Figure 29-18. Scratch Pad Register (SCR)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		SCR	
R-0		R-0	

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

Table 29-19. Scratch Pad Register (MSR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	SCR	0	These bits are intended for the programmer's use as a scratch pad in the sense that it temporarily holds the programmer's data without affecting any other UART operation.

29.3.11 Divisor Latches (DLL and DLH)

Two 8-bit register fields (DLL and DLH), called divisor latches, store the 16-bit divisor for generation of the baud clock in the baud generator. The latches are in DLH and DLL. DLH holds the most-significant bits of the divisor, and DLL holds the least-significant bits of the divisor. These divisor latches must be loaded during initialization of the UART in order to ensure desired operation of the baud generator. Writing to the divisor latches results in two wait states being inserted during the write access while the baud generator is loaded with the new value.

Access considerations:

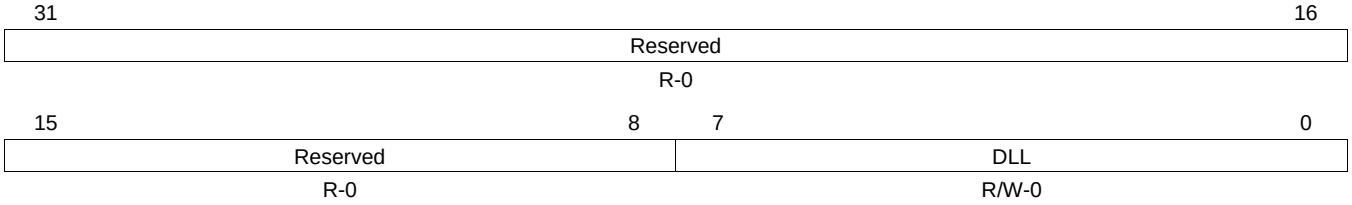
- RBR, THR, and DLL share one address. When DLAB = 1 in LCR, all accesses at the shared address are accesses to DLL. When DLAB = 0, reading from the shared address gives the content of RBR, and writing to the shared address modifies THR.
- IER and DLH share one address. When DLAB = 1 in LCR, accesses to the shared address read or modify to DLH. When DLAB = 0, all accesses at the shared address read or modify IER.

DLL and DLH also have dedicated addresses. If you use the dedicated addresses, you can keep the DLAB bit cleared, so that RBR, THR, and IER are always selected at the shared addresses.

The divisor LSB latch (DLL) is shown in Figure 29-19 and described in Table 29-20. The divisor MSB latch (DLH) is shown in Figure 29-20 and described in Table 29-21.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 29-19. Divisor LSB Latch (DLL)

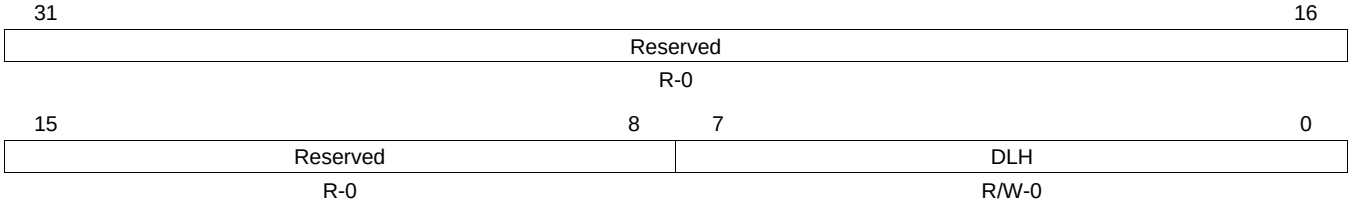


LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 29-20. Divisor LSB Latch (DLL) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	DLL	0-FFh	The 8 least-significant bits (LSBs) of the 16-bit divisor for generation of the baud clock in the baud rate generator.

Figure 29-20. Divisor MSB Latch (DLH)



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

Table 29-21. Divisor MSB Latch (DLH) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	DLH	0-FFh	The 8 most-significant bits (MSBs) of the 16-bit divisor for generation of the baud clock in the baud rate generator.

Copyright © 2013–2016, Texas Instruments Incorporated

29.3.12 Revision Identification Registers (REVID1 and REVID2)

The revision identification registers (REVID1 and REVID2) contain peripheral identification data for the peripheral. REVID1 is shown in Figure 29-21 and described in Table 29-22. REVID2 is shown in Figure 29-22 and described in Table 29-23.

Figure 29-21. Revision Identification Register 1 (REVID1)

31	0
REVID1	
R-1102 0002h	

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

Table 29-22. Revision Identification Register 1 (REVID1) Field Descriptions

Bit	Field	Value	Description
31-0	REVID1	1102 0002h	Peripheral Identification Number

Figure 29-22. Revision Identification Register 2 (REVID2)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		REVID2	
R-0		R-0	

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

Table 29-23. Revision Identification Register 2 (REVID2) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	REVID2	0	Peripheral Identification Number

Copyright © 2013–2016, Texas Instruments Incorporated

29.3.13 Power and Emulation Management Register (PWREMU_MGMT)

The power and emulation management register (PWREMU_MGMT) is shown in Figure 29-23 and described in Table 29-24.

Figure 29-23. Power and Emulation Management Register (PWREMU_MGMT)

31	16
Reserved	
R-0	

15	14	13	12	1	0
Rsvd	UTRST	URRST	Reserved		FREE
R/W-0	R/W-0	R/W-0	R-1		R/W-0

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

Table 29-24. Power and Emulation Management Register (PWREMU_MGMT) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	Reserved	0	Reserved. This bit must always be written with a 0.
14	UTRST		UART transmitter reset. Resets and enables the transmitter.
		0	Transmitter is disabled and in reset state.
		1	Transmitter is enabled.
13	URRST		UART receiver reset. Resets and enables the receiver.
		0	Receiver is disabled and in reset state.
		1	Receiver is enabled.
12-1	Reserved	1	Reserved
0	FREE		Free-running enable mode bit. This bit determines the emulation mode functionality of the UART. When halted, the UART can handle register read/write requests, but does not generate any transmission/reception, interrupts or events.
		0	If a transmission is not in progress, the UART halts immediately. If a transmission is in progress, the UART halts after completion of the one-word transmission.
		1	Free-running mode is enabled; UART continues to run normally.

Copyright © 2013–2016, Texas Instruments Incorporated

29.3.14 Mode Definition Register (MDR)

The Mode Definition register (MDR) determines the over-sampling mode for the UART. MDR is shown in Figure 29-24 and described in Table 29-25.

Figure 29-24. Mode Definition Register (MDR)

31	16
Reserved	
R-0	

15	1	0
Reserved		OSM_SEL
R-0		R/W-0

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

Table 29-25. Mode Definition Register (MDR) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	OSM_SEL		Over-Sampling Mode Select.
		0	16× over-sampling.
		1	13× over-sampling.

Copyright © 2013–2016, Texas Instruments Incorporated

Universal Parallel Port (uPP)

This chapter describes the universal parallel port (uPP).

Topic **Page**

30.1 Introduction **1328**
30.2 Architecture **1331**
30.3 Registers **1348**

Copyright © 2013–2016, Texas Instruments Incorporated

30.1 Introduction

30.1.1 Purpose of the Peripheral

The universal parallel port (uPP) peripheral is a multichannel, high-speed parallel interface with dedicated data lines and minimal control signals. It is designed to interface cleanly with high-speed analog-to-digital converters (ADCs) or digital-to-analog converters (DACs) with up to 16-bit data width (per channel). It may also be interconnected with field-programmable gate arrays (FPGAs) or other uPP devices to achieve high-speed digital data transfer. It can operate in receive mode, transmit mode, or duplex mode, in which its individual channels operate in opposite directions.

The uPP peripheral includes an internal DMA controller to maximize throughput and minimize CPU overhead during high-speed data transmission. All uPP transactions use the internal DMA to feed data to or retrieve data from the I/O channels. The DMA controller includes two DMA channels, which typically service separate I/O channels. The uPP peripheral also supports data interleave mode, in which all DMA resources service a single I/O channel. In this mode, only one I/O channel may be used.

30.1.2 Features

For more information on the features and performance supported by the uPP peripheral, see your device-specific data manual.

30.1.3 Functional Block Diagram

Figure 30-1 provides a high-level view of the uPP peripheral internal logic. Note that this figure shows one particular configuration: Channel A receives and Channel B transmits. In general, each channel may operate in either direction.

Figure 30-2, Figure 30-3, Figure 30-4, and Figure 30-5 show simplified data paths through the uPP peripheral for various configurations. Note that these figures are examples and do not represent all possible configurations. More information on these and other modes of operation is given in subsequent sections.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 30-1. uPP Functional Block Diagram

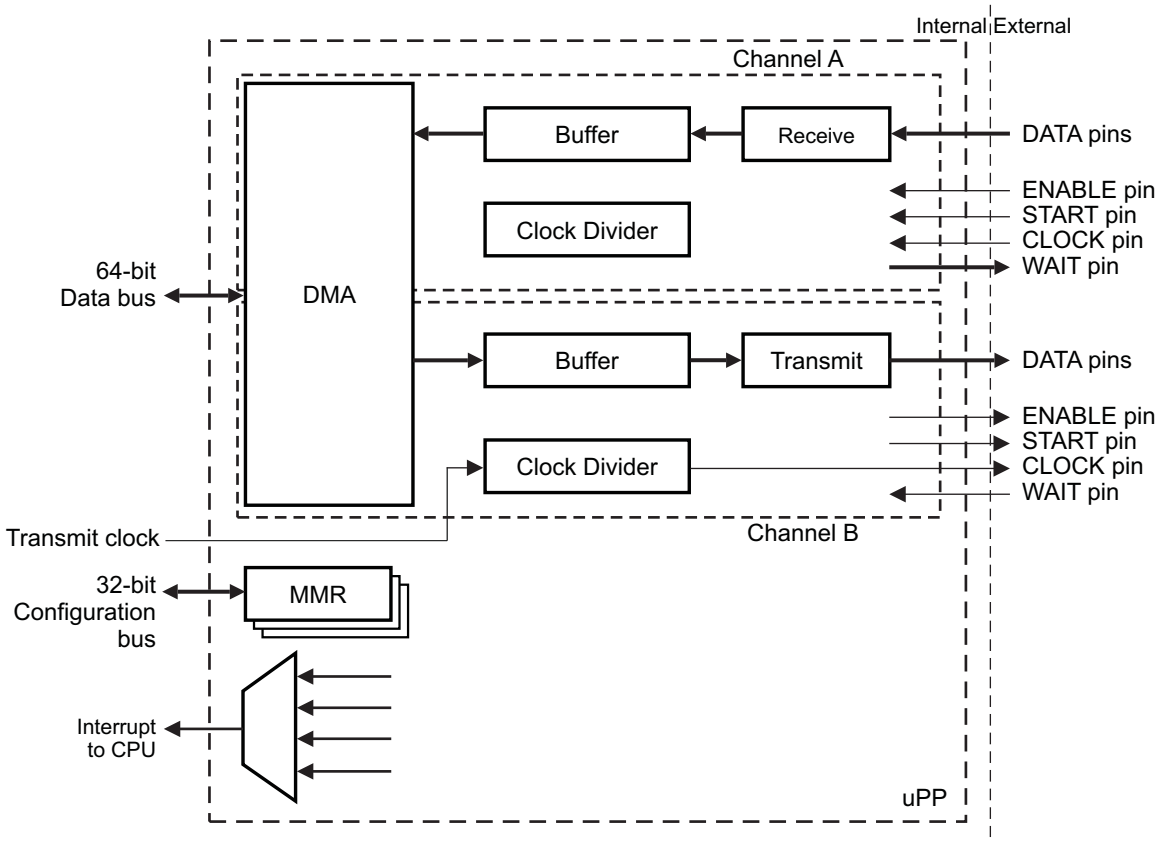


Figure 30-2. Data Flow for Single-Channel Receive Mode

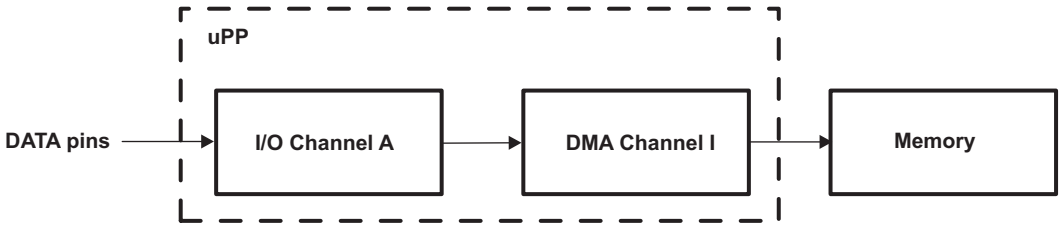
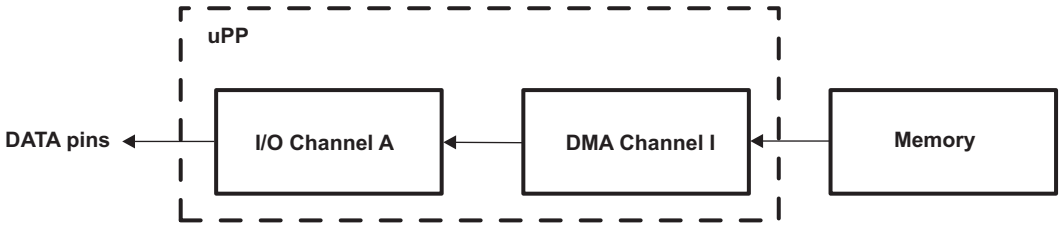


Figure 30-3. Data Flow for Single-Channel Transmit Mode



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 30-4. Data Flow for Digital Loopback (DLB) Mode (Duplex Mode 0)

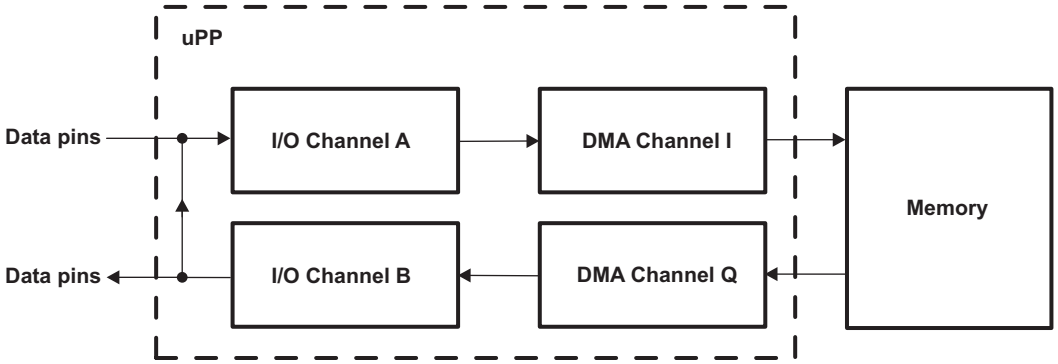
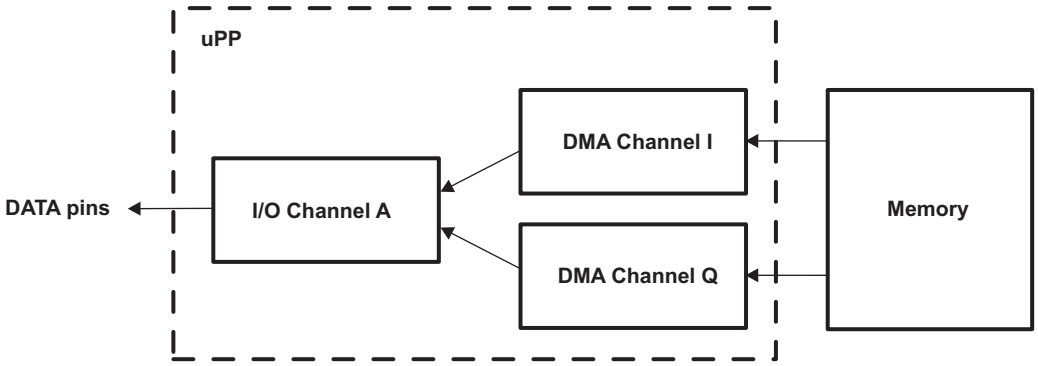


Figure 30-5. Data Flow for Single-Channel Transmit with Data Interleave



Copyright © 2013–2016, Texas Instruments Incorporated

30.2 Architecture

30.2.1 Clock Generation and Control

The uPP peripheral uses two separate clocks: a module clock that controls its internal logic, and a transmit clock that runs either interface channel in transmit mode. The source for each of these clocks may be configurable; see the *Device Clocking* chapter for more information. Neither the module clock nor the transmit clock can be faster than one-half the device CPU clock speed.

Each channel's CLOCK pin, or I/O clock, is obtained independently based on its operating direction.

30.2.1.1 Transmit Mode (Single Data Rate)

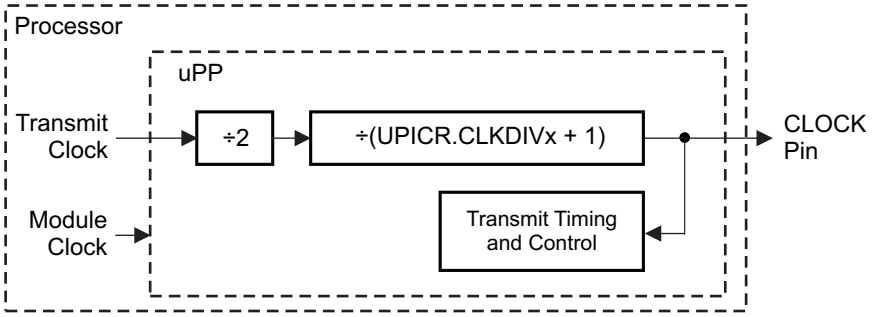
The channel drives a clock signal on its CLOCK pin. The uPP transmit clock is divided by a fixed value of 2, then divided again by a user-specified value between 1 and 16 (UPICR.CLKDIV*n* + 1). The resulting signal then drives the CLOCK pin. The following formula determines the final I/O clock speed:

```
I/O Clock = Transmit Clock/(2 ×
            (UPICR.CLKDIVn + 1))
```

The fixed divisor restricts the maximum speed of the I/O clock to one-fourth the device CPU clock speed.

Figure 30-6 shows the clock generation system for a channel configured in transmit mode.

Figure 30-6. Clock Generation for a Channel Configured in Transmit Mode

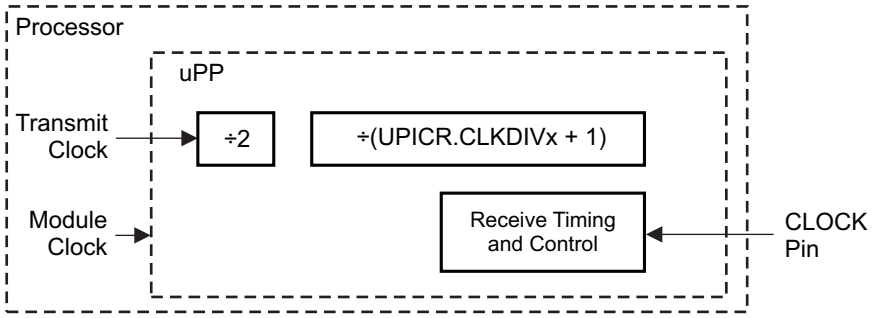


30.2.1.2 Receive Mode (Single Data Rate)

The channel requires an external clock to drive its CLOCK pin. The incoming clock is not divided, and its maximum allowed speed is one fourth (¼) the device CPU clock speed.

Figure 30-7 shows the clock generation system for a channel configured in receive mode.

Figure 30-7. Clock Generation for a Channel Configured in Receive Mode



Copyright © 2013–2016, Texas Instruments Incorporated

30.2.1.3 Double Data Rate

The uPP peripheral supports two I/O clocking schemes. The first, single data rate (SDR), clocks data from the DATA pins on either the rising edge or the falling edge (depending on UPICR.CLKINV*n*) of the I/O clock.

The second clocking scheme is double data rate (DDR). In this mode, data is clocked on both the rising and falling edges of the I/O clock. However, DDR mode imposes a lower I/O clock speed limit of one eighth (1/8) the device CPU clock for both transmit and receive modes. The operating speed for transmit mode with various divisors in each data rate are summarized in Table 30-1 (in this table, a data word is defined as the data represented on the DATA pins; uPP supports data words in the 8-bit to 16-bit range). In receive mode, a channel I/O clock is generated by an external source, but the same speed limit applies.

Table 30-1. I/O Clock Speeds for Channel in Transmit Mode Given 150 MHz Transmit Clock

		Word Rate (Mw/s)	
UPICR.CLKDIV*n*	I/O Clock (MHz)	Single Data Rate	Double Data Rate
0	75.00	75.00	...
1	37.50	37.50	75.00
2	25.00	25.00	50.00
3	18.75	18.75	37.50
...	...	...	...
15	4.69	4.69	9.38

Additional restrictions may apply, check the device datasheet to see if your particular uPP peripheral has any additional clock requirements.

30.2.2 Signal Description

Each uPP channel has its own set of control and data signals. Table 30-2 lists every signal and briefly describes their functions. Section 30.2.5 explains the uPP protocol.

Table 30-2. uPP Signal Descriptions

Signal	I/O Channel	Type (Transmit)	Type (Receive)	Description
DATA[15:0]	—	Output	Input	Parallel data bus
XDATA[15:0]	—	Output	Input	Extended parallel data bus
CHA_START	A	Output	Input	Indicates first data word per line of data
CHA_ENABLE	A	Output	Input	Indicates data transmission active
CHA_WAIT	A	Input	Output	Requests transmitter halt temporarily
CHA_CLOCK	A	Output	Input	Source-synchronous clock signal
CHB_START	B	Output	Input	Indicates first data word per line of data
CHB_ENABLE	B	Output	Input	Indicates data transmission active
CHB_WAIT	B	Input	Output	Requests transmitter halt temporarily
CHB_CLOCK	B	Output	Input	Source-synchronous clock signal
UPP_2xTXCLK	—	Input	—	Optional external source for transmit clock(1)

(1) This clock can only be used in transmit mode, and must be twice the speed of your desired I/O clock. See Section 30.2.1 and the *Device Clocking* chapter for more information.

Note that the DATA and XDATA pins are not dedicated to a single I/O channel in the same way as the control signals. For practical reasons, uPP data pin channel assignments are not static. Instead, the data pins used by each I/O channel (A, B) depend on the operating mode of the uPP peripheral. Table 30-3 summarizes the assignment of the DATA and XDATA pins to each channel for various operating modes, along with the relevant register settings. For more information on these pins, see your device-specific data manual.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 30-3. DATA and XDATA Pin Assignments to Channels A and B According to Operating Mode

uPP Channel Control Register (UPCTL) Bit			Assigned Channel			
CHN	IWA	IWB	DATA[15:8]	DATA[7:0]	XDATA[15:8]	XDATA[7:0]
0	0	x	—	A[7:0]	—	—
0	1	x	A[15:8]	A[7:0]	—	—
1	0	0	B[7:0]	A[7:0]	—	—
1	0	1	B[7:0]	A[7:0]	B[15:8]	—
1	1	0	B[7:0]	A[7:0]	—	A[15:8]
1	1	1	B[7:0]	A[7:0]	B[15:8]	A[15:8]

30.2.3 Pin Multiplexing

Extensive pin multiplexing is used to accommodate the largest number of peripheral functions in the smallest possible package. Pin multiplexing is controlled using a combination of hardware configuration at device reset and software programmable register settings. To determine how pin multiplexing affects the uPP peripheral, see your device-specific data manual.

30.2.4 Internal DMA Controller Description

The uPP peripheral includes an internal DMA controller separate from any device-level DMA, such as EDMA. The internal DMA controller consists of two DMA channels, I and Q, which move data to/from the uPP peripheral interface (I/O) channels in all operating modes. This section describes how to program the internal DMA channels.

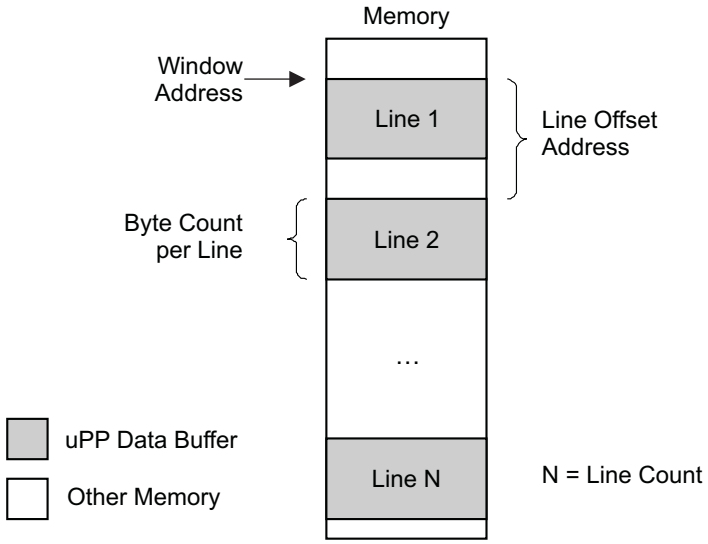
30.2.4.1 DMA Programming Concepts

The uPP internal DMA controller uses a simplified programming model similar to 2D transfers performed by the EDMA (see the *Enhanced Direct Memory Access (EDMA3) Controller* chapter for more information). Each DMA channel may be configured with four parameters: window address, byte count, line count, and line offset address. Figure 30-8 shows a typical DMA window defined by these parameters.

- **Window Address** (UPxD0.ADDR) – The location in memory of the first byte in the data buffer. When the uPP operates in receive mode, the DMA channel begins writing to this address as it takes incoming data from the uPP I/O channel. When the uPP operates in transmit mode, the DMA channel begins reading from this address and pass the data to the uPP I/O channel. The window address can reside in any available memory space (including EMIF), but it must be aligned to a 64-bit boundary (that is, the 3 LSBs must equal 0). Nonaligned addresses are automatically adjusted to a properly aligned value when written to UPxD0.
- **Byte Count** (UPxD1.BCNT) – The number of bytes per line. The byte count must be an even number.
- **Line Count** (UPxD1.LNCNT) – The number of lines per window. The total number of bytes transferred equals B × L, where B is the byte count per line and L is the line count.
- **Line Offset Address** (UPxD2.LNOFFSET) – The offset address between the first byte in successive lines. The line offset address cannot exceed 65528 (FFF8h) bytes, and must be aligned to a 64-bit boundary in memory (that is, the 3 LSBs must equal 0).

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 30-8. Structure of DMA Window and Lines in Memory



Certain values of the line offset address have special implications on the structure of the data buffer:

- **Line Offset Address = Byte Count** – Data buffer is a contiguous block in memory with size equal to (Line Count) × (Byte Count).
- **Line Offset Address = 0** – Data buffer consists of a single line, with total size equal to Byte Count. If the I/O channel is configured in transmit mode, this line is transmitted (Line Count) consecutive times before the DMA transfer completes. If the I/O channel is configured in receive mode, the buffer is repeatedly written and overwritten by incoming data.

To program a DMA transfer, write the appropriate fields in the DMA channel descriptor registers, UPID*n* for DMA Channel I or UPQD*n* for DMA Channel Q. If the associated I/O channel is initialized and idle, the DMA transfer and I/O transaction begins immediately. Section 30.2.6 describes a step-by-step process for configuring the I/O and DMA channels and starting a uPP transfer.

Each DMA channel allows a second descriptor to be queued while the previously programmed DMA transfer is still running. The UPxS2.PEND bit reports whether a new set of DMA parameters may be written to the DMA descriptor registers. Each DMA channel can have at most one active transfer and one queued transfer. This allows each I/O channel to perform uninterrupted, consecutive transactions across DMA transfer boundaries.

The internal DMA controller does not support automatically reloading DMA transfer descriptors. Each successive descriptor set must be explicitly written to the UPxD*n* registers by software.

All uPP interrupt events originate in the internal DMA controller. Section 30.2.8 lists and explains all uPP interrupt events.

The internal DMA controller always writes data in bursts of 64 bytes. However, DMA read operations have configurable burst size, which may be set per channel using the RDSIZEI and RDSIZEQ bits in the uPP threshold configuration register (UPTCR). A DMA channel waits until the specified number of bytes leaves its internal buffer before performing another burst read from memory.

Note that the TXSIZEA and TXSIZEB bits in UPTCR are not DMA parameters; instead, they control transmit thresholds for the uPP interface channels.

Copyright © 2013–2016, Texas Instruments Incorporated

30.2.4.2 Data Interleave Mode

The data interleave mode is a special configuration that maps both DMA channels to a single interface channel. Since there are only two DMA channels in the uPP peripheral, data interleave mode can only be used when the uPP peripheral is operated in single-channel mode. There are two variants on data interleave mode, each with special conditions:

- Single Data Rate (SDR) Interleave – Transmit Only
  - UPCTL.CHN = 0 (single-channel mode)
  - UPCTL.DRA = 0 (single data rate)
  - UPCTL.MODE = 1 (transmit mode)
  - UPCTL.SDRTXIL = 1 (enable SDR transmit interleave)
  - UPCTL.DDRDEMUX = 0 (disable DDR interleave)
- Double Data Rate (DDR) Interleave – Transmit or Receive
  - UPCTL.CHN = 0 (single-channel mode)
  - UPCTL.DRA = 1 (double data rate)
  - UPCTL.MODE = 0 or 1 (receive or transmit mode; not duplex)
  - UPCTL.SDRTXIL = 0 (disable SDR transmit interleave)
  - UPCTL.DDRDEMUX = 1 (enable DDR interleave)

Section 30.2.1 describes the differences between single data rate (SDR) and double data rate (DDR).

In data interleave mode, only I/O Channel A is used. This single channel is associated with two data buffers, each serviced by its own DMA channel (I and Q). In SDR interleave mode, the START signal is used as a buffer selection line: START = 1 indicates that the current word comes from DMA Channel I; START = 0 indicates that the current word comes from DMA Channel Q. In DDR Interleave mode, the data buffers alternate every word beginning with Channel I: Channel I Word 0, Channel Q Word 0, Channel I Word 1, Channel Q Word 1, etc. Section 30.2.5 shows signal diagrams for both data interleave modes.

30.2.4.3 Interface and DMA Channel Mapping

Typically, DMA Channels I and Q are mapped to interface Channels A and B, respectively. Data interleave mode is the exception, since it allocates both DMA channels to service interface Channel A. Table 30-4 summarizes DMA channel mapping for various modes of operation.

Table 30-4. Interface and DMA Channel Mapping for Various Operating Modes

Operating Mode	I/O Channel Serviced	
	DMA I	DMA Q
1-Channel Receive	A	—
1-Channel Transmit	A	—
2-Channel Receive	A	B
2-Channel Transmit	A	B
2-Channel Duplex	A	B
1-Channel Transmit (Interleave)	A	A
1-Channel Receive (Interleave)	A	A

Copyright © 2013–2016, Texas Instruments Incorporated

30.2.5 Protocol Description

The uPP peripheral consists of two independent channels, each possessing its own data lines and control signals. A channel may be configured to run in transmit or receive mode and to use either 8 or 16 data lines (8-bit or 16-bit mode) using the uPP channel control register (UPCTL). A channel may also be configured to ignore certain control signals using the uPP interface configuration register (UPICR). Each uPP defaults to 8-bit mode and uses all four control signals, unless otherwise configured. Table 30-5 summarizes the signals that are required for basic operation in receive and transmit modes. The following subsections describe the role of each signal.

Table 30-5. Required Signals for Various Modes

	Signal Required?	
Signal Name	Transmit Mode	Receive Mode
DATA[7:0]	√	√
DATA[15:8]		
START	√	
ENABLE	√	
WAIT		√
CLOCK	√	√

30.2.5.1 DATA[7:0] Signals

In 8-bit mode, DATA[7:0] comprise the channel's entire data bus. In 16-bit mode, DATA[7:0] comprise the least-significant bits of the 16-bit word. The channel's data width is selected using the IWx bit in UPCTL.

In transmit mode, these pins are outputs that transmit data supplied by the channel's associated DMA channel. While the channel is idle, their behavior depends on the TRISx bit in UPICR. These pins can be configured to drive an idle value (TRISx = 0, VALx field in the uPP interface idle value register (UPIVR)) or be in a high-impedance state while idle (TRISx = 1).

In receive mode, these pins are inputs that provide data to the channel's associated DMA channel.

Note that the DATA signals map differently to the DATA and XDATA pins for various uPP configurations, see Section 30.2.2 for more information.

30.2.5.2 DATA[15:8] Signals

In 8-bit mode, DATA[15:8] are not used. In 16-bit mode, DATA[15:8] comprise the most-significant bits of the 16-bit word. The channel's data width is selected using the IWx bit in UPCTL. A channel may be further configured to use only part of its DATA[15:8] pins, which allows any total data width from 8 to 16 bits. Section 30.2.5.8 describes data format and packing in the 9-bit to 15-bit configurations.

While in use, the direction and behavior of DATA[15:8] in transmit and receive modes are the same as the direction and behavior of DATA[7:0].

Note that the DATA signals map differently to the DATA and XDATA pins for various uPP configurations, see Section 30.2.2 for more information.

30.2.5.3 START Signal

The uPP transmitter asserts the START signal when it transfers the first word of a data line. A line is defined in terms of the channel's associated DMA channel; for more on DMA programming concepts, see Section 30.2.4. The START signal is active-high by default, but its polarity is controlled by the STARTPOLx bit in UPICR.

In transmit mode, START is an output signal and is always driven; in receive mode, START is an input signal and may be disabled using the STARTx bit in UPICR.

Copyright © 2013–2016, Texas Instruments Incorporated

When the channel is configured in transmit mode with data interleave enabled (SDRTXIL = 1 in UPCTL), the START signal function changes completely. The START signal now asserts on every data word that is provided by DMA Channel I. See Section 30.2.5.7 for this alternative behavior.

30.2.5.4 ENABLE Signal

The uPP transmitter asserts the ENABLE signal when it transfers a valid data word. The ENABLE signal is active-high by default, but its polarity is controlled by the ENAPOLx bit in UPICR.

In transmit mode, ENABLE is an output signal and is always driven; in receive mode, ENABLE is an input signal and may be disabled using the ENAx bit in UPICR.

30.2.5.5 WAIT Signal

The WAIT signal allows the receiver to request a temporary halt in data transmission. When the receiver asserts WAIT, the transmitter responds by stopping transmission (starting with the next word) until WAIT is released. The receiver ignores all incoming data until WAIT is released. Once WAIT is released, the transmitter can resume transmission on the next word. Figure 30-10 shows WAIT signal timing. The WAIT signal is active-high by default, but its polarity is controlled by the WAITPOLx bit in UPICR.

In transmit mode, WAIT is an input signal and may be disabled using the WAITx bit in UPICR; in receive mode, WAIT is an output signal.

30.2.5.6 CLOCK Signal

The uPP transmitter drives the CLOCK signal to align all other uPP signals. By default, other signals align on the rising edge of CLOCK, but its polarity is controlled by the CLKINVx bit in UPICR. The active edge(s) of CLOCK should always slightly precede transitions of other uPP signals.

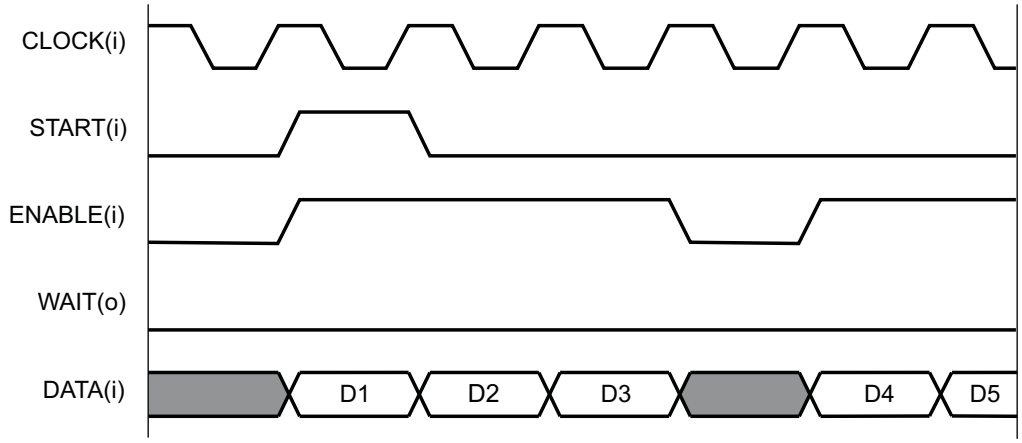
In transmit mode, CLOCK is an output signal; in receive mode, CLOCK is an input signal. See Section 30.2.1 for more information on clock generation and allowed frequencies.

30.2.5.7 Signal Timing Diagrams

In the following diagrams, signals are marked (I) to indicate that they are inputs to the uPP peripheral and (o) to indicate that they are outputs from the uPP peripheral. Data words from a single DMA channel are designated Dx, while data words that must come from a specific DMA channel are designated Ix or Qx to indicate DMA Channel I or Q, respectively. For more information on DMA channels and data interleave mode, see Section 30.2.4.

All signal diagrams are drawn with signal polarities in their default states. All signals except DATA are independently configurable in the uPP interface configuration register (UPICR).

Figure 30-9. Signal Timing for uPP Channel in Receive Mode with Single Data Rate



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 30-10. Signal Timing for uPP Channel in Transmit Mode with Single Data Rate

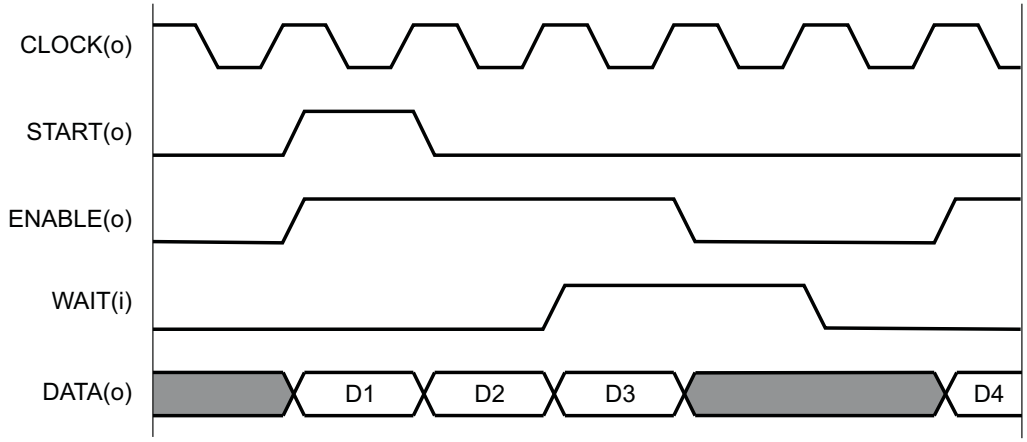


Figure 30-11. Signal Timing for uPP Channel in Receive Mode with Double Data Rate

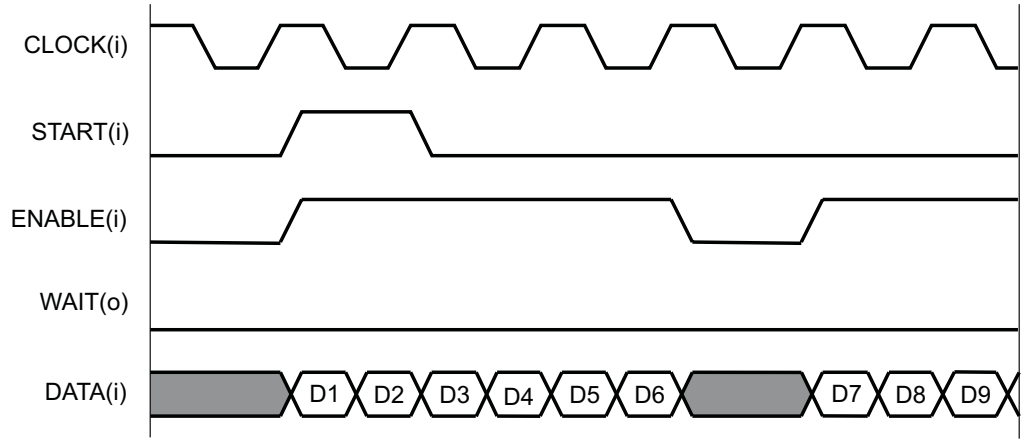
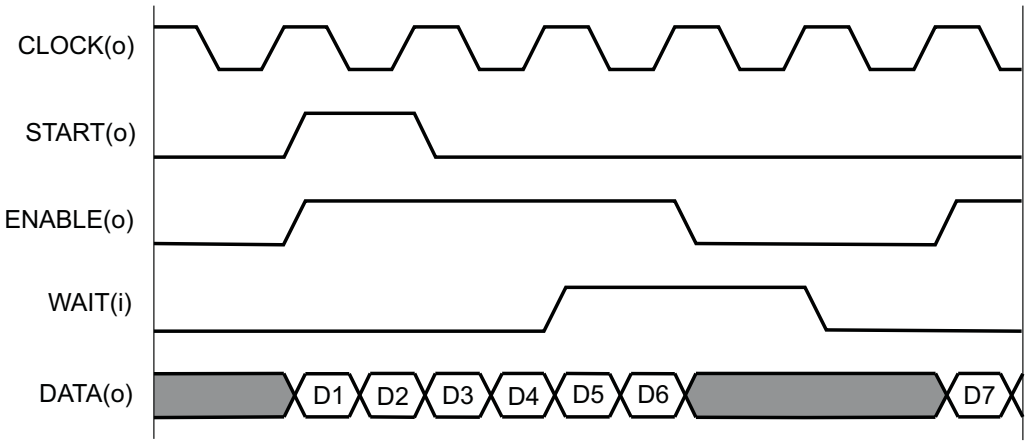


Figure 30-12. Signal Timing for uPP Channel in Transmit Mode with Double Data Rate



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 30-13. Signal Timing for uPP Channel in Receive Mode with Double Data Rate and Data Interleave Enabled (via UPCTL.DDRDEMUX)

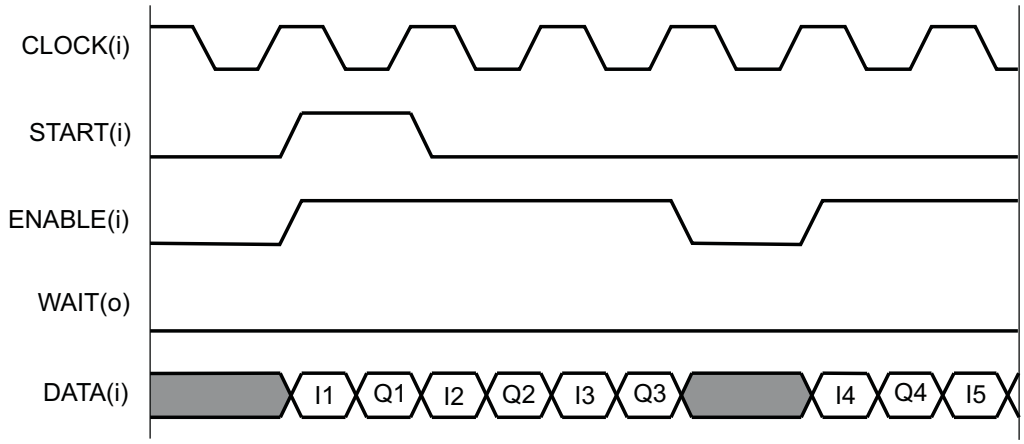


Figure 30-14. Signal Timing for uPP Channel in Transmit Mode with Double Data Rate and Data Interleave Enabled (via UPCTL.DDRDEMUX)

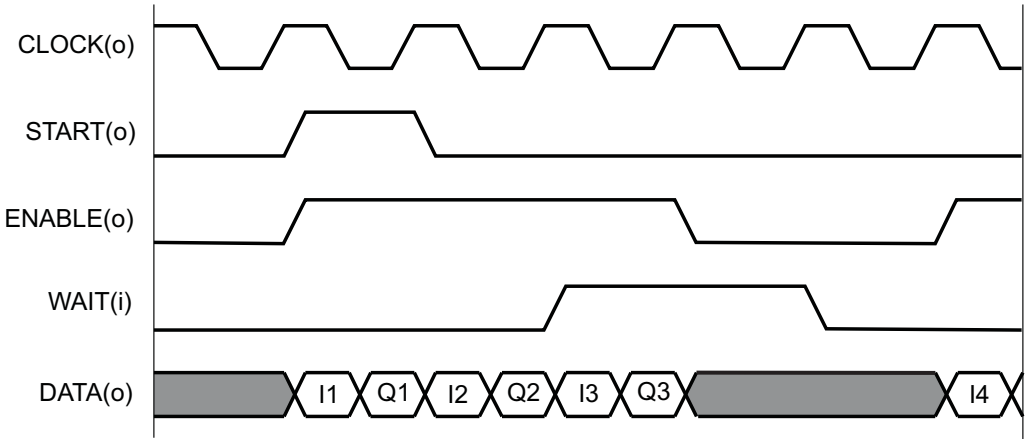
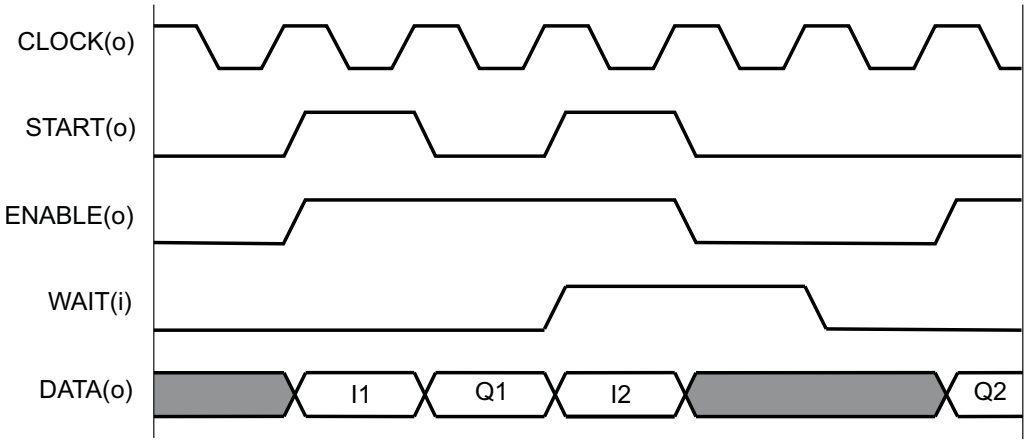


Figure 30-15. Signal Timing for uPP Channel in Transmit Mode with Single Data Rate and Data Interleave Enabled (via UPCTL.SDRTXIL)



NOTE: START asserts on every data word from DMA Channel I.

Copyright © 2013–2016, Texas Instruments Incorporated

30.2.5.8 Data Format

The uPP peripheral supports any data word width between 8 to 16 bits using the IWx and DPWx bits in the uPP channel control register (UPCTL). For 8-bit operation, uPP reads/writes 8-bit words in memory; for 16-bit operation, uPP reads/writes 16-bit words in memory.

For N-bit operation (8 < N < 16), the uPP peripheral reads/writes 16-bit words in memory. The “extra” bits are filled by the uPP according to a data packing scheme, selected by the DFWx bit in UPCTL. There are three selectable data packing modes:

- Right-Justify, Zero Extend – Data occupies N LSBs. The (16 – N) MSBs are cleared to 0.
- Right-Justify, Sign Extend – Data occupies N LSBs. The (16 – N) MSBs are the same value as the (N – 1) bit.
- Left-Justify, Zero Fill – Data occupies N MSBs. The (16 – N) LSBs are cleared to 0.

Table 30-6 lists some example data for N = 12 (that is, 12-bit operation). In transmit mode, the packed version of each data word from memory is transmitted using all 16 data pins allotted to the uPP channel. In receive mode, the packed version of each incoming data word (using only N data pins) is stored in memory.

Table 30-6. Data Packing Examples for 12-Bit Data Words

12-Bit Data Word	Right-Justify, Zero Extend	Right-Justify, Sign Extend	Left-Justify, Zero Fill
123h	0123h	0123h	1230h
ABCh	0ABCh	FABCh	ABC0h
000h	0000h	0000h	0000h
800h	0800h	F800h	8000h
FFFh	0FFFh	FFFFh	FFF0h

Copyright © 2013–2016, Texas Instruments Incorporated

30.2.6 Initialization and Operation

30.2.6.1 Step-by-Step Procedure

NOTE: When initializing the uPP peripheral, the uPP interface configuration register (UPICR) must be programmed using a single, 32-bit write. Writing UPICR fields one-by-one can lead to unexpected results.

This section provides step-by-step instructions for initializing and running the uPP peripheral in various modes. These instructions are given assuming that the device has just come out of a power-on reset (POR) state.

1. Apply the appropriate pin multiplexing settings. See the *System Configuration (SYSCFG) Module* chapter, your device-specific data manual, and/or pin multiplexing utility for more information.
2. Enable the power and clocks to the uPP peripheral. See the *Device Clocking* chapter for more information.
3. Set the SWRST bit in the uPP peripheral control register (UPPCR) to 1 to place uPP in software reset.
4. Wait at least 200 device clock cycles, then clear the SWRST bit to 0 to bring the module out of reset.
5. Program the uPP configuration registers: UPCTL, UPICR, UPIVR, UPTCR, and UPDLB. The basic function of each register is summarized here; for more information, see Section 30.3.
   (a) UPCTL – Transmit/receive selection (see Table 30-7), data width, data format, data rate, data interleave enable
   (b) UPICR – Signal enable, signal inversion, clock divisor (transmit only)
   (c) UPIVR – Idle value (transmit only)
   (d) UPTCR – I/O transmit threshold (transmit only), DMA read burst size
   (e) UPDLB – Digital loopback (see Table 30-7)
6. Program the uPP interrupt enable set register (UPIES) to interrupt generation for the desired events. Register an interrupt service routine (ISR) if desired; otherwise, polling is required.
7. Set the EN bit in the uPP peripheral control register (UPPCR) to 1 to turn on the uPP peripheral.
8. Allocate and/or initialize data buffers for use with uPP.
9. Program the DMA channels with their first transfers using the uPP DMA channel descriptor registers: UPID0-2 and/or UPQD0-2.
10. Watch for interrupt events. Reprogram the DMA as necessary (checking that the PEND bit in the uPP DMA channel status register (UPxS2) is 0).
    (a) If polling, check UPIES. Reading a bit as 1 indicates the corresponding event has occurred. Write the corresponding bit with 1 to clear.
    (b) If using ISR, check UPIES inside your ISR. Structure your ISR according to the pseudo-code in Section 30.2.6.4.

Copyright © 2013–2016, Texas Instruments Incorporated

30.2.6.2 Sample Configuration Settings

The uPP peripheral is flexible, with several orthogonal configuration choices. Table 30-7 summarizes selecting the fundamental operating mode of the module.

NOTE: Digital loopback (DLB) mode is a configuration that the uPP peripheral internally routes data and control signals from one channel to the other. DLB can only be used when the peripheral is configured in duplex mode (that is, UPCTL.MODE = 2h or 3h). DLB is primarily useful for debug purposes, and requires no physical connections between channels. The standard uPP pin multiplexing must be applied, however, even though the pins are not used.

Table 30-7. Basic Operating Mode Selection

	uPP Channel Control Register (UPCTL) Bit		uPP Digital Loopback Register (UPDLB) Bit	
Operating Mode	**CHN**	**MODE**	**AB**	**BA**
1-Channel Transmit	0	1	0	0
1-Channel Receive	0	0	0	0
2-Channel Transmit	1	1	0	0
2-Channel Receive	1	0	0	0
2-Channel Duplex 0	1	2h	0	0
2-Channel Duplex 1	1	3h	0	0
2-Channel Duplex 0 (DLB)	1	2h	0	1
2-Channel Duplex 1 (DLB)	1	3h	1	0

Other than Table 30-7, there are several more choices to make (per channel):

- Data width – 8-bit, 9-bit to 16-bit
- Data packing – 9-bit to 15-bit data width only
- Data rate – single, double
- Data interleave –single channel only
- Clock divisor – transmit only
- Individual control signal enable
- Individual control signal polarity
- Idle value – transmit only
- Transmit threshold – transmit only
- DMA read burst size

Copyright © 2013–2016, Texas Instruments Incorporated

Table 30-8 lists an example set of uPP parameters for duplex mode 0. This configuration places the uPP peripheral in duplex mode with Channel A receiving and Channel B transmitting. Each channel uses a 16-bit interface with a different data format.

Table 30-8. Sample uPP Parameters for Duplex Mode 0

Register	Register Field[1]	Setting	Description
UPCTL	DPFB	2h	Data packing: left-justified, zero fill
	DPWB	4h	12-bit data format
	IWB	1	16-bit
	DRB	0	Single data rate
	DPFA	—	Unused
	DPWA	0	16-bit data format
	IWA	1	16-bit
	DRA	0	Single data rate
	CHN	1	2-Channel
	MODE	2h	Duplex 0: A receive, B transmit
UPICR	CLKDIVB	1	Divide by 2 (total division of transmit clock: 4)
	CLKDIVA	-—	Unused
UPIVR	VALB	0BBBh	Note idle value is 12-bit data format; 4 MSBs unused
	VALA	—	Unused
UPIES	EOLQ	1	Turn on EOL interrupt for Channel B (DMA Channel Q)
	EOWQ	1	Turn on EOW interrupt for Channel B (DMA Channel Q)
	EOLI	1	Turn on EOL interrupt for Channel A (DMA Channel I)
	EOWI	1	Turn on EOW interrupt for Channel A (DMA Channel I)

(1) Unlisted register fields are left at their default values (typically 0), see Section 30.3.

Copyright © 2013–2016, Texas Instruments Incorporated

30.2.6.3 System Tuning Tips

The uPP peripheral can operate at high speed and transfer data at a very high rate. When operating the uPP near its upper limits, tuning certain parameters can help decrease the incidence of errors and the software overhead incurred servicing uPP data. Table 30-9 lists several parameters that can be useful in system tuning. A parameter is defined as a “coarse” adjustment, if changing the parameter directly alters the peripheral throughput. A “fine” adjustment does not change the peripheral throughput, but it does affect general system performance.

Table 30-9. uPP Parameters Useful for System Tuning

Parameter	Register	Register Field	Edge Value	Safe Value	Description
Data Rate	UPCTL	DRB DRA	1	0	Double data rate increases data transfer by a factor of 2 and greatly increases system loading for the same clock divisor. This is a coarse adjustment and is probably fixed due to design constraints.
Clock Division	UPICR	CLKDIVB CLKDIVA	0	1+	Increasing clock division is the most straight-forward way to decrease system loading. This is a coarse adjustment; the difference between CLKDIVx = 0 and 1 is the same (in terms of data rate) as the difference between single and double data rate.
DMA Read Burst Size	UPTCR	RDSIZEQ RDSIZEI	0	3h	Increasing the DMA read threshold decreases system loading by generating fewer, larger DMA events. This is a fine adjustment.
DMA Line Size, Count	UPxD1	LNCNT BCNT		(1)(1)	Condensing uPP transfers into fewer, larger lines generates fewer end-of-line interrupts and, thus, invokes fewer ISR calls. This is a fine adjustment.
Total Transfer Size	UPxD1	LNCNT BCNT	(1)	(1)	Performing many small uPP transfers can require excessive software overhead (programming DMA descriptors, handling interrupts, etc.) at high data rates. This is a fine adjustment.
System Priority			(2)(2)	(2)	When the uPP operates in parallel with other data masters, such as EDMA, assigning higher priority to the uPP may help the uPP avoid underflow or overflow conditions. This is a fine adjustment.

(1) These values vary per application. One example could be a 16-KB transfer. The same total data could be transferred as 16 1-KB lines or 2 8-KB lines.

(2) System priority settings are not set within the uPP peripheral. See the *System Configuration (SYSCFG) Module* chapter for more information.

30.2.6.4 Sample Interrupt Service Routine

The following pseudo-code serves as a template for writing a uPP interrupt service routing (ISR) function. Note that the uPP combines all events into a single CPU interrupt, and a new interrupt does not call the ISR if the previous interrupt still has not returned from the ISR. To allow future ISR calls, the uPP end-of-interrupt register (UPEOI) must be written with a zero value. Thus, the ISR should check for multiple events, and should continue rechecking after handling each individual event until no more events are found. Then, it must write UPEOI = 0 before returning.

Copyright © 2013–2016, Texas Instruments Incorporated

```
Function upp_isr
{
      interrupt_status = UPIER
      while (interrupt_status != 0)
      {
            if (interrupt_status.EOLI)
            {
                  UPIER.EOLI = 1      // clear EOLI
                  // Handle EOLI...
            }
            if (interrupt_status.EOWI)
            {
                  UPIER.EOWI = 1      // clear EOWI
                  // Handle EOWI...
            }
            if (interrupt_status.ERRI)
            {
                  UPIER.ERRI = 1      // clear ERRI
                  // Handle ERRI...
            }
            if (interrupt_status.UORI)
            {
                  UPIER.UORI = 1      // clear UORI
                  // Handle UORI...
            }
            if (interrupt_status.DPEI)
            {
                  UPIER.DPEI = 1      // clear DPEI
                  // Handle DPEI...
            }
            if (interrupt_status.EOLQ)
            {
                  UPIER.EOLQ = 1      // clear EOLQ
                  // Handle EOLQ...
            }
            if (interrupt_status.EOWQ)
            {
                  UPIER.EOWQ = 1      // clear EOWQ
                  // Handle EOWQ...
            }
            if (interrupt_status.ERRQ)
            {
                  UPIER.ERRQ = 1      // clear ERRQ
                  // Handle ERRQ...
            }
            if (interrupt_status.UORQ)
            {
                  UPIER.UORQ = 1      // clear UORQ
                  // Handle UORQ...
            }
            if (interrupt_status.DPEQ)
            {
                  UPIER.DPEQ = 1      // clear DPEQ
                  // Handle DPEQ...
            }

            // loop again if any interrupts are left
               interrupt_status = UPIER
      } // end of while

       // write end of interrupt vector to allow future calls
          UPEOI = 0
} // end of function
```


Copyright © 2013–2016, Texas Instruments Incorporated

30.2.7 Reset Considerations

30.2.7.1 Software Reset

Software reset clears the uPP internal state machines but does not reset the contents of the uPP registers. The following procedure performs a software reset on the uPP peripheral.

1. Write the EN bit in the uPP peripheral control register (UPPCR) to 0 (disables the uPP).
2. Poll the DB bit in UPPCR for activity; wait until DMA controller is inactive/idle.
3. Write the SWRST bit UPPCR to 1 (places uPP in software reset).
4. Write the SWRST bit UPPCR to 0 to (brings uPP out of software reset).

30.2.7.2 Hardware Reset

When the processor RESET pin is asserted, the entire processor is reset and is held in the reset state until the RESET pin is released. As part of a device reset, the uPP state machines are reset, and the uPP registers are forced to their default states (see Section 30.3).

30.2.8 Interrupt Support

The uPP peripheral generates eight interrupt events, all tied to internal DMA Channels I and Q. The uPP peripheral automatically combines all interrupt events into a single chip-level (CPU) interrupt. Individual events may be enabled using the uPP interrupt enable set register (UPIES) and disabled using the uPP interrupt enable clear register (UPIEC). Only enabled events generate interrupts and assert bits in the interrupt enabled status register (UPIER). Disabled events do not generate interrupts but do assert bits in the interrupt raw status register (UPISR).

An interrupt service routine (ISR) may be assigned to handle uPP CPU-level interrupts using the interrupt controller module. If uPP events occur in close proximity to one another, a single CPU interrupt (and a single call to the ISR) may represent multiple interrupt events. Thus, the uPP ISR must meet certain structural requirements:

- The ISR must be able to handle multiple events before returning.
- The ISR must handle any subsequent events that occur after it is called but before it returns.
- The ISR must write 00h to the uPP end-of-interrupt register (UPEOI) just before it returns. This allows future uPP events to generate a CPU interrupt.

Section 30.2.6 provides a sample ISR that demonstrates these requirements.

30.2.8.1 End of Line (EOL) Event

This event occurs each time that the DMA channel reaches the end of a line in the data window. Note that if the associated uPP interface channel is operating in transmit mode, this event may occur before the line's final bytes are actually transmitted over the data pins.

For small line size and fast data transfer, it is possible to "miss" EOL events if they occur faster than the user's code can handle them. This does not hinder uPP operation; the uPP peripheral continue processing data uninterrupted until the EOW event or some error condition is encountered.

30.2.8.2 End of Window (EOW) Event

This event occurs when the DMA channel reaches the end of its current data window. Note that if the associated uPP interface channel is operating in transmit mode, this event may occur shortly before the window's final bytes are actually transmitted over the data pins.

When an EOW event occurs, the DMA channel automatically begins the next DMA transfer if one has been pre-programmed into the channel descriptor registers. If no new transfer is preprogrammed, the DMA channel becomes idle. For small window size and fast data transfer, code overhead may make it impossible to maintain a constant flow of data through the uPP interface channel. This problem can be solved by increasing the DMA window size or decreasing the peripheral clock speed.

Copyright © 2013–2016, Texas Instruments Incorporated

30.2.8.3 Internal Bus Error (ERR) Event

This event occurs when the uPP peripheral or its internal DMA controller encounters an internal bus error. After encountering this error, the uPP peripheral should be reset to avoid further problem.

30.2.8.4 Underrun or Overflow (UOR) Event

This event occurs when the DMA channel fails to keep up with incoming or outgoing data on its associated interface channel. Typically, this error indicates that background system activity has interfered with normal operation of the peripheral. It does not occur simply when a channel is allowed to idle. After encountering this error, the uPP peripheral should be reset when this event occurs.

This error should primarily occur when operating the uPP at high speed with significant system loading. To avoid this error, run the uPP at slower speeds or reduce background activity, such as non-uPP peripheral or DMA transactions. Additional tuning tips are given in Section 30.2.6.3.

30.2.8.5 DMA Programming Error (DPE) Event

This event occurs when the DMA channel descriptors are programmed while its PEND bit in the uPP DMA channel status register (UPxS2) is set to 1. A channel's descriptors should only be programmed while the channel's PEND bit is cleared to 0.

30.2.9 Power Management

The uPP peripheral can be placed in reduced-power modes to conserve power during periods of inactivity. For information on power management, see the *Power Management* chapter.

30.2.10 Emulation Considerations

The uPP peripheral stops running if any of three conditions are met:

- Peripheral Disable – EN bit in the uPP peripheral control register (UPPCR) is 0.
- Clock Stop – uPP acknowledges a clock stop request from the device power management module.
- Emulation Suspend – JTAG emulator halts chip while FREE = 0 and SOFT = 1 in UPPCR.

For other settings of FREE and SOFT, the uPP peripheral continues running during emulation halt.

When the uPP encounters a stop condition, it completes the current DMA burst transaction (if one is active) before stopping.

An I/O channel configured in transmit mode immediately places its pins in a high-impedance state and preserves the state of its internal state machines. Unless some reset event occurs (see Section 30.2.7), the channel can resume where it left off when the stop condition is cleared.

An I/O channel configured in receive mode immediately asserts its WAIT signal (see Section 30.2.5.5) and captures one additional data word. Further incoming data words are ignored as long as the stop condition persists.

30.2.11 Transmit and Receive FIFOs

Each of the uPP peripheral I/O channels has a 512-byte FIFO. In receive mode, the FIFO is divided into eight 64-byte blocks. In transmit mode, the FIFO is divided into blocks that can be set to 64, 128, or 256 bytes, configured by the TXSIZEA or TXSIZEB field in the uPP threshold configuration register (UPTCR). Transmission will not begin until the channel has loaded enough data to fill at least one full FIFO block.

The internal DMA channels may also be configured to use a read threshold of 64, 128, or 256 bytes using the RDSIZEI or RDSIZEQ field in UPTCR. The DMA write threshold is fixed at 64 bytes.

Copyright © 2013–2016, Texas Instruments Incorporated

30.3 Registers

The system programmer has access to and control over any of the uPP registers that are listed in Table 30-10. These registers, which control uPP I/O and DMA operations, are available at 32-bit addresses in the device memory-map. The device-specific data sheet lists the base memory address of these registers.

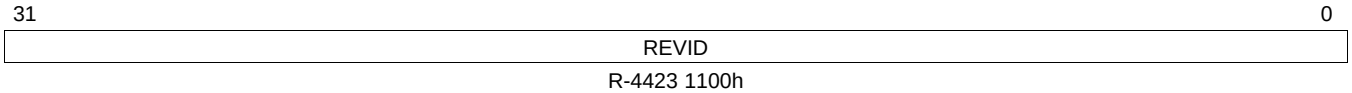
Table 30-10. uPP Registers

Address Offset	Acronym	Register Description	Section
0	UPPID	uPP Peripheral Identification Register	Section 30.3.1
4h	UPPCR	uPP Peripheral Control Register	Section 30.3.2
8h	UPDLB	uPP Digital Loopback Register	Section 30.3.3
10h	UPCTL	uPP Channel Control Register	Section 30.3.4
14h	UPICR	uPP Interface Configuration Register	Section 30.3.5
18h	UPIVR	uPP Interface Idle Value Register	Section 30.3.6
1Ch	UPTCR	uPP Threshold Configuration Register	Section 30.3.7
20h	UPISR	uPP Interrupt Raw Status Register	Section 30.3.8
24h	UPIER	uPP Interrupt Enabled Status Register	Section 30.3.9
28h	UPIES	uPP Interrupt Enable Set Register	Section 30.3.10
2Ch	UPIEC	uPP Interrupt Enable Clear Register	Section 30.3.11
30h	UPEOI	uPP End-of-Interrupt Register	Section 30.3.12
40h	UPID0	uPP DMA Channel I Descriptor 0 Register	Section 30.3.13
44h	UPID1	uPP DMA Channel I Descriptor 1 Register	Section 30.3.14
48h	UPID2	uPP DMA Channel I Descriptor 2 Register	Section 30.3.15
50h	UPIS0	uPP DMA Channel I Status 0 Register	Section 30.3.16
54h	UPIS1	uPP DMA Channel I Status 1 Register	Section 30.3.17
58h	UPIS2	uPP DMA Channel I Status 2 Register	Section 30.3.18
60h	UPQD0	uPP DMA Channel Q Descriptor 0 Register	Section 30.3.19
64h	UPQD1	uPP DMA Channel Q Descriptor 1 Register	Section 30.3.20
68h	UPQD2	uPP DMA Channel Q Descriptor 2 Register	Section 30.3.21
70h	UPQS0	uPP DMA Channel Q Status 0 Register	Section 30.3.22
74h	UPQS1	uPP DMA Channel Q Status 1 Register	Section 30.3.23
78h	UPQS2	uPP DMA Channel Q Status 2 Register	Section 30.3.24

30.3.1 uPP Peripheral Identification Register (UPPID)

The uPP peripheral identification register (UPPID) reports the revision ID of the uPP peripheral. The UPPID is shown in Figure 30-16 and described in Table 30-11.

Figure 30-16. uPP Peripheral Identification Register (UPPID)



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

Table 30-11. uPP Peripheral Identification Register (UPPID) Field Descriptions

Bit	Field	Value	Description
31-0	REVID	4423 1100h	Peripheral identification number.

Copyright © 2013–2016, Texas Instruments Incorporated

30.3.2 uPP Peripheral Control Register (UPPCR)

The uPP peripheral control register (UPPCR) controls certain peripheral-level configuration settings for the uPP peripheral. Among these are global enable and reset states and rules governing its behavior during CPU emulation halt. This register also reports the current activity state of the uPP internal DMA controller. The UPPCR is shown in Figure 30-17 and described in Table 30-12.

Figure 30-17. uPP Peripheral Control Register (UPPCR)

31	16
Reserved	
R-0	

15	8	7	6	5	4	3	2	1	0
Reserved		DB	Reserved		SWRST	EN	RTEMU	SOFT	FREE
R-0		R-0	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 30-12. uPP Peripheral Control Register (UPPCR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	DB		DMA burst status. Used to poll whether internal DMA is currently active. Writes to this field have no effect.
		0	Internal DMA is idle.
		1	Internal DMA is active.
6-5	Reserved	0	Reserved
4	SWRST		Software reset control. Asserting reset clears internal state machines and prevents device from running. For graceful reset, you should first clear the EN bit and poll the DB bit to make sure the DMA is idle, then assert the SWRST bit.
		0	Peripheral running (out of reset)
		1	Peripheral in reset
3	EN		Peripheral enable control. When transitioning to disabled status, peripheral completes any DMA transactions already in progress before stopping.
		0	Peripheral is disabled.
		1	Peripheral is enabled.
2	RTEMU		Real-time emulation control. When asserted, emulation halts/breakpoints halts pending transactions.
		0	Real-time emulation disabled. Peripheral continues pending transactions while program is halted.
		1	Real-time emulation enabled. Peripheral halts transactions while program is halted.
1	SOFT		Soft stop enable. Must be enabled to allow emulation to halt the peripheral.
		0	Soft stop is disabled.
		1	Soft stop is enabled.
0	FREE		Enable free run. This must be disabled to allow emulation to halt the peripheral.
		0	Free run is disabled.
		1	Free run is enabled.

Copyright © 2013–2016, Texas Instruments Incorporated

30.3.3 uPP Digital Loopback Register (UPDLB)

The uPP digital loopback register (UPDLB) enables or disables the use of internal digital loopback in the uPP peripheral. Internal loopback may be used to transfer data from one uPP interface channel to another when the peripheral is configured in duplex mode. The interface Channel A and Channel B pins do not need to be connected for this operation, but the proper pin multiplexing must still be applied. The UPDLB is shown in Figure 30-18 and described in Table 30-13.

Figure 30-18. uPP Digital Loopback Register (UPDLB)

31	16
Reserved	
R-0	

15	14	13	12	11	0
Reserved		BA	AB	Reserved	
R-0		R/W-0	R/W-0	R-0	

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

Table 30-13. uPP Digital Loopback Register (UPDLB) Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Reserved
13	BA		B-to-A digital loopback control. Assert to enable digital loopback, transmitting from Channel B to Channel A.
		0	Disable B-to-A digital loopback. Allows AB to be asserted.
		1	Enable B-to-A digital loopback. Requires AB = 0 and the MODE bit in the uPP channel control register (UPCTL) to be set to 2h.
12	AB		A-to-B digital loopback control. Set to enable digital loopback, transmitting from Channel A to Channel B.
		0	Disable A-to-B digital loopback. Allows BA to be asserted.
		1	Enable A-to-B digital loopback. Requires BA = 0 and the MODE bit in the uPP channel control register (UPCTL) to be set to 3h.
11-0	Reserved	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

30.3.4 uPP Channel Control Register (UPCTL)

The uPP channel control register (UPCTL) controls uPP interface channel settings. This includes global settings, such as the channel count and data interleave, and channel-specific settings such as bit width and data rate.

The UPCTL also controls data formatting for 9-bit to 15-bit operating modes. For a channel with an N-bit interface, the 16 – N MSBs are written or replaced (in receive or transmit mode, respectively) according to the DPFB and DPFA fields. The uPP peripheral performs no data formatting for 8-bit and 16-bit modes. The UPCTL is shown in Figure 30-19 and described in Table 30-14.

Figure 30-19. uPP Channel Control Register (UPCTL)

31	30 29	28 26	25	24	23	22 21	20 18	17	16
Rsvd	DPFB	DPWB	IWB	DRB	Rsvd	DPFA	DPWA	IWA	DRA
R-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-0

15 8
Reserved
R-0

7 5	4	3	2	1 0
Reserved	DDRDEMUX	SDRTXIL	CHN	MODE
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 30-14. uPP Channel Control Register (UPCTL) Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reserved
30-29	DPFB	0-3h	Channel B data packing format. Applies only to 9-bit to 15-bit modes (IWB = 1 and DPWB != 0).
		0	Right-justified, zero extended
		1h	Right-justified, sign extended
		2h	Left-justified, zero filled
		3h	Reserved
28-26	DPWB	0-7h	Channel B bit width. Applies only if IWB = 1.
		0	No data packing (8-bit or 16-bit case)
		1h	9-bit data format
		2h	10-bit data format
		3h	11-bit data format
		4h	12-bit data format
		5h	13-bit data format
		6h	14-bit data format
		7h	15-bit data format
25	IWB		Channel B interface width. Controls whether Channel B performs 8-bit or 16-bit transactions.
		0	8-bit interface
		1	16-bit interface
24	DRB		Channel B data rate. Controls whether Channel B operates at single or double rate.
		0	Single data rate
		1	Double data rate
23	Reserved	0	Reserved
22-21	DPFA	0-3h	Channel A data packing format. Applies only to 9-bit to 15-bit modes (IWA = 1 and DPWA != 0).
		0	Right-justified, zero extended
		1h	Right-justified, sign extended
		2h	Left-justified, zero filled
		3h	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

Table 30-14. uPP Channel Control Register (UPCTL) Field Descriptions (continued)

Bit	Field	Value	Description
20-18	DPWA	0-7h	Channel A bit width. Applies only if IWA = 1.
		0	No data packing (8-bit or 16-bit case)
		1h	9-bit data format
		2h	10-bit data format
		3h	11-bit data format
		4h	12-bit data format
		5h	13-bit data format
		6h	14-bit data format
		7h	15-bit data format
17	IWA		Channel A interface width. Controls whether Channel A performs 8-bit or 16-bit transactions.
		0	8-bit interface
		1	16-bit interface
16	DRA		Channel A data rate. Controls whether Channel A operates at single or double rate.
		0	Single data rate
		1	Double data rate
15-5	Reserved	0	Reserved
4	DDRDEMUX		Double data rate demultiplexing mode. Only applies when DRA = 1.
		0	Disable. Each peripheral channel is associated with its own DMA channel.
		1	Enable. Both DMA channels service peripheral Channel A. Requires CHN = 0 and MODE = 0 or 1.
3	SDRTXIL		Single data rate transmit interleave mode. Only applies when DRA = 0.
		0	Disable. Each peripheral channel is associated with its own DMA channel.
		1	Enable. Both DMA channels service peripheral Channel A. Requires CHN = 0 and MODE = 1.
2	CHN		Interface channel number. Controls whether one or both interface channels (A, B) are active.
		0	Single channel mode. Only Channel A is active.
		1	Dual channel mode. Channel A and Channel B are both active.
1-0	MODE	0-3h	Operating mode. Controls the direction each active interface channel operates.
		0	All receive mode
		1h	All transmit mode
		2h	Duplex Mode 0. Channel A receives and Channel B transmits. Requires CHN = 1.
		3h	Duplex Mode 1. Channel A transmits and Channel B receives. Requires CHN = 1.

Copyright © 2013–2016, Texas Instruments Incorporated

30.3.5 uPP Interface Configuration Register (UPICR)

The uPP interface configuration register (UPICR) controls the enable state and polarity of each uPP interface channel's pins. The polarity selection applies regardless of channel direction. The signal enable states only apply in either receive or transmit mode, but never both. The UPICR is shown in Figure 30-20 and described in Table 30-15.

NOTE: When initializing the uPP peripheral, the uPP interface configuration register (UPICR) must be programmed using a single, 32-bit write. Writing UPICR fields one-by-one can lead to unexpected results.

Figure 30-20. uPP Interface Configuration Register (UPICR)

31	30	29	28	27			24
Reserved		TRISB	CLKINVB	CLKDIVB			
R-0		R/W-0	R/W-0	R/W-0			
23	22	21	20	19	18	17	16
Reserved		WAITB	ENAB	STARTB	WAITPOLB	ENAPOLB	STARTPOLB
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11			8
Reserved		TRISA	CLKINVA	CLKDIVA			
R-0		R/W-0	R/W-0	R/W-0			
7	6	5	4	3	2	1	0
Reserved		WAITA	ENAA	STARTA	WAITPOLA	ENAPOLA	STARTPOLA
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 30-15. uPP Interface Configuration Register (UPICR) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Reserved
29	TRISB		Channel B high-impedance state. Controls interface Channel B while idle in transmit mode. Only applies when Channel B is configured in transmit mode using the MODE bit in the uPP channel control register (UPCTL).
		0	Channel B drives value from the VALB bit in the uPP interface idle value register (UPIVR) while idle.
		1	Channel B data pins are in a high-impedance state while idle.
28	CLKINVB		Channel B clock inversion. Controls clock signal polarity for interface Channel B.
		0	Clock is not inverted. Channel B signals align on rising edge of clock.
		1	Clock is inverted. Channel B signals align on falling edge of clock.
27-24	CLKDIVB	0-Fh	Clock divisor for Channel B. Only used when interface Channel B is configured in transmit mode using the MODE bit in the uPP channel control register (UPCTL). Applied divisor equals CLKDIVB + 1.
23-22	Reserved	0	Reserved
21	WAITB		Channel B WAIT signal enable. Controls use of WAIT signal for interface Channel B. Only applied when Channel B is configured in transmit mode using the MODE bit in the uPP channel control register (UPCTL). In receive mode, WAIT is always driven low.
		0	WAIT signal is disabled. Channel B ignores WAIT in transmit mode.
		1	WAIT signal is enabled. Channel B honors WAIT in transmit mode.
20	ENAB		Channel B ENABLE Signal Enable. Controls use of ENABLE signal for interface Channel B. Only applied when Channel B is configured in receive mode using the MODE bit in the uPP channel control register (UPCTL). In transmit mode, ENABLE is always driven.
		0	ENABLE signal is disabled. Channel B ignores ENABLE in receive mode.
		1	ENABLE signal is enabled. Channel B honors ENABLE in receive mode.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 30-15. uPP Interface Configuration Register (UPICR) Field Descriptions (continued)

Bit	Field	Value	Description
19	STARTB		Channel B START Signal Enable. Controls use of START signal for interface Channel B. Only applied when Channel B is configured in receive mode using the MODE bit in the uPP channel control register (UPCTL). In transmit mode, STARTB is always driven.
		0	START signal is disabled. Channel B ignores START in receive mode.
		1	START signal is enabled. Channel B honors START in receive mode.
18	WAITPOLB		Channel B WAIT Signal Polarity. Controls polarity of WAIT signal for interface Channel B.
		0	WAIT is active-high for Channel B.
		1	WAIT is active-low for Channel B.
17	ENAPOLB		Channel B ENABLE Signal Polarity. Controls polarity of ENABLE signal for interface Channel B.
		0	ENABLE is active-high for Channel B.
		1	ENABLE is active-low for Channel B.
16	STARTPOLB		Channel B START Signal Polarity. Controls polarity of START signal for interface Channel B.
		0	START is active-high for Channel B.
		1	START is active-low for Channel B.
15-14	Reserved	0	Reserved
13	TRISA		Channel A high-impedance state. Controls interface Channel A while idle in transmit mode. Only applies when Channel A is configured in transmit mode using the MODE bit in the uPP channel control register (UPCTL).
		0	Channel A drives value from the VALA bit in the uPP interface idle value register (UPIVR) while idle.
		1	Channel A data pins are in a high-impedance state while idle.
12	CLKINVA		Channel A clock inversion. Controls clock signal polarity for interface Channel A.
		0	Clock is not inverted. Channel A signals align on rising edge of clock.
		1	Clock is inverted. Channel A signals align on falling edge of clock.
11-8	CLKDIVA	0-Fh	Clock divisor for Channel A. Only used when interface Channel A is configured in transmit mode using the MODE bit in the uPP channel control register (UPCTL). Applied divisor equals CLKDIVA + 1.
7-6	Reserved	0	Reserved
5	WAITA		Channel A WAIT signal enable. Controls use of WAIT signal for interface Channel A. Only applied when Channel A is configured in transmit mode using the MODE bit in the uPP channel control register (UPCTL). In receive mode, WAIT is always driven low.
		0	WAIT signal is disabled. Channel A ignores WAIT in transmit mode.
		1	WAIT signal is enabled. Channel A honors WAIT in transmit mode.
4	ENAA		Channel A ENABLE Signal Enable. Controls use of ENABLE signal for interface Channel B. Only applied when Channel A is configured in receive mode using the MODE bit in the uPP channel control register (UPCTL). In transmit mode, ENABLE is always driven.
		0	ENABLE signal is disabled. Channel A ignores ENABLE in receive mode.
		1	ENABLE signal is enabled. Channel A honors ENABLE in receive mode.
3	STARTA		Channel A START Signal Enable. Controls use of START signal for interface Channel A. Only applied when Channel A is configured in receive mode using the MODE bit in the uPP channel control register (UPCTL). In transmit mode, STARTA is always driven.
		0	START signal is disabled. Channel A ignores START in receive mode.
		1	START signal is enabled. Channel A honors START in receive mode.
2	WAITPOLA		Channel A WAIT Signal Polarity. Controls polarity of WAIT signal for interface Channel A.
		0	WAIT is active-high for Channel A.
		1	WAIT is active-low for Channel A.
1	ENAPOLA		Channel A ENABLE Signal Polarity. Controls polarity of ENABLE signal for interface Channel A.
		0	ENABLE is active-high for Channel A.
		1	ENABLE is active-low for Channel A.
0	STARTPOLA		Channel A START Signal Polarity. Controls polarity of START signal for interface Channel A.
		0	START is active-high for Channel A.
		1	START is active-low for Channel A.

Copyright © 2013–2016, Texas Instruments Incorporated

30.3.6 uPP Interface Idle Value Register (UPIVR)

The uPP interface idle value register (UPIVR) controls the value that each interface channel transmits while idle. Note that this only applies when the associated channel is configured in transmit mode and also depends on the value of the TRISA and TRISB bits in the uPP interface configuration register (UPICR). The UPIVR is shown in Figure 30-21 and described in Table 30-16.

Figure 30-21. uPP Interface Idle Value Register (UPIVR)

31		16
	VALB	
	R/W-0	
15		0
	VALA	
	R/W-0	

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

Table 30-16. uPP Interface Idle Value Register (UPIVR) Field Descriptions

Bit	Field	Value	Description
31-16	VALB	0-FFFFh	Channel B idle value. Sets idle value for interface Channel B. This value is output on the Channel B data pins when the channel is idle, configured in transmit mode, and the TRISB bit in the uPP interface configuration register (UPICR) is cleared to 0.
15-0	VALA	0-FFFFh	Channel A idle value. Sets idle value for interface Channel A. This value is output on the Channel A data pins when the channel is idle, configured in transmit mode, and the TRISA bit in the uPP interface configuration register (UPICR) is cleared to 0.

Copyright © 2013–2016, Texas Instruments Incorporated

30.3.7 uPP Threshold Configuration Register (UPTCR)

The uPP threshold configuration register (UPTCR) controls the transmit threshold for each interface channel and the read threshold for each DMA channel. Using larger thresholds can decrease internal bus traffic and increase performance, especially when the uPP is operating in a loaded system. The UPTCR is shown in Figure 30-22 and described in Table 30-17.

Figure 30-22. uPP Threshold Configuration Register (UPTCR)

31	26	25	24	23	18	17	16
Reserved		TXSIZEB		Reserved		TXSIZEA	
R-0		R/W-0		R-0		R/W-0	

15	10	9	8	7	2	1	0
Reserved		RDSIZEQ		Reserved		RDSIZEI	
R-0		R/W-0		R-0		R/W-0	

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

Table 30-17. uPP Threshold Configuration Register (UPTCR) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	Reserved
25-24	TXSIZEB	0-3h	Transmit threshold for Channel B. Controls the number of bytes that interface Channel B waits before beginning transmission. Only applies when Channel B is configured in transmit mode using the MODE bit in the uPP channel control register (UPCTL).
		0	64 bytes
		1	128 bytes (requires DMA descriptor byte count greater than 64)
		2h	Reserved
		3h	256 bytes (requires DMA descriptor byte count greater than 192)
23-18	Reserved	0	Reserved
17-16	TXSIZEA	0-3h	Transmit threshold for Channel A. Controls the number of bytes that interface Channel A waits before beginning transmission. Only applies when Channel A is configured in transmit mode using the MODE bit in the uPP channel control register (UPCTL).
		0	64 bytes
		1	128 bytes (requires DMA descriptor byte count greater than 64)
		2h	Reserved
		3h	256 bytes (requires DMA descriptor byte count greater than 192)
15-10	Reserved	0	Reserved
9-8	RDSIZEQ	0-3h	Read threshold for DMA Channel Q. Controls burst size for DMA Channel Q.
		0	64 bytes
		1	128 bytes
		2h	Reserved
		3h	256 bytes
7-2	Reserved	0	Reserved
1-0	RDSIZEI	0-3h	Read threshold for DMA Channel I. Controls burst size for DMA Channel I.
		0	64 bytes
		1	128 bytes
		2h	Reserved
		3h	256 bytes

Copyright © 2013–2016, Texas Instruments Incorporated

30.3.8 uPP Interrupt Raw Status Register (UPISR)

The uPP interrupt raw status register (UPISR) reports the raw interrupt status for various conditions. Each status bit reads 1 when the associated event occurs, even if that interrupt event is disabled in the uPP interrupt enable clear register (UPIEC). Writing 1 to any bit simulates the associated interrupt event; writing 0 has no effect. The UPISR is shown in Figure 30-23 and described in Table 30-18.

Figure 30-23. uPP Interrupt Raw Status Register (UPISR)

31	16
Reserved	
R-0	

15	13	12	11	10	9	8
Reserved		EOLQ	EOWQ	ERRQ	UORQ	DPEQ
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	5	4	3	2	1	0
Reserved		EOLI	EOWI	ERRI	UORI	DPEI
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 30-18. uPP Interrupt Raw Status Register (UPISR) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reserved
12	EOLQ		Reports raw interrupt status for end-of-line condition (EOL) on DMA Channel Q.
		0	No EOL occurred
		1	EOL occurred
11	EOWQ		Reports raw interrupt status for end-of-window condition (EOW) on DMA Channel Q.
		0	No EOW occurred
		1	EOW occurred
10	ERRQ		Reports raw interrupt status for internal bus error condition (ERR) on DMA Channel Q.
		0	No error occurred
		1	Error occurred
9	UORQ		Reports raw interrupt status for underrun or overflow condition (UOR) on DMA Channel Q.
		0	No underrun or overflow occurred
		1	Underrun or overflow occurred
8	DPEQ		Reports raw interrupt status for programming error condition (DPE) on DMA Channel Q.
		0	No error occurred
		1	Error occurred
7-5	Reserved	0	Reserved
4	EOLI		Reports raw interrupt status for end-of-line condition (EOL) on DMA Channel I.
		0	No EOL occurred
		1	EOL occurred
3	EOWI		Reports raw interrupt status for end-of-window condition (EOW) on DMA Channel I.
		0	No EOW occurred
		1	EOW occurred
2	ERRI		Reports raw interrupt status for internal bus error condition (ERR) on DMA Channel I.
		0	No error occurred
		1	Error occurred
1	UORI		Reports raw interrupt status for underrun or overflow condition (UOR) on DMA Channel I.
		0	No underrun or overflow occurred
		1	Underrun or overflow occurred

Copyright © 2013–2016, Texas Instruments Incorporated

Table 30-18. uPP Interrupt Raw Status Register (UPISR) Field Descriptions (continued)

Bit	Field	Value	Description
0	DPEI		Reports raw interrupt status for programming error condition (DPE) on DMA Channel I.
		0	No error occurred
		1	Error occurred

Copyright © 2013–2016, Texas Instruments Incorporated

30.3.9 uPP Interrupt Enabled Status Register (UPIER)

The uPP interrupt enabled status register (UPIER) reports the enabled interrupt status for various conditions. Each status bit reads 1, if the associated event occurs while that event is enabled in the uPP interrupt enabled set register (UPIES). If the interrupt event is disabled in the uPP interrupt enable clear register (UPIEC), the associated status bit always reads 0. Writing 1 to any bit clears the associated interrupt event; writing 0 has no effect. The UPIER is shown in Figure 30-24 and described in Table 30-19.

Figure 30-24. uPP Interrupt Enabled Status Register (UPIER)

31	16
Reserved	
R-0	

15	13	12	11	10	9	8
Reserved		EOLQ	EOWQ	ERRQ	UORQ	DPEQ
R-0		R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0

7	5	4	3	2	1	0
Reserved		EOLI	EOWI	ERRI	UORI	DPEI
R-0		R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear bit (writing 0 has no effect); -*n* = value after reset

Table 30-19. uPP Interrupt Enabled Status Register (UPIER) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reserved
12	EOLQ		Interrupt Status for Channel Q End-of-Line. Reports enabled interrupt status for end-of-line condition (EOL) on DMA Channel Q.
		0	No EOL
		1	EOL occurred
11	EOWQ		Interrupt Status for Channel Q End-of-Window. Reports enabled interrupt status for end-of-window condition (EOW) on DMA Channel Q.
		0	No EOW
		1	EOW occurred
10	ERRQ		Interrupt Status for Channel Q Error. Reports enabled interrupt status for internal bus error condition on DMA Channel Q.
		0	No error
		1	Error occurred
9	UORQ		Interrupt Status for Channel Q Underrun/Overflow condition. Reports enabled interrupt status for underrun or overflow condition on DMA Channel Q.
		0	No underrun or overflow
		1	Underrun or overflow occurred
8	DPEQ		Interrupt Status for Channel Q Programming Error. Reports enabled interrupt status for programming error condition on DMA Channel Q.
		0	No error
		1	Error occurred
7-5	Reserved	0	Reserved
4	EOLI		Interrupt Status for Channel I End-of-Line. Reports enabled interrupt status for end-of-line condition (EOL) on DMA Channel I.
		0	No EOL
		1	EOW occurred
3	EOWI		Interrupt Status for Channel I End-of-Window. Reports enabled interrupt status for end-of-window condition (EOW) on DMA Channel I.
		0	No EOW
		1	EOW occurred

Copyright © 2013–2016, Texas Instruments Incorporated

Table 30-19. uPP Interrupt Enabled Status Register (UPIER) Field Descriptions (continued)

Bit	Field	Value	Description
2	ERRI		Interrupt Status for Channel I Error. Reports enabled interrupt status for internal bus error condition on DMA Channel I.
		0	No error
		1	Error occurred
1	UORI		Interrupt Status for Channel I Underrun/Overflow condition. Reports enabled interrupt status for underrun or overflow condition on DMA Channel I.
		0	No underrun or overflow
		1	Underrun or overflow occurred
0	DPEI		Interrupt Status for Channel I Programming Error. Reports enabled interrupt status for programming error condition on DMA Channel I.
		0	No error
		1	Error occurred

Copyright © 2013–2016, Texas Instruments Incorporated

30.3.10 uPP Interrupt Enable Set Register (UPIES)

The uPP interrupt enabled set register (UPIES) controls whether individual interrupt events generate a CPU interrupt. Writing 1 to any bit enables CPU interrupt generation for the associated uPP event; writing 0 has no effect.

Reads from both UPIES and the uPP interrupt enable clear register (UPIEC) access the same internal interrupt enable register, and a value of 1 indicates that the corresponding interrupt is enabled. The UPIES is shown in Figure 30-25 and described in Table 30-20.

Figure 30-25. uPP Interrupt Enable Set Register (UPIES)

31	16
Reserved	
R-0	

15	13	12	11	10	9	8
Reserved		EOLQ	EOWQ	ERRQ	UORQ	DPEQ
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	5	4	3	2	1	0
Reserved		EOLI	EOWI	ERRI	UORI	DPEI
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 30-20. uPP Interrupt Enable Set Register (UPIES) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reserved
12	EOLQ		Interrupt Enable Set for Channel Q End-of-Line. Reports interrupt enable for end-of-line condition (EOL) on DMA Channel Q.
		0	Read: EOL interrupt is disabled. Write: no effect.
		1	Read: EOL interrupt is enabled. Write: enable EOL interrupt.
11	EOWQ		Interrupt Enable Set for Channel Q End-of-Window. Reports interrupt enable for end-of-window condition (EOW) on DMA Channel Q.
		0	Read: EOW interrupt is disabled. Write: no effect.
		1	Read: EOW interrupt is enabled. Write: enable EOW interrupt.
10	ERRQ		Interrupt Enable Set for Channel Q Error. Reports interrupt enable for internal bus error condition on DMA Channel Q.
		0	Read: Error interrupt is disabled. Write: no effect.
		1	Read: Error interrupt is enabled. Write: enable ERR interrupt.
9	UORQ		Interrupt Enable Set for Channel Q Underrun/Overflow condition. Reports interrupt enable for underrun or overflow condition on DMA Channel Q.
		0	Read: Underrun or overflow interrupt is disabled. Write: no effect.
		1	Read: Underrun or overflow interrupt is enabled. Write: enable UOR interrupt.
8	DPEQ		Interrupt Enable Set for Channel Q Programming Error. Reports interrupt enable for programming error condition on DMA Channel Q.
		0	Read: Programming error interrupt is disabled. Write: no effect.
		1	Read: Programming error interrupt is enabled. Write: enable DPE interrupt.
7-5	Reserved	0	Reserved
4	EOLI		Interrupt Enable Set for Channel I End-of-Line. Reports interrupt enable for end-of-line condition (EOL) on DMA Channel I.
		0	Read: EOL interrupt is disabled. Write: no effect.
		1	Read: EOL interrupt is enabled. Write: enable EOL interrupt.
3	EOWI		Interrupt Enable Set for Channel I End-of-Window. Reports interrupt enable for end-of-window condition (EOW) on DMA Channel I.
		0	Read: EOW interrupt is disabled. Write: no effect.
		1	Read: EOW interrupt is enabled. Write: enable EOW interrupt.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 30-20. uPP Interrupt Enable Set Register (UPIES) Field Descriptions (continued)

Bit	Field	Value	Description
2	ERRI		Interrupt Enable Set for Channel I Error. Reports interrupt enable for internal bus error condition on DMA Channel I.
		0	Read: Error interrupt is disabled. Write: no effect.
		1	Read: Error interrupt is enabled. Write: enable ERR interrupt.
1	UORI		Interrupt Enable Set for Channel I Underrun/Overflow condition. Reports interrupt enable for underrun or overflow condition on DMA Channel I.
		0	Read: Underrun or overflow interrupt is disabled. Write: no effect.
		1	Read: Underrun or overflow interrupt is enabled. Write: enable UOR interrupt.
0	DPEI		Interrupt Enable Set for Channel I Programming Error. Reports interrupt enable for programming error condition on DMA Channel I.
		0	Read: Programming error interrupt is disabled. Write: no effect.
		1	Read: Programming error interrupt is enabled. Write: enable DPE interrupt.

Copyright © 2013–2016, Texas Instruments Incorporated

30.3.11 uPP Interrupt Enable Clear Register (UPIEC)

The uPP interrupt enable clear register (UPIEC) controls whether individual interrupt events generate a CPU interrupt. Writing 1 to any bit disables CPU interrupt generation for the associated uPP event; writing 0 has no effect.

Reads from both UPIEC and the uPP interrupt enabled set register (UPIES) access the same internal interrupt enable register, and a value of 1 indicates that the corresponding interrupt is enabled. The UPIEC is shown in Figure 30-26 and described in Table 30-21.

Figure 30-26. uPP Interrupt Enable Clear Register (UPIEC)

31	16
Reserved	
R-0	

15	13	12	11	10	9	8
Reserved		EOLQ	EOWQ	ERRQ	UORQ	DPEQ
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	5	4	3	2	1	0
Reserved		EOLI	EOWI	ERRI	UORI	DPEI
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 30-21. uPP Interrupt Enable Clear Register (UPIEC) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reserved
12	EOLQ		Interrupt Enable Clear for Channel Q End-of-Line. Reports interrupt enable for end-of-line condition (EOL) on DMA Channel Q.
		0	Read: EOL interrupt is disabled. Write: no effect.
		1	Read: EOL interrupt is enabled. Write: disable EOL interrupt
11	EOWQ		Interrupt Enable Clear for Channel Q End-of-Window. Reports interrupt enable for end-of-window condition (EOW) on DMA Channel Q.
		0	Read: EOW interrupt is disabled. Write: no effect.
		1	Read: EOW interrupt is enabled. Write: disable EOW interrupt
10	ERRQ		Interrupt Enable Clear for Channel Q Error. Reports interrupt enable for internal bus error condition on DMA Channel Q.
		0	Read: Error interrupt is disabled. Write: no effect.
		1	Read: Error interrupt is enabled. Write: disable ERR interrupt.
9	UORQ		Interrupt Enable Clear for Channel Q Underrun/Overflow condition. Reports interrupt enable for underrun or overflow condition on DMA Channel Q.
		0	Read: Underrun or overflow interrupt is disabled. Write: no effect.
		1	Read: Underrun or overflow interrupt is enabled. Write: disable UOR interrupt.
8	DPEQ		Interrupt Enable Clear for Channel Q Programming Error. Reports interrupt enable for programming error condition on DMA Channel Q.
		0	Read: Programming error interrupt is disabled. Write: no effect.
		1	Read: Programming error interrupt is enabled. Write: disable DPE interrupt.
7-5	Reserved	0	Reserved
4	EOLI		Interrupt Enable Clear for Channel I End-of-Line. Reports interrupt enable for end-of-line condition (EOL) on DMA Channel I.
		0	Read: EOL interrupt is disabled. Write: no effect.
		1	Read: EOL interrupt is enabled. Write: disable EOL interrupt.
3	EOWI		Interrupt Enable Clear for Channel I End-of-Window. Reports interrupt enable for end-of-window condition (EOW) on DMA Channel I.
		0	Read: EOW interrupt is disabled. Write: no effect.
		1	Read: EOW interrupt is enabled. Write: disable EOW interrupt.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 30-21. uPP Interrupt Enable Clear Register (UPIEC) Field Descriptions (continued)

Bit	Field	Value	Description
2	ERRI		Interrupt Enable Clear for Channel I Error. Reports interrupt enable for internal bus error condition on DMA Channel I.
		0	Read: Error interrupt is disabled. Write: no effect.
		1	Read: Error interrupt is enabled. Write: disable ERR interrupt.
1	UORI		Interrupt Enable Clear for Channel I Underrun/Overflow condition. Reports interrupt enable for underrun or overflow condition on DMA Channel I.
		0	Read: Underrun or overflow interrupt is disabled. Write: no effect.
		1	Read: Underrun or overflow interrupt is enabled. Write: disable UOR interrupt.
0	DPEI		Interrupt Enable Clear for Channel I Programming Error. Reports interrupt enable for programming error condition on DMA Channel I.
		0	Read: Programming error interrupt is disabled. Write: no effect.
		1	Read: Programming error interrupt is enabled. Write: disable DPE interrupt.

Copyright © 2013–2016, Texas Instruments Incorporated

30.3.12 uPP End of Interrupt Register (UPEOI)

The uPP end of interrupt register (UPEOI) acknowledges CPU interrupts generated from uPP events. The EOI bit field must be written with 00h at the end of the interrupt service routine (ISR) that handles uPP interrupts. Until this acknowledgement occurs, no uPP event can generate another CPU interrupt. The UPEOI is shown in Figure 30-27 and described in Table 30-22.

Figure 30-27. uPP End of Interrupt Register (UPEOI)

31	16
Reserved	
R-0	

15	8	7	0
Reserved		EOI	
R-0		R/W-0	

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

Table 30-22. uPP End of Interrupt Register (UPEOI) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	EOI	0-FFh	End of interrupt value. Write 00h after uPP interrupt to allow interrupt generation from subsequent uPP events.

30.3.13 uPP DMA Channel I Descriptor 0 Register (UPID0)

The uPP DMA channel I descriptor 0 register (UPID0) programs the starting address of the data buffer, or window, for DMA Channel I. The address is programmed by writing a 32-bit value to the entire register. Note that the 3 lower bits are read-only and always equal 0, so that data buffers are properly aligned in memory. The UPID0 is shown in Figure 30-28 and described in Table 30-23.

Figure 30-28. uPP DMA Channel I Descriptor 0 Register (UPID0)

31	16
ADDRH	
R/W-0	

15	3	2	0
ADDRH		ADDR	
R/W-0		R-0	

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

Table 30-23. uPP DMA Channel I Descriptor 0 Register (UPID0) Field Descriptions

Bit	Field	Value	Description
31-3	ADDRH	0-1FFF FFFFh	Window Address MSBs. Sets the 29 most-significant bits of starting address for DMA Channel I window.
2-0	ADDR	0	Window Address LSBs. Forces window address to align to multiple of 8 bytes (64-bit buffer alignment).

Copyright © 2013–2016, Texas Instruments Incorporated

30.3.14 uPP DMA Channel I Descriptor 1 Register (UPID1)

The uPP DMA channel I descriptor 1 register (UPID1) programs the line count per window and byte count per line for DMA Channel I. The line count (LNCNT) may be set to any number from 1 to 65 535 (FFFFh), but must not be cleared to 0. The byte count (BCNT) may only be set to an even number. For a simple transfer, LNCNT may be set to 1, and BCNT may be set to N >> 1, where N is the desired byte count of the entire DMA transfer. Note that the lower bit is read-only and is always equal 0, so that N is an even number. The UPID1 is shown in Figure 30-29 and described in Table 30-24.

Figure 30-29. uPP DMA Channel I Descriptor 1 Register (UPID1)

31	16
LNCNT	
R/W-0	

15	1	0
BCNTH		BCNT
R/W-0		R-0

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

Table 30-24. uPP DMA Channel I Descriptor 1 Register (UPID1) Field Descriptions

Bit	Field	Value	Description
31-16	LNCNT	1-FFFFh	Line Count. Sets the number of lines in the DMA Channel I window.
		0	Invalid value
15-1	BCNTH	1-7FFFh	Byte Count MSBs. Sets the 15 most-significant bits of the number of bytes per line in the DMA Channel I window.
		0	Invalid value
0	BCNT	0	Byte Count LSB. Forces the number of bytes per line to an even value (multiple of 2 bytes).

30.3.15 uPP DMA Channel I Descriptor 2 Register (UPID2)

The uPP DMA channel I descriptor 2 register (UPID2) programs the offset address between lines within the DMA Channel I window. Note that the 3 lower bits are read-only and always equal 0, so that the line offset address is aligned to a multiple of 8 bytes, similar to the window address in UPID0. Writing a value of 0 to UPID2 effectively repeats the same (first) line *UPID1.LNCNT* times. The UPID2 is shown in Figure 30-30 and described in Table 30-25.

Figure 30-30. uPP DMA Channel I Descriptor 2 Register (UPID2)

31	16
Reserved	
R-0	

15	3	2	0
LNOFFSETH		LNOFFSET	
R/W-0		R-0	

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

Table 30-25. uPP DMA Channel I Descriptor 2 Register (UPID2) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-3	LNOFFSETH	0-1FFFh	Line Offset Address MSBs. Sets the 13 most-significant bits of the offset address (in bytes) between lines in the DMA Channel I window. This is a signed 2s-complement value.
2-0	LNOFFSET	0	Line Offset Address LSBs. Forces the line offset address to align to a multiple of 8 bytes (64-bit alignment).

Copyright © 2013–2016, Texas Instruments Incorporated

30.3.16 uPP DMA Channel I Status 0 Register (UPIS0)

The uPP DMA channel I status 0 register (UPIS0) reports the current address of the DMA Channel I transfer. The UPIS0 is shown in Figure 30-31 and described in Table 30-26.

Figure 30-31. uPP DMA Channel I Status 0 Register (UPIS0)

31	0
ADDR	
R-0	

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

Table 30-26. uPP DMA Channel I Status 0 Register (UPIS0) Field Descriptions

Bit	Field	Value	Description
31-0	ADDR	0-FFFF FFFFh	DMA Current Address. Reports the current address of the DMA Channel I transfer.

30.3.17 uPP DMA Channel I Status 1 Register (UPIS1)

The uPP DMA channel I status 1 register (UPIS1) reports the current line number and the byte position within the current line of the DMA Channel I transfer. The UPIS1 is shown in Figure 30-32 and described in Table 30-27.

Figure 30-32. uPP DMA Channel I Status 1 Register (UPIS1)

31	16
LNCNT	
R-0	

15	0
BCNT	
R-0	

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

Table 30-27. uPP DMA Channel I Status 1 Register (UPIS1) Field Descriptions

Bit	Field	Value	Description
31-16	LNCNT	0-FFFFh	DMA Current Line Number. Reports the current line number of the DMA Channel I transfer.
15-0	BCNT	0-FFFFh	DMA Byte Number. Reports the current byte position within the current line of the DMA Channel I transfer.

Copyright © 2013–2016, Texas Instruments Incorporated

30.3.18 uPP DMA Channel I Status 2 Register (UPIS2)

The uPP DMA channel I status register 2 (UPIS2) reports the status of the current DMA Channel I transfer. The PEND bit is used to determine when a new transfer may be programmed into the uPP DMA channel I descriptor registers (UPID*n*). The UPIS2 is shown in Figure 30-33 and described in Table 30-28.

Figure 30-33. uPP DMA Channel I Status 2 Register (UPIS2)

31	16
Reserved	
R-0	

15	8	7	4	3	2	1	0
Reserved		WM		Reserved		PEND	ACT
R-0		R-0		R-0		R-0	R-0

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

Table 30-28. uPP DMA Channel I Status 2 Register (UPIS2) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-4	WM	0-Fh	DMA Watermark. When the associated interface channel operates in receive mode, this field records the maximum FIFO block occupancy reached during any transaction. When the associated interface channel operates in transmit mode, this field records the FIFO block emptiness and is overwritten every uPP interface clock cycle.
3-2	Reserved	0	Reserved
1	PEND		DMA Transfer Pending. Reports whether another DMA transfer is pending for DMA Channel I. This field must be low before another transfer may be programmed.
		0	No transfer pending. Channel I descriptors may be written.
		1	Transfer pending. Channel I descriptors may not be written.
0	ACT		DMA Active. Reports the current status of DMA Channel I. This field should not be used to determine whether the DMA Channel I descriptors are programmable; use the PEND bit instead.
		0	DMA is inactive.
		1	DMA is active.

Copyright © 2013–2016, Texas Instruments Incorporated

30.3.19 uPP DMA Channel Q Descriptor 0 Register (UPQD0)

The uPP DMA channel Q descriptor 0 register (UPQD0) programs the starting address of the data buffer, or window, for DMA Channel Q. The address can be programmed by simply writing a 32-bit value to the entire register. Note that the 3 lower bits are read-only and always equal to 0, so that data buffers must be properly aligned in memory. The UPQD0 is shown in Figure 30-34 and described in Table 30-29.

Figure 30-34. uPP DMA Channel Q Descriptor 0 Register (UPQD0)

31	16
ADDRH	
R/W-0	

15	3	2	0
ADDRH		ADDR	
R/W-0		R-0	

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

Table 30-29. uPP DMA Channel Q Descriptor 0 Register (UPQD0) Field Descriptions

Bit	Field	Value	Description
31-3	ADDRH	0-1FFF FFFFh	Window Address MSBs. Sets the 29 most-significant bits of starting address for DMA Channel Q window.
2-0	ADDR	0	Window Address LSBs. Forces window address to align to multiple of 8 bytes (64-bit buffer alignment).

30.3.20 uPP DMA Channel Q Descriptor 1 Register (UPQD1)

The uPP DMA channel Q descriptor 1 register (UPQD1) programs the line count per window and byte count per line for DMA Channel Q. The line count (LNCNT) may be set to any number from 1 to 65 535 (FFFFh), but must not be cleared to 0. The byte count (BCNT) may only be set to an even number. For a simple transfer, LNCNT may be set to 1, and BCNT may be set to N >> 1, where N is the desired byte count of the entire DMA transfer. Note that the lower bit is read-only and is always equal 0, so that N is an even number. The UPQD1 is shown in Figure 30-35 and described in Table 30-30.

Figure 30-35. uPP DMA Channel Q Descriptor 1 Register (UPQD1)

31	16
LNCNT	
R/W-0	

15	1	0
BCNTH		BCNT
R/W-0		R-0

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

Table 30-30. uPP DMA Channel Q Descriptor 1 Register (UPQD1) Field Descriptions

Bit	Field	Value	Description
31-16	LNCNT	1-FFFFh	Line Count. Sets the number of lines in the DMA Channel Q window.
		0	Invalid value
15-1	BCNTH	1-7FFFh	Byte Count MSBs. Sets the 15 most-significant bits of the number of bytes per line in the DMA Channel Q window.
		0	Invalid value
0	BCNT	0	Byte Count LSB. Forces the number of bytes per line to an even value (multiple of 2 bytes).

Copyright © 2013–2016, Texas Instruments Incorporated

30.3.21 uPP DMA Channel Q Descriptor 2 Register (UPQD2)

The uPP DMA channel Q descriptor 2 register (UPQD2) programs the offset address between lines within the DMA Channel Q window. Note that the 3 lower bits are read-only and always equal 0, so that the line offset address is aligned to a multiple of 8 bytes, similar to the window address in UPQD0. Writing a value of 0 to UPQD2 effectively repeats the same (first) line *UPQD1.LNCNT* times. The UPQD2 is shown in Figure 30-36 and described in Table 30-31.

Figure 30-36. uPP DMA Channel Q Descriptor 2 Register (UPID2)

31	16
Reserved	
R-0	

15	3	2	0
LNOFFSETH		LNOFFSET	
R/W-0		R-0	

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

Table 30-31. uPP DMA Channel Q Descriptor 2 Register (UPID2) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-3	LNOFFSETH	0-1FFFh	Line Offset Address MSBs. Sets the 13 most-significant bits of the offset address (in bytes) between lines in the DMA Channel Q window. This is a signed 2s-complement value.
2-0	LNOFFSET	0	Line Offset Address LSBs. Forces the line offset address to align to a multiple of 8 bytes (64-bit alignment).

Copyright © 2013–2016, Texas Instruments Incorporated

30.3.22 uPP DMA Channel Q Status 0 Register (UPQS0)

The uPP DMA channel Q status 0 register (UPQS0) reports the current address of the DMA Channel Q transfer. The UPQS0 is shown in Figure 30-37 and described in Table 30-32.

Figure 30-37. uPP DMA Channel Q Status 0 Register (UPQS0)

31	0
ADDR	
R-0	

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

Table 30-32. uPP DMA Channel Q Status 0 Register (UPQS0) Field Descriptions

Bit	Field	Value	Description
31-0	ADDR	0-FFFF FFFFh	DMA Current Address. Reports the current address of the DMA Channel Q transfer.

30.3.23 uPP DMA Channel Q Status 1 Register (UPQS1)

The uPP DMA channel Q status 1 register (UPQS1) reports the current line number and the byte position within the current line of the DMA Channel Q transfer. The UPQS1 is shown in Figure 30-38 and described in Table 30-33.

Figure 30-38. uPP DMA Channel Q Status 1 Register (UPQS1)

31	16
LNCNT	
R-0	

15	0
BCNT	
R-0	

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

Table 30-33. uPP DMA Channel Q Status 1 Register (UPQS1) Field Descriptions

Bit	Field	Value	Description
31-16	LNCNT	0-FFFFh	DMA Current Line Number. Reports the current line number of the DMA Channel Q transfer.
15-0	BCNT	0-FFFFh	DMA Byte Number. Reports the current byte position within the current line of the DMA Channel Q transfer.

Copyright © 2013–2016, Texas Instruments Incorporated

30.3.24 uPP DMA Channel Q Status 2 Register (UPQS2)

The uPP DMA channel Q status 2 register (UPQS2) reports the status of the current DMA Channel Q transfer. The PEND bit is used to determine when a new transfer may be programmed into the into the uPP DMA channel Q descriptor registers (UPQD*n*). The UPQS2 is shown in Figure 30-39 and described in Table 30-34.

Figure 30-39. uPP DMA Channel Q Status 2 Register (UPQS2)

31	16
Reserved	
R-0	

15	8	7	4	3	2	1	0
Reserved		WM		Reserved		PEND	ACT
R-0		R-0		R-0		R-0	R-0

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

Table 30-34. uPP DMA Channel Q Status 2 Register (UPQS2) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-4	WM	0-Fh	DMA Watermark. When the associated interface channel operates in receive mode, this field records the maximum FIFO block occupancy reached during any transaction. When the associated interface channel operates in transmit mode, this field records the FIFO block emptiness and is overwritten every uPP interface clock cycle.
3-2	Reserved	0	Reserved
1	PEND		DMA Transfer Pending. Reports whether another DMA transfer is pending for DMA Channel Q. This field must be low before another transfer may be programmed.
		0	No transfer pending. Channel Q descriptors may be written.
		1	Transfer pending. Channel Q descriptors may not be written.
0	ACT		DMA Active. Reports the current status of DMA Channel Q. This field should not be used to determine whether the DMA Channel Q descriptors are programmable; use the PEND bit instead.
		0	DMA is inactive.
		1	DMA is active.

Copyright © 2013–2016, Texas Instruments Incorporated

Universal Serial Bus 2.0 (USB) Controller

This chapter describes the universal serial bus (USB) controller.

Topic **Page**

31.1 Introduction **1374**
31.2 Architecture **1375**
31.3 Use Cases **1441**
31.4 Registers **1453**

Copyright © 2013–2016, Texas Instruments Incorporated

31.1 Introduction

The controller complies with the USB 2.0 standard high-speed and full-speed functions and low-speed, full-speed, and high-speed limited host mode operations. It also includes support for the Session Request and Host Negotiation Protocols used in point-to-point communications, details of which are given in the USB On-The-Go supplement to the USB 2.0 specification. In addition, the four test modes for high-speed operation described in the USB 2.0 specification are supported. It also allows options that allow the USB controller to be forced into full-speed mode, high-speed mode, or host mode that may be used for debug purposes.

31.1.1 Purpose of the Peripheral

The USB controller provides a low-cost connectivity solution for consumer portable devices by providing a mechanism for data transfer between USB devices up to 480 Mbps. Its support for a dual-role feature allows for additional versatility supporting operation capability as a host or peripheral.

31.1.2 Features

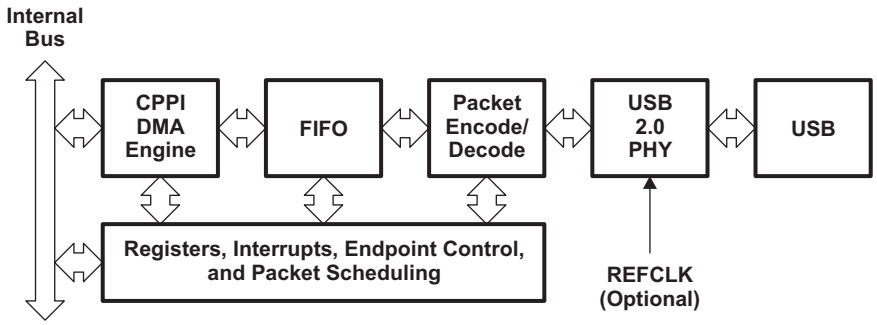
The USB has the following features:

- Operating as a host, it complies with the USB 2.0 standard for high-speed (480 Mbps), full-speed (12 Mbps), and low-speed (1.5 Mbps) operations with a peripheral
- Operating as a peripheral, it complies with the USB 2.0 standard for high-speed (480 Mbps) and full-speed (12 Mbps) operation with a host.
- Supports USB extensions for Session Request (SRP) and Host Negotiation (HNP) – OTG
- Supports 4 simultaneous RX and TX endpoints, in addition to control endpoint, more devices can be supported by dynamically switching endpoints states
- Each endpoint (other than endpoint 0) can support all transfer types (control, bulk, interrupt, and isochronous)
- Includes a 4K endpoint FIFO RAM, and supports programmable FIFO sizes
- External 5V power supply for VBUS, when operating as host, enabled directly by the USB controller through a dedicated signal
- Includes a DMA controller that supports 4 TX and 4 RX DMA channels
- Includes four types of Communications Port Programming Interface (CPPI) 4.1 DMA compliant transfer modes, Transparent, Generic RNDIS, RNDIS, and Linux CDC mode of DMA for accelerating RNDIS type protocols using short packet termination over USB.
- DMA supports single data transfer size up to 4Mbytes

31.1.3 Functional Block Diagram

The USB functional block diagram is shown in Figure 31-1.

Figure 31-1. Functional Block Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

31.1.4 Industry Standard(s) Compliance Statement

This device conforms to USB 2.0 Specification.

31.2 Architecture

31.2.1 Clock Control

Figure 31-2 shows the clock connections for the USB2.0 module. Note that there is no built-in oscillator. The USB2.0 subsystem requires a reference clock for its internal PLL. This reference clock can be sourced from either the USB_REFCLKIN pin or from the AUXCLK of the system PLL. The reference clock input to the USB2.0 subsystem is selected by programming the USB0PHYCLKMUX bit in the chip configuration 2 register (CFGCHIP2) of the System Configuration Module. The USB_REFCLKIN source should be selected when it is not possible (such as when specific audio rates are required) to operate the device at one of the allowed input frequencies to the USB2.0 subsystem. The USB2.0 subsystem peripheral bus clock is sourced from SYSCLK2. Table 31-1 determines the source origination as well as the source input frequency to the USB 2.0 PHY. Once the clock source origination (internal/external) and its frequency is determined, the firmware should program the PHY PLL with the correct input frequency via CFGCHIP2.USB0REF_FREQ (see Table 31-2).

NOTE: Prior to accessing any of the device configuration registers, including CFGCHIP2, in order to avoid inadvertent access, two Access Key Registers (KICK0R and KICK1R) should be written with key values. For more information on the device configuration registers, see your device-specific data manual.

Figure 31-2. USB Clocking Diagram

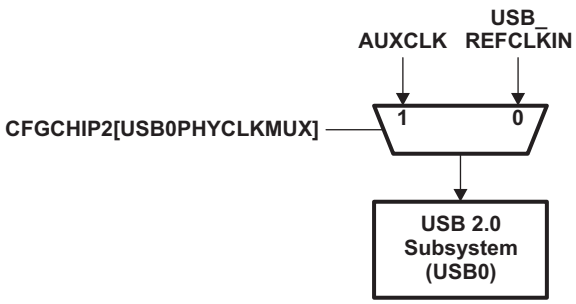


Table 31-1. USB Clock Multiplexing Options

CFGCHIP2. USB0PHYCLKMUX bit	USB2.0 Clock Source	Additional Conditions
0	USB_REFCLKIN	USB_REFCLKIN must be 12, 24, 48, 19.2, 38.4, 13, 26, 20, or 40 MHz. The PLL inside the USB2.0 PHY can be configured to accept any of these input clock frequencies.
1	PLL0_AUXCLK	PLL0_AUXCLK must be 12, 24, 48, 19.2, 38.4, 13, 26, 20, or 40 MHz. The PLL inside the USB2.0 PHY can be configured to accept any of these input clock frequencies.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 31-2. PHY PLL Clock Frequencies Supported

CFGCHIP2.USB0REF_FREQ bit	Frequency
1h	12.0 MHz
2h	24.0 MHz
3h	48.0 MHz
4h	19.2 MHz
5h	38.4 MHz
6h	13.0 MHz
7h	26.0 MHz
8h	20.0 MHz
9h	40.0 MHz

31.2.2 Signal Descriptions

The USB controller provides the I/O signals listed in Table 31-3.

Table 31-3. USB Terminal Functions

Name	I/O[1]	Description
USB0_DP	A I/O/Z	USB0 D+ (differential signal pair)
USB0_DM	A I/O/Z	USB0 D- (differential signal pair)
USB0_ID	A I/O	USB0 operating mode identification pin. For OTG mode or device only mode of operation, do NOT connect the USB0_ID pin, that is, leave the pin floating. For host only mode of operation, connect the USB0_ID pin to ground via a 0 ohm resistor or connect the pin directly to ground.
USB0_VBUS	A I/O/Z	5 volt input that signifies that VBUS is connected. The OTG section of the PHY can also pull-up/pull-down on this signal for HNP and SRP. For device or host only mode of operation, pull-up this pin to 5V. For host mode of operation, also pull-up the USB power signal on the USB connector to 5V. For mixed host/device mode of operation, tie this to the charge pump.
USB0_DRVVBUS	I/O/Z	Digital output to control external 5-V supply
USB0_VDDA33	I/O/Z	USB0 PHY 3.3V supply
USB0_VDDA18	I/O/Z	USB0 PHY 1.8V supply input
USB_REFCLKIN	I/O/Z	External clock input for USB PHY
USB0_VDDA12	I/O/Z	USB PHY 1.2V LD0 output for bypass CAP

(1) A = Analog signal; I = Input; O = Output; Z = High impedance

31.2.3 Indexed and Non-Indexed Registers

The USB controller provides two mechanisms of accessing the endpoint control and status registers:

- Indexed Endpoint Control/Status Registers: These registers are memory-mapped at offset 410h to 41Fh. The endpoint is selected by programming the INDEX register of the controller.
- Non-indexed Endpoint Control/Status Registers: These registers are memory-mapped at offset 500h to 54Fh. Registers at offset 500h to 50Fh map to Endpoint 0; at offset 510h to 51Fh map to Endpoint 1, and so on.

For detailed information about the USB controller registers, see Section 31.4.

Copyright © 2013–2016, Texas Instruments Incorporated

31.2.4 USB PHY Initialization

Two boot configuration registers, pin multiplexing control registers, and chip configuration 2 register (CFGCHIP2) are used to configure the multiplexed pins for USB 2.0 uses. See the *System Configuration (SYSCFG) Module* chapter for more information on the pin multiplexing control registers and CFGCHIP2.

The general procedure for USB PHY initialization starts by releasing the PHY from reset and programming the corresponding bits within the pin multiplexing control registers and CFGCHIP2, for achieving the multiplexed pins for the use of the USB2.0. Next, configuring PHY input clock related information and other PHY general configuration attributes.

When the USB2.0 controller assumes the role of a host, it is tasked to source the required 5V supply via USB0_VBUS (must be at least ≥ 4.75V) pin. The USB2.0 controller makes use of an external charge pump or logic by enabling and disabling the external power logic from the USB2.0 controller core level. It uses the USB0_DRVVBUS for controlling the enable/disable state of the external power logic. In order to achieve this task, the pin multiplexing control registers should be configured accordingly to map the USB0_DRVVBUS pin to be used for USB2.0 purposes. In addition, the source (internal or external) and frequency of the PHY clock should be identified and should be configured by the firmware. This is achieved using CFGCHIP2. Other PHY related fields within CFGCHIP2 should be programmed as: USB0PHYPWDN and USB0OTGPWRDN should be cleared to 0 and USB0DATPOL, USB0SESNDEN, and USB0VBDTCTEN should be set to 1. This will configure the PHY for normal operation as well as also turn on the PHYs VBUS comparator logic. The final task is to turn on the PHY PLL and wait until it locks. You should wait for the PHY clock good status to be set prior to ending the PHY initialization process.

31.2.5 VBUS Voltage Sourcing Control

When the USB controller assumes the role of a host, it is required to supply 5V power to an attached device through its VBUS line. In order to achieve this task, the USB controller requires the use of an external logic (or charge pump) capable of sourcing 5V power. A USB_DRVVBUS is used as a control signal to enable/disable the external logic to either source or disable power on the VBUS line. This control is automatic and is handled by the controller. The USB controller drives the USB_DRVVBUS signal high when it assumes the role of a host while the controller is in Session. When assuming the role of a device, the controller drives the USB_DRVVBUS signal low disabling the external charge pump; hence, no power is driven on the VBUS line.

31.2.6 Dynamic FIFO Sizing

The USB controller supports a total of 4K RAM to dynamically allocate FIFO to all endpoints. The allocation of FIFO space to the different endpoints requires the specification for each Tx and Rx endpoint of:

- The start address of the FIFO within the RAM block
- The maximum size of packet to be supported
- Whether double-buffering is required.

These details are specified through four registers, which are added to the indexed area of the memory map. That is, the registers for the desired endpoint are accessed after programming the INDEX register with the desired endpoint value. Section 31.4.55, Section 31.4.56, Section 31.4.57, and Section 31.4.58 provide details of these registers.

NOTE: The option of dynamically setting FIFO sizes only applies to Endpoints 1-4. The Endpoint 0 FIFO has a fixed size (64 bytes) and a fixed location (start address 0).

It is the responsibility of the firmware to ensure that all the Tx and Rx endpoints that are active in the current USB configuration have a block of RAM assigned exclusively to that endpoint. The RAM must be at least as large as the maximum packet size set for that endpoint.

Copyright © 2013–2016, Texas Instruments Incorporated

31.2.7 USB Controller Host and Peripheral Modes Operation

The USB controller can be used in a range of different environments. It can be used as either a high-speed or a full-speed USB peripheral device attached to a conventional USB host (such as a PC). It can be used as either a host or a peripheral device in a point-to-point type of setup/arrangement or it can be used as a host connecting to a range of peripheral devices in a multi-point setup (that is, using a hub).

The USB2.0 controller role adaptation of a host or peripheral (device) is dependent upon the state of the USB0_ID pin on its mini-AB receptacle. If the USB0_ID pin state is not driven by the connector or is left floating, the USB2.0 controller would assume the role of a peripheral; if the USB0_ID pin state is driven low or is grounded, the USB2.0 controller would assume the role of a host. The state of the USB ID pin is controlled by the type of USB plug attached to the mini-AB connector. A mini/micro-B plug (peripheral) would leave the USB0_ID pin floating and a mini/micro-A plug (host) grounds the USB0_ID pin low. The procedure for the USB2.0 controller determining its operating modes (role of a host or a peripheral) starts when the USB 2.0 controller is in session. The USB 2.0 controller is in session when either it senses a voltage on the USB0_VBUS pin or when the firmware sets the DEVCTL[SESSION] bit.

Usually, the firmware sets the SESSION bit, when it assumes that it will be operating as a host. When the SESSION bit is set, the controller will start sensing the state of the USB0_ID pin. If the USB0_ID pin has been grounded low, then the USB2.0 controller will assume the role of a host; however, if the USB0_ID pin is left floating, then the USB2.0 controller will assume the role of a device. Upon determining its role as a host, it will drive the USB0_DRVVBUS pin high to enable the external power logic so that it start sourcing the required 5V power (must be ≥ 4.75V). The USB2.0 controller will then wait to for the voltage of the USB0_VBUS goes high. If it does not see the power on the USB0_VBUS pin greater than Vbus Valid (4.4V), it will generate an interrupt to the user indicating the existence of a problem. Assuming that the voltage level of the USB0_VBUS is found to be above Vbus Valid, then the USB 2.0 controller will wait for a device to connect, that is, for it to see one of its data lines USB0_DP/DM to be pulled high.

When assuming the role of a peripheral, assuming that the firmware has set the POWER[SOFTCONN] bit and has enabled the data lines and there is an external host sourcing power on the USB0_VBUS line, the USB2.0 controller will transition to session when it sees power (must be greater or equal to 4.01V) on the USB0_VBUS pin. The USB 2.0 controller will then set the SESSION bit upon detecting the power on the USB0_VBUS line. This will force the USB2.0 controller to sense the state of the USB0_ID pin. If the USB0_ID pin has been left floating, it will know that it has to assume the role of a device and will enable its 1.5 kohm pull-up resistor to signify to the attached external host that it is a Full-Speed device. Note that even when operating as a High-Speed; it has to first come up as Full-Speed. The USB2.0 controller will then await for a reset signal from the host.

The USB controller interrupts the CPU on completion of the data transfer on any of the endpoints or on detecting reset, resume, suspend, connect, disconnect, or SOF on the bus.

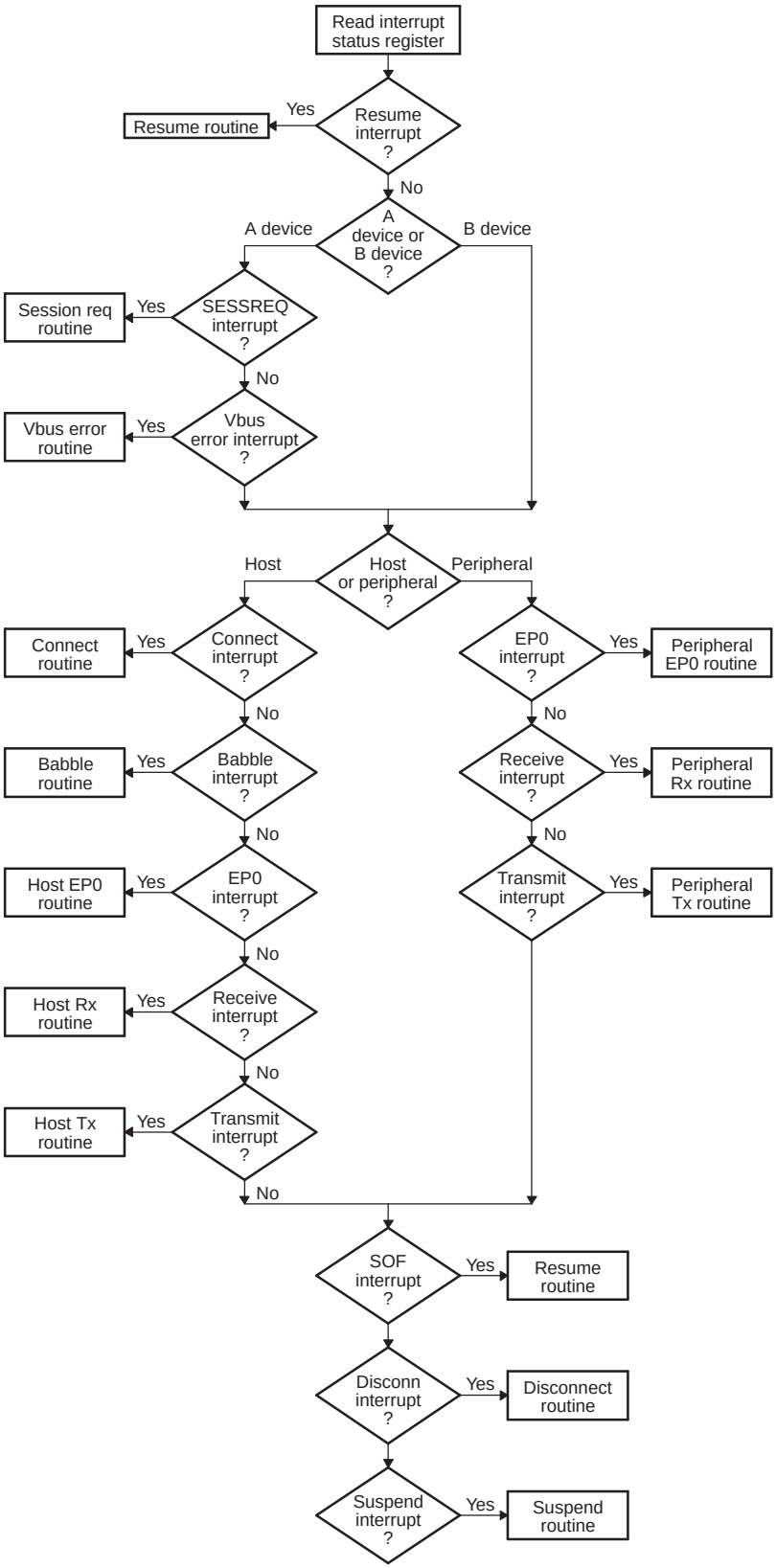
When the CPU is interrupted with a USB interrupt, it needs to read the interrupt status register to determine the endpoints that have caused the interrupt and jump to the appropriate routine. If multiple endpoints have caused the interrupt, endpoint 0 should be serviced first, followed by the other endpoints. The suspend interrupt should be serviced last.

The flowchart in Figure 31-3 describes the interrupt service routine for the USB module.

The following sections describe the programming of USB controller in Peripheral mode and Host mode. DMA operations and interrupt handler mechanisms are common to both peripheral and host mode operations and are discussed after the programming in peripheral and Host mode.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 31-3. Interrupt Service Routine Flow Chart



Copyright © 2013–2016, Texas Instruments Incorporated

31.2.7.1 USB Controller Peripheral Mode Operation

- *Soft connect* - After a reset, the SOFTCONN bit of POWER register (bit 6) is cleared to 0. The controller will therefore appear disconnected until the software has set the SOFTCONN bit to 1. 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.

  Once the SOFTCONN bit has been set, the software can also simulate a disconnect by clearing this bit to 0.
- *Entry into suspend mode* - When operating as a peripheral device, the controller monitors activity on the bus and when no activity has occurred for 3 ms, it goes into Suspend mode. If the Suspend interrupt has been enabled, an interrupt will be generated at this time.

  At this point, the controller can then be left active (and hence able to detect when Resume signaling occurs on the USB), or the application may arrange to disable the controller by stopping its clock. However, the controller will not then be able to detect Resume signaling on the USB. As a result, some external hardware will be needed to detect Resume signaling (by monitoring the DM and DP signals), so that the clock to the controller can be restarted.
- *Resume Signaling* - When resume signaling occurs on the bus, first the clock to the controller must be restarted if necessary. Then the controller will automatically exit Suspend mode. If the Resume interrupt is enabled, an interrupt will be generated.
- *Initiating a remote wakeup* - If the software wants to initiate a remote wakeup while the controller is in Suspend mode, it should write to the Power register to set the RESUME bit to 1. The software should leave then this bit set for approximately 10 ms (minimum of 2 ms, a maximum of 15 ms) before resetting it to 0.

  NOTE: No resume interrupt will be generated when the software initiates a remote wakeup.
- *Reset Signaling* - When reset signaling occurs on the bus, the controller performs the following actions:
  - Clears FADDR register to 0
  - Clears INDEX register to 0
  - Flushes all endpoint FIFOs
  - Clears all control/status registers
  - Generates a reset interrupt.

  If the HSENA bit in the POWER register (bit 5) was set, the controller also tries to negotiate for high-speed operation.

  Whether high-speed operation is selected is indicated by HSMODE bit of POWER register (bit 4).

  When the application software receives a reset interrupt, it should close any open pipes and wait for bus enumeration to begin.

Copyright © 2013–2016, Texas Instruments Incorporated

31.2.7.1.1 Control Transactions

Endpoint 0 is the main control endpoint of the core. The software is required to handle all the standard device requests that may be sent or received via endpoint 0. These are described in Universal Serial Bus Specification, Revision 2.0, Chapter 9. The protocol for these device requests involves different numbers and types of transactions per transfer. To accommodate this, the software needs to take a state machine approach to command decoding and handling.

The Standard Device Requests received by a USB peripheral device can be divided into three categories: Zero Data Requests (in which all the information is included in the command), Write Requests (in which the command will be followed by additional data), and Read Requests (in which the device is required to send data back to the host).

This section looks at the sequence of actions that the software must perform to process these different types of device request.

NOTE: The Setup packet associated with any standard device request should include an 8-byte command. Any setup packet containing a command field of anything other than 8 bytes will be automatically rejected by the controller.

31.2.7.1.1.1 Zero Data Requests

Zero data requests have all their information included in the 8-byte command and require no additional data to be transferred. Examples of Zero Data standard device requests are:

- SET_FEATURE
- CLEAR_FEATURE
- SET_ADDRESS
- SET_CONFIGURATION
- SET_INTERFACE

The sequence of events will begin, as with all requests, when the software receives an endpoint 0 interrupt. The RXPKTRDY bit of PERI_CSR0 (bit 0) will also have been set. The 8-byte command should then be read from the endpoint 0 FIFO, decoded and the appropriate action taken.

For example, if the command is SET_ADDRESS, the 7-bit address value contained in the command should be written to the FADDR register. The PERI_CSR0 register should then be written to set the SERV_RXPKTRDY bit (bit 6) (indicating that the command has been read from the FIFO) and to set the DATAEND bit (bit 3) (indicating that no further data is expected for this request). The interval between setting SERV_RXPKTRDY bit and DATAEND bit should be very small to avoid getting a SetupEnd error condition.

When the host moves to the status stage of the request, a second endpoint 0 interrupt will be generated to indicate that the request has completed. No further action is required from the software. The second interrupt is just a confirmation that the request completed successfully. For SET_ADDRESS command, the address should be set in FADDR register only after the status stage interrupt is received.

If the command is an unrecognized command, or for some other reason cannot be executed, then when it has been decoded, the PERI_CSR0 register should be written to set the SERV_RXPKTRDY bit (bit 6) and to set the SENDSTALL bit (bit 5). When the host moves to the status stage of the request, the controller will send a STALL to tell the host that the request was not executed. A second endpoint 0 interrupt will be generated and the SENTSTALL bit (bit 2 of PERI_CSR0) will be set.

If the host sends more data after the DATAEND bit has been set, then the controller will send a STALL. An endpoint 0 interrupt will be generated and the SENTSTALL bit (bit 2 of PERI_CSR0) will be set.

NOTE: DMA is not supported for endpoint 0, so the command should be read by accessing the endpoint 0 FIFO register.

Copyright © 2013–2016, Texas Instruments Incorporated

31.2.7.1.1.2 Write Requests

Write requests involve an additional packet (or packets) of data being sent from the host after the 8-byte command. An example of a Write standard device request is: SET_DESCRIPTOR.

The sequence of events will begin, as with all requests, when the software receives an endpoint 0 interrupt. The RXPKTRDY bit of PERI_CSR0 will also have been set. The 8-byte command should then be read from the Endpoint 0 FIFO and decoded.

As with a zero data request, the PERI_CSR0 register should then be written to set the SERV_RXPKTRDY bit (bit 6) (indicating that the command has been read from the FIFO) but in this case the DATAEND bit (bit 3) should not be set (indicating that more data is expected).

When a second endpoint 0 interrupt is received, the PERI_CSR0 register should be read to check the endpoint status. The RXPKTRDY bit of PERI_CSR0 should be set to indicate that a data packet has been received. The COUNT0 register should then be read to determine the size of this data packet. The data packet can then be read from the endpoint 0 FIFO.

If the length of the data associated with the request (indicated by the wLength field in the command) is greater than the maximum packet size for endpoint 0, further data packets will be sent. In this case, PERI_CSR0 should be written to set the SERV_RXPKTRDY bit, but the DATAEND bit should not be set.

When all the expected data packets have been received, the PERI_CSR0 register should be written to set the SERV_RXPKTRDY bit and to set the DATAEND bit (indicating that no more data is expected).

When the host moves to the status stage of the request, another endpoint 0 interrupt will be generated to indicate that the request has completed. No further action is required from the software, the interrupt is just a confirmation that the request completed successfully.

If the command is an unrecognized command, or for some other reason cannot be executed, then when it has been decoded, the PERI_CSR0 register should be written to set the SERV_RXPKTRDY bit (bit 6) and to set the SENDSTALL bit (bit 5). When the host sends more data, the controller will send a STALL to tell the host that the request was not executed. An endpoint 0 interrupt will be generated and the SENTSTALL bit of PERI_CSR0 (bit 2) will be set.

If the host sends more data after the DATAEND has been set, then the controller will send a STALL. An endpoint 0 interrupt will be generated and the SENTSTALL bit of PERI_CSR0 (bit 2) will be set.

Copyright © 2013–2016, Texas Instruments Incorporated

31.2.7.1.1.3 Read Requests

Read requests have a packet (or packets) of data sent from the function to the host after the 8-byte command. Examples of Read Standard Device Requests are:

- GET_CONFIGURATION
- GET_INTERFACE
- GET_DESCRIPTOR
- GET_STATUS
- SYNCH_FRAME

The sequence of events will begin, as with all requests, when the software receives an endpoint 0 interrupt. The RXPKTRDY bit of PERI_CSR0 (bit 0) will also have been set. The 8-byte command should then be read from the endpoint 0 FIFO and decoded. The PERI_CSR0 register should then be written to set the SERV_RXPKTRDY bit (bit 6) (indicating that the command has read from the FIFO).

The data to be sent to the host should then be written to the endpoint 0 FIFO. If the data to be sent is greater than the maximum packet size for endpoint 0, only the maximum packet size should be written to the FIFO. The PERI_CSR0 register should then be written to set the TXPKTRDY bit (bit 1) (indicating that there is a packet in the FIFO to be sent). When the packet has been sent to the host, another endpoint 0 interrupt will be generated and the next data packet can be written to the FIFO.

When the last data packet has been written to the FIFO, the PERI_CSR0 register should be written to set the TXPKTRDY bit and to set the DATAEND bit (bit 3) (indicating that there is no more data after this packet).

When the host moves to the status stage of the request, another endpoint 0 interrupt will be generated to indicate that the request has completed. No further action is required from the software: the interrupt is just a confirmation that the request completed successfully.

If the command is an unrecognized command, or for some other reason cannot be executed, then when it has been decoded, the PERI_CSR0 register should be written to set the SERV_RXPKTRDY bit (bit 6) and to set the SENDSTALL bit (bit 5). When the host requests data, the controller will send a STALL to tell the host that the request was not executed. An endpoint 0 interrupt will be generated and the SENTSTALL bit of PERI_CSR0 (bit 2) will be set.

If the host requests more data after DATAEND (bit 3) has been set, then the controller will send a STALL. An endpoint 0 interrupt will be generated and the SENTSTALL bit of PERI_CSR0 (bit 2) will be set.

31.2.7.1.1.4 Endpoint 0 States

When the USB controller is operating as a peripheral device, the endpoint 0 control needs three modes – IDLE, TX and RX – corresponding to the different phases of the control transfer and the states endpoint 0 enters for the different phases of the transfer (described in later sections).

The default mode on power-up or reset should be IDLE. RXPKTRDY bit of PERI_CSR0 (bit 0) becoming set when endpoint 0 is in IDLE state indicates a new device request. Once the device request is unloaded from the FIFO, the controller decodes the descriptor to find whether there is a data phase and, if so, the direction of the data phase of the control transfer (in order to set the FIFO direction). See Figure 31-4.

Depending on the direction of the data phase, endpoint 0 goes into either TX state or RX state. If there is no Data phase, endpoint 0 remains in IDLE state to accept the next device request.

The actions that the CPU needs to take at the different phases of the possible transfers (for example, loading the FIFO, setting TXPKTRDY) are indicated in Figure 31-5 .

NOTE: The controller changes the FIFO direction, depending on the direction of the data phase independently of the CPU.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 31-4. CPU Actions at Transfer Phases

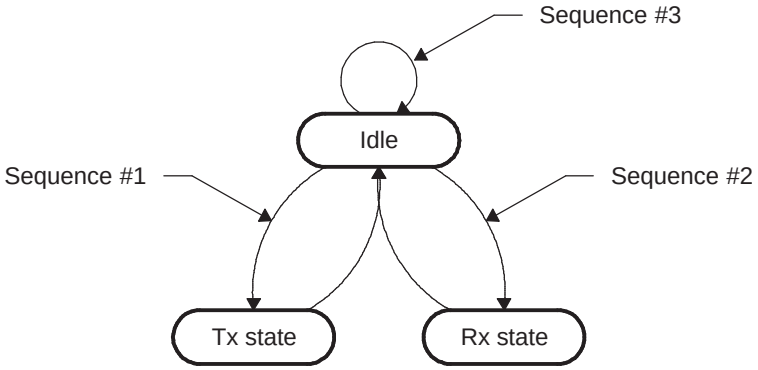
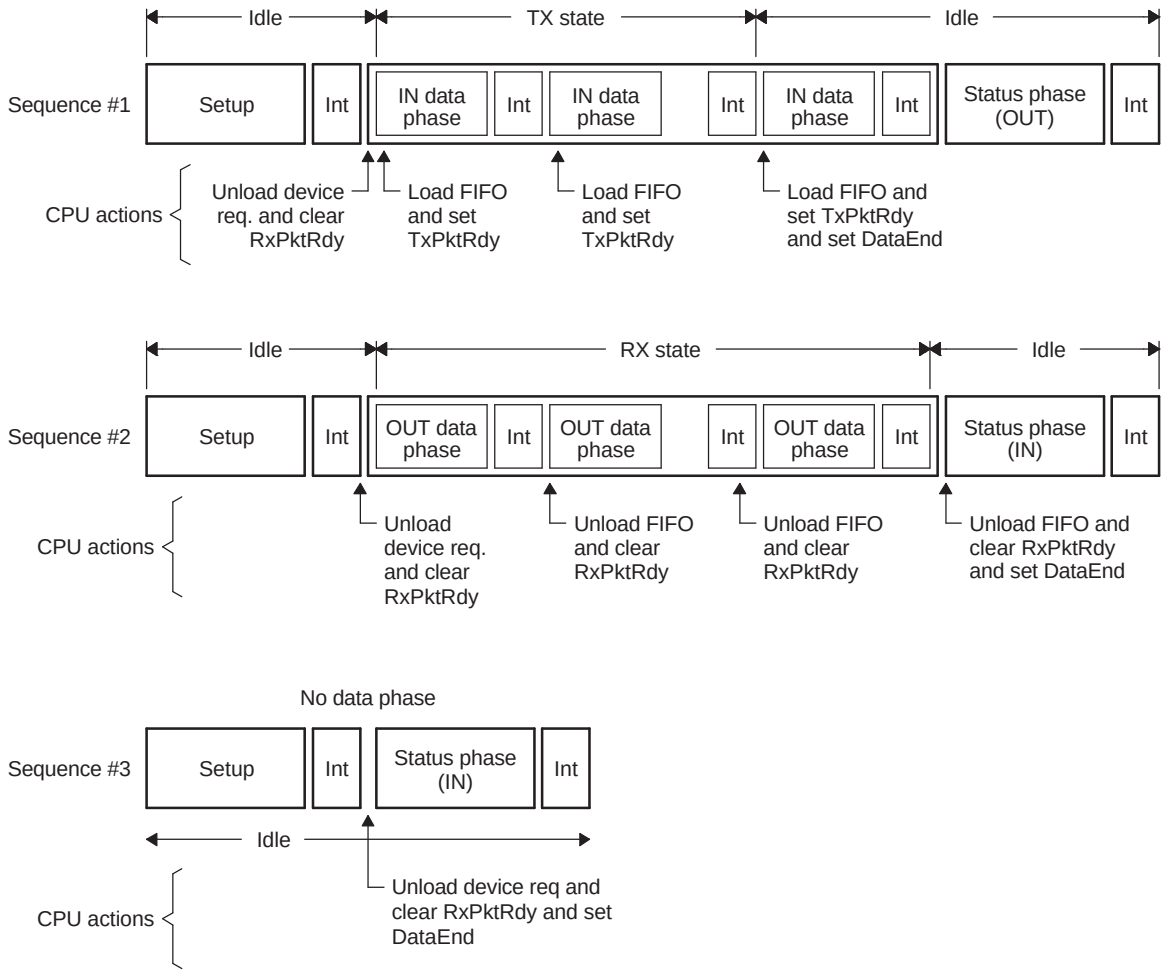


Figure 31-5. Sequence of Transfer



Copyright © 2013–2016, Texas Instruments Incorporated

31.2.7.1.1.5 Endpoint 0 Service Routine

An Endpoint 0 interrupt is generated when:

- The controller sets the RXPKTRDY bit of PERI_CSR0 (bit 0) after a valid token has been received and data has been written to the FIFO.
- The controller clears the TXPKTRDY bit of PERI_CSR0 (bit 1) after the packet of data in the FIFO has been successfully transmitted to the host.
- The controller sets the SENTSTALL bit of PERI_CSR0 (bit 2) after a control transaction is ended due to a protocol violation.
- The controller sets the SETUPEND bit of PERI_CSR0 (bit 4) because a control transfer has ended before DATAEND (bit 3 of PERI_CSR0) is set.

Whenever the endpoint 0 service routine is entered, the software must first check to see if the current control transfer has been ended due to either a STALL condition or a premature end of control transfer. If the control transfer ends due to a STALL condition, the SENTSTALL bit would be set. If the control transfer ends due to a premature end of control transfer, the SETUPEND bit would be set. In either case, the software should abort processing the current control transfer and set the state to IDLE.

Once the software has determined that the interrupt was not generated by an illegal bus state, the next action taken depends on the endpoint state. Figure 31-6 shows the flow of this process.

If endpoint 0 is in IDLE state, the only valid reason an interrupt can be generated is as a result of the controller receiving data from the bus. The service routine must check for this by testing the RXPKTRDY bit of PERI_CSR0 (bit 0). If this bit is set, then the controller has received a SETUP packet. This must be unloaded from the FIFO and decoded to determine the action the controller must take. Depending on the command contained within the SETUP packet, endpoint 0 will enter one of three states:

- If the command is a single packet transaction (SET_ADDRESS, SET_INTERFACE etc.) without any data phase, the endpoint will remain in IDLE state.
- If the command has an OUT data phase (SET_DESCRIPTOR etc.), the endpoint will enter RX state.
- If the command has an IN data phase (GET_DESCRIPTOR etc.), the endpoint will enter TX state.

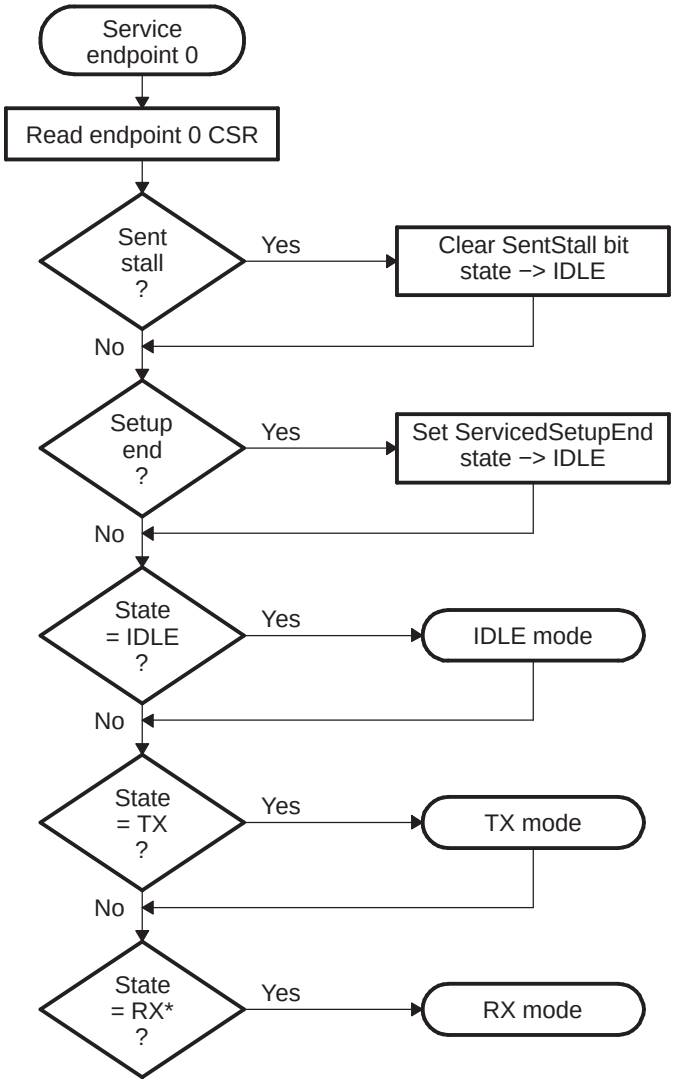
If the endpoint 0 is in TX state, the interrupt indicates that the core has received an IN token and data from the FIFO has been sent. The software must respond to this either by placing more data in the FIFO if the host is still expecting more data or by setting the DATAEND bit to indicate that the data phase is complete. Once the data phase of the transaction has been completed, endpoint 0 should be returned to IDLE state to await the next control transaction.

NOTE: All command transactions include a field that indicates the amount of data the host expects to receive or is going to send.

If the endpoint is in RX state, the interrupt indicates that a data packet has been received. The software must respond by unloading the received data from the FIFO. The software must then determine whether it has received all of the expected data. If it has, the software should set the DATAEND bit and return endpoint 0 to IDLE state. If more data is expected, the firmware should set the SERV_RXPKTRDY bit of PERI_CSR0 (bit 6) to indicate that it has read the data in the FIFO and leave the endpoint in RX state.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 31-6. Service Endpoint 0 Flow Chart



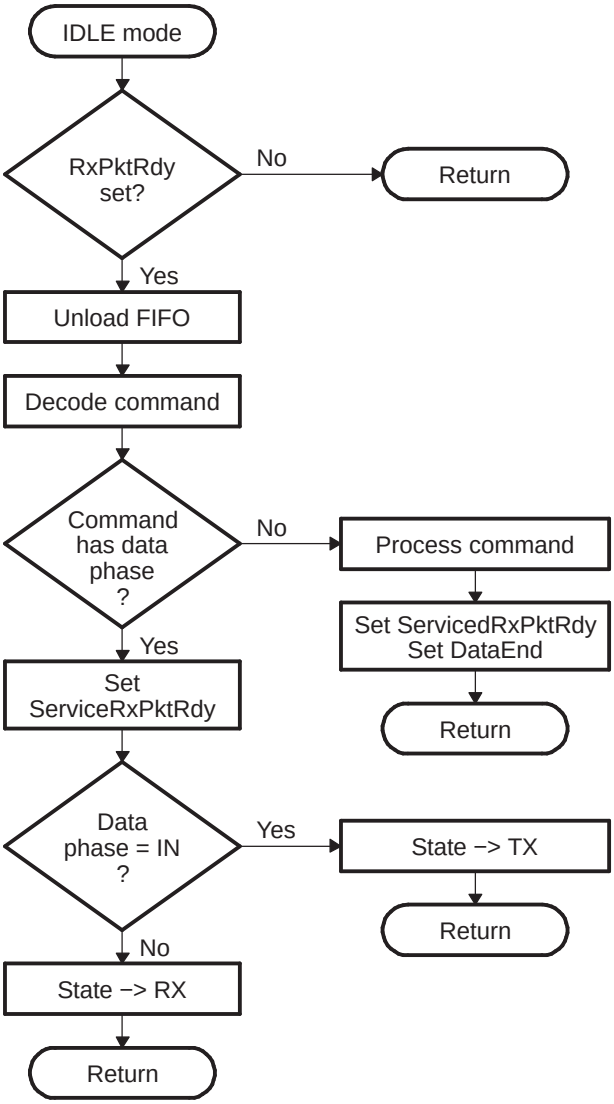
* By default

Copyright © 2013–2016, Texas Instruments Incorporated

31.2.7.1.1.5.1 IDLE Mode

IDLE mode is the mode the endpoint 0 control must select at power-on or reset and is the mode to which the endpoint 0 control should return when the RX and TX modes are terminated. It is also the mode in which the SETUP phase of control transfer is handled (as outlined in Figure 31-7).

Figure 31-7. IDLE Mode Flow Chart



Copyright © 2013–2016, Texas Instruments Incorporated

31.2.7.1.1.5.2 TX Mode

When the endpoint is in TX state all arriving IN tokens need to be treated as part of a data phase until the required amount of data has been sent to the host. If either a SETUP or an OUT token is received while the endpoint is in the TX state, this will cause a SetupEnd condition to occur as the core expects only IN tokens. See Figure 31-8.

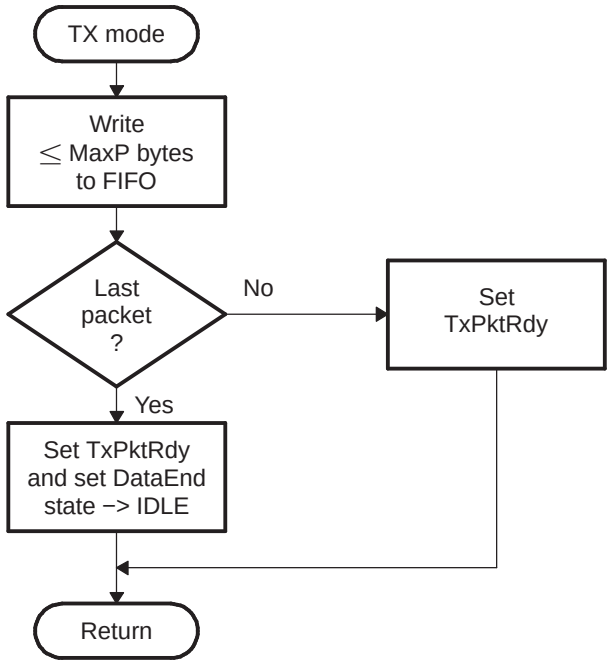
Three events can cause TX mode to be terminated before the expected amount of data has been sent:

1. The host sends an invalid token causing a SETUPEND condition (bit 4 of PERI_CSR0 set).
2. The software sends a packet containing less than the maximum packet size for endpoint 0.
3. The software sends an empty data packet.

Until the transaction is terminated, the software simply needs to load the FIFO when it receives an interrupt that indicates a packet has been sent from the FIFO. (An interrupt is generated when TXPKTRDY is cleared.)

When the software forces the termination of a transfer (by sending a short or empty data packet), it should set the DATAEND bit of PERI_CSR0 (bit 3) to indicate to the core that the data phase is complete and that the core should next receive an acknowledge packet.

Figure 31-8. TX Mode Flow Chart



Copyright © 2013–2016, Texas Instruments Incorporated

31.2.7.1.1.5.3 RX Mode

In RX mode, all arriving data should be treated as part of a data phase until the expected amount of data has been received. If either a SETUP or an IN token is received while the endpoint is in RX state, a SetupEnd condition will occur as the controller expects only OUT tokens.

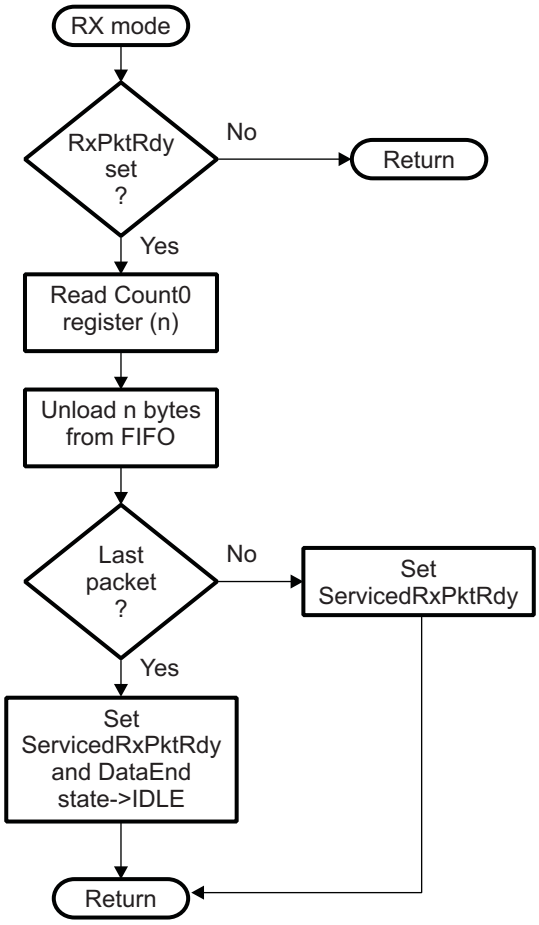
Three events can cause RX mode to be terminated before the expected amount of data has been received as shown in Figure 31-9:

1. The host sends an invalid token causing a SETUPEND condition (setting bit 4 of PERI_CSR0).
2. The host sends a packet which contains less than the maximum packet size for endpoint 0.
3. The host sends an empty data packet.

Until the transaction is terminated, the software unloads the FIFO when it receives an interrupt that indicates new data has arrived (setting RXPKTRDY bit of PERI_CSR0) and to clear RXPKTRDY by setting the SERV_RXPKTRDY bit of PERI_CSR0 (bit 6).

When the software detects the termination of a transfer (by receiving either the expected amount of data or an empty data packet), it should set the DATAEND bit (bit 3 of PERI_CSR0) to indicate to the controller that the data phase is complete and that the core should receive an acknowledge packet next.

Figure 31-9. RX Mode Flow Chart



Copyright © 2013–2016, Texas Instruments Incorporated

31.2.7.1.1.5.4 Error Handling

A control transfer may be aborted due to a protocol error on the USB, the host prematurely ending the transfer, or if the software wishes to abort the transfer (for example, because it cannot process the command).

The controller automatically detects protocol errors and sends a STALL packet to the host under the following conditions:

- The host sends more data during the OUT Data phase of a write request than was specified in the command. This condition is detected when the host sends an OUT token after the DATAEND bit (bit 3 of PERI_CSR0) has been set.
- The host requests more data during the IN Data phase of a read request than was specified in the command. This condition is detected when the host sends an IN token after the DATAEND bit in the PERI_CSR0 register has been set.
- The host sends more than Max Packet Size data bytes in an OUT data packet.
- The host sends a non-zero length DATA1 packet during the STATUS phase of a read request.

When the controller has sent the STALL packet, it sets the SENTSTALL bit (bit 2 of PERI_CSR0) and generates an interrupt. When the software receives an endpoint 0 interrupt with the SENTSTALL bit set, it should abort the current transfer, clear the SENTSTALL bit, and return to the IDLE state.

If the host prematurely ends a transfer by entering the STATUS phase before all the data for the request has been transferred, or by sending a new SETUP packet before completing the current transfer, then the SETUPEND bit (bit 4 of PERI_CSR0) will be set and an endpoint 0 interrupt generated. When the software receives an endpoint 0 interrupt with the SETUPEND bit set, it should abort the current transfer, set the SERV_SETUPEND bit (bit 7 of PERI_CSR0), and return to the IDLE state. If the RXPKTRDY bit (bit 0 of PERI_CSR0) is set this indicates that the host has sent another SETUP packet and the software should then process this command.

If the software wants to abort the current transfer, because it cannot process the command or has some other internal error, then it should set the SENDSTALL bit (bit 5 of PERI_CSR0). The controller will then send a STALL packet to the host, set the SENTSTALL bit (bit 2 of PERI_CSR0) and generate an endpoint 0 interrupt.

31.2.7.1.1.5.5 Additional Conditions

When working as a peripheral device, the controller automatically responds to certain conditions on the USB bus or actions by the host. The details are:

- Stall Issued to Control Transfers
  - 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 controller when the host sends an OUT token (instead of an IN token) after the software has unloaded the last OUT packet and set DataEnd.
  - 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 controller when the host sends an IN token (instead of an OUT token) after the software has cleared TXPKTRDY and set DataEnd in response to the ACK issued by the host to what should have been the last packet.
  - The host sends more than MaxPktSize data with an OUT data token.
  - The host sends the wrong PID for the OUT Status phase of a Control transfer.
  - The host sends more than a zero length data packet for the OUT Status phase.
- Zero Length Out Data Packets In Control Transfer
  - 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 (i.e., after the software has set DataEnd). If, however, the host sends a zero length OUT data packet before the entire length of device request has been transferred, this signals the premature end of the transfer. In this case, the controller will automatically flush any IN token loaded by software ready for the Data phase from the FIFO and set SETUPEND bit (bit 4 of PERI_CSR0).

Copyright © 2013–2016, Texas Instruments Incorporated

31.2.7.1.2 Bulk Transactions

31.2.7.1.2.1 Bulk In Transactions

A Bulk IN transaction is used to transfer non-periodic data from the USB peripheral device to the host.

The following optional features are available for use with a Tx endpoint used in peripheral mode for Bulk IN transactions:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO awaiting transmission to the host. Double packet buffering is enabled by setting the DPB bit of TXFIFOSZ register (bit 4).
- DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint is able to accept another packet in its FIFO. This feature allows the DMA controller to load packets into the FIFO without processor intervention.

31.2.7.1.2.1.1 Setup

In configuring a TX endpoint for bulk transactions, the TXMAXP register must be written with the maximum packet size (in bytes) for the endpoint. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the endpoint and the PERI_TXCSR register should be set as shown in Table 31-4 when using DMA:

Table 31-4. PERI_TXCSR Register Bit Configuration for Bulk IN Transactions

Bit Position	Bit Field Name	Configuration
Bit 14	ISO	Cleared to 0 for bulk mode operation.
Bit 13	MODE	Set to 1 to make sure the FIFO is enabled (only necessary if the FIFO is shared with an RX endpoint).
Bit 12	DMAEN	Set to 1 if DMA requests must be enabled.
Bit 11	FRCDATATOG	Cleared to 0 to allow normal data toggle operations.
Bit 10	DMAMODE	Set to 1 when DMA is enabled.

When the endpoint is first configured (following a SET_CONFIGURATION or SET_INTERFACE command on Endpoint 0), the lower byte of PERI_TXCSR should be written to set the CLRDATATOG bit (bit 6). This will ensure that the data toggle (which is handled automatically by the controller) starts in the correct state.

Also if there are any data packets in the FIFO, indicated by the FIFONOTEMPTY bit (bit 1 of PERI_TXCSR) being set, they should be flushed by setting the FLUSHFIFO bit (bit 3 of PERI_TXCSR).

NOTE: It may be necessary to set this bit twice in succession if double buffering is enabled.

31.2.7.1.2.1.2 Operation

When data is to be transferred over a Bulk IN pipe, a data packet needs to be loaded into the FIFO and the PERI_TXCSR register written to set the TXPKTRDY bit (bit 0). When the packet has been sent, the TXPKTRDY bit will be cleared by the USB controller and an interrupt generated so that the next packet can be loaded into the FIFO. If double packet buffering is enabled, then after the first packet has been loaded and the TXPKTRDY bit set, the TXPKTRDY bit will immediately be cleared by the USB controller and an interrupt generated so that a second packet can be loaded into the FIFO. The software should operate in the same way, loading a packet when it receives an interrupt, regardless of whether double packet buffering is enabled or not.

In the general case, the packet size must not exceed the size specified by the lower 11 bits of the TXMAXP register. This part of the register defines the payload (packet size) for transfers over the USB and is required by the USB Specification to be either 8, 16, 32, 64 (Full-Speed or High-Speed) or 512 bytes (High-Speed only).

Copyright © 2013–2016, Texas Instruments Incorporated

The host may determine that all the data for a transfer has been sent by knowing the total amount of data that is expected. Alternatively it may infer that all the data has been sent when it receives a packet which is smaller than the stated payload (TXMAXP[10-0]). In the latter case, if the total size of the data block is a multiple of this payload, it will be necessary for the function to send a null packet after all the data has been sent. This is done by setting TXPKTRDY when the next interrupt is received, without loading any data into the FIFO.

If large blocks of data are being transferred, then the overhead of calling an interrupt service routine to load each packet can be avoided by using DMA.

31.2.7.1.2.1.3 Error Handling

If the software wants to shut down the Bulk IN pipe, it should set the SENDSTALL bit (bit 4 of PERI_TXCSR). When the controller receives the next IN token, it will send a STALL to the host, set the SENTSTALL bit (bit 5 of PERI_TXCSR) and generate an interrupt.

When the software receives an interrupt with the SENTSTALL bit (bit 5 of PERI_TXCSR) set, it should clear the SENTSTALL bit. It should however leave the SENDSTALL bit set until it is ready to re-enable the Bulk IN pipe.

NOTE: If the host failed to receive the STALL packet for some reason, it will send another IN token, so it is advisable to leave the SENDSTALL bit set until the software is ready to re-enable the Bulk IN pipe. When a pipe is re-enabled, the data toggle sequence should be restarted by setting the CLRDATATOG bit in the PERI_TXCSR register (bit 6).

31.2.7.1.2.2 Bulk OUT Transactions

A Bulk OUT transaction is used to transfer non-periodic data from the host to the function controller.

The following optional features are available for use with an Rx endpoint used in peripheral mode for Bulk OUT transactions:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO on reception from the host. Double packet buffering is enabled by setting the DPB bit of the RXFIFOSZ register (bit 4).
- DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint has a packet in its FIFO. This feature can be used to allow the DMA controller to unload packets from the FIFO without processor intervention.

  When DMA is enabled, endpoint interrupt will not be generated for completion of packet reception. Endpoint interrupt will be generated only in the error conditions.

31.2.7.1.2.2.1 Setup

In configuring an Rx endpoint for Bulk OUT transactions, the RXMAXP register must be written with the maximum packet size (in bytes) for the endpoint. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the endpoint. In addition, the relevant interrupt enable bit in the INTRRXE register should be set (if an interrupt is required for this endpoint) and the PERI_RXCSR register should be set as shown in Table 31-5.

Table 31-5. PERI_RXCSR Register Bit Configuration for Bulk OUT Transactions

Bit Position	Bit Field Name	Configuration
Bit 14	ISO	Cleared to 0 to enable Bulk protocol.
Bit 13	DMAEN	Set to 1 if a DMA request is required for this endpoint.
Bit 12	DISNYET	Cleared to 0 to allow normal PING flow control. This will affect only high speed transactions.
Bit 11	DMAMODE	Always clear this bit to 0.

Copyright © 2013–2016, Texas Instruments Incorporated

When the endpoint is first configured (following a SET_CONFIGURATION or SET_INTERFACE command on Endpoint 0), the lower byte of PERI_RXCSR should be written to set the CLRDATATOG bit (bit 7). This will ensure that the data toggle (which is handled automatically by the USB controller) starts in the correct state.

Also if there are any data packets in the FIFO (indicated by the RXPKTRDY bit (bit 0 of PERI_RXCSR) being set), they should be flushed by setting the FLUSHFIFO bit (bit 4 of PERI_RXCSR).

NOTE: It may be necessary to set this bit twice in succession if double buffering is enabled.

31.2.7.1.2.2.2 Operation

When a data packet is received by a Bulk Rx endpoint, the RXPKTRDY bit (bit 0 of PERI_RXCSR) is set and an interrupt is generated. The software should read the RXCOUNT register for the endpoint to determine the size of the data packet. The data packet should be read from the FIFO, then the RXPKTRDY bit should be cleared.

The packets received should not exceed the size specified in the RXMAXP register (as this should be the value set in the wMaxPacketSize field of the endpoint descriptor sent to the host). When a block of data larger than wMaxPacketSize needs to be sent to the function, it will be sent as multiple packets. All the packets will be wMaxPacketSize in size, except the last packet which will contain the residue. The software may use an application specific method of determining the total size of the block and hence when the last packet has been received. Alternatively it may infer that the entire block has been received when it receives a packet which is less than wMaxPacketSize in size. (If the total size of the data block is a multiple of wMaxPacketSize, a null data packet will be sent after the data to signify that the transfer is complete.)

In the general case, the application software will need to read each packet from the FIFO individually. If large blocks of data are being transferred, the overhead of calling an interrupt service routine to unload each packet can be avoided by using DMA.

31.2.7.1.2.2.3 Error Handling

If the software wants to shut down the Bulk OUT pipe, it should set the SENDSTALL bit (bit 5 of PERI_RXCSR). When the controller receives the next packet it will send a STALL to the host, set the SENTSTALL bit (bit 6 of PERI_RXCSR) and generate an interrupt.

When the software receives an interrupt with the SENTSTALL bit (bit 6 of PERI_RXCSR) set, it should clear this bit. It should however leave the SENDSTALL bit set until it is ready to re-enable the Bulk OUT pipe.

NOTE: If the host failed to receive the STALL packet for some reason, it will send another packet, so it is advisable to leave the SENDSTALL bit set until the software is ready to re-enable the Bulk OUT pipe. When a Bulk OUT pipe is re-enabled, the data toggle sequence should be restarted by setting the CLRDATATOG bit (bit 7) in the PERI_RXCSR register.

31.2.7.1.3 Peripheral Mode: Interrupt Transactions

An Interrupt IN transaction uses the same protocol as a Bulk IN transaction and can be used the same way. Similarly, an Interrupt OUT transaction uses almost the same protocol as a Bulk OUT transaction and can be used the same way.

Tx endpoints in the USB controller have one feature for Interrupt IN transactions that they do not support in Bulk IN transactions. In Interrupt IN transactions, the endpoints support continuous toggle of the data toggle bit.

This feature is enabled by setting the FRCDATATOG bit in the PERI_TXCSR register (bit 11). When this bit is set, the controller will consider the packet as having been successfully sent and toggle the data bit for the endpoint, regardless of whether an ACK was received from the host.

Copyright © 2013–2016, Texas Instruments Incorporated

Another difference is that interrupt endpoints do not support PING flow control. This means that the controller should never respond with a NYET handshake, only ACK/NAK/STALL. To ensure this, the DISNYET bit in the PERI_RXCSR register (bit 12) should be set to disable the transmission of NYET handshakes in high-speed mode.

Though DMA can be used with an interrupt OUT endpoint, it generally offers little benefit as interrupt endpoints are usually expected to transfer all their data in a single packet.

31.2.7.1.4 Isochronous Transactions

31.2.7.1.4.1 Peripheral Mode: Isochronous IN Transactions

An Isochronous IN transaction is used to transfer periodic data from the function controller to the host.

The following optional features are available for use with a Tx endpoint used in Peripheral mode for Isochronous IN transactions:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO awaiting transmission to the host. Double packet buffering is enabled by setting the DPB bit of TXFIFOSZ register (bit 4).

  NOTE: Double packet buffering is generally advisable for Isochronous transactions in order to avoid Underrun errors as described in later section.

- DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint is able to accept another packet in its FIFO. This feature allows the DMA controller to load packets into the FIFO without processor intervention.

  However, this feature is not particularly useful with Isochronous endpoints because the packets transferred are often not maximum packet size and the PERI_TXCSR register needs to be accessed following every packet to check for Underrun errors.

  When DMA is enabled and DMAMODE bit of PERI_TXCSR is set, endpoint interrupt will not be generated for completion of packet transfer. Endpoint interrupt will be generated only in the error conditions.

31.2.7.1.4.1.1 Setup

In configuring a Tx endpoint for Isochronous IN transactions, the TXMAXP register must be written with the maximum packet size (in bytes) for the endpoint. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the endpoint. In addition, the relevant interrupt enable bit in the INTRTXE register should be set (if an interrupt is required for this endpoint) and the PERI_TXCSR register should be set as shown in Table 31-6.

Table 31-6. PERI_TXCSR Register Bit Configuration for Isochronous IN Transactions

Bit Position	Bit Field Name	Configuration
Bit 14	ISO	Set to 1 to enable Isochronous transfer protocol.
Bit 13	MODE	Set to 1 to ensure the FIFO is enabled (only necessary if the FIFO is shared with an Rx endpoint).
Bit 12	DMAEN	Set to 1 if DMA Requests have to be enabled.
Bit 11	FRCDATATOG	Ignored in Isochronous mode.
Bit 10	DMAMODE	Set to 1 when DMA is enabled.

Copyright © 2013–2016, Texas Instruments Incorporated

31.2.7.1.4.1.2 Operation

An Isochronous endpoint does not support data retries, so if data underrun is to be avoided, the data to be sent to the host must be loaded into the FIFO before the IN token is received. The host will send one IN token per frame (or microframe in High-speed mode), however the timing within the frame (or microframe) can vary. If an IN token is received near the end of one frame and then at the start of the next frame, there will be little time to reload the FIFO. For this reason, double buffering of the endpoint is usually necessary.

An interrupt is generated whenever a packet is sent to the host and the software may use this interrupt to load the next packet into the FIFO and set the TXPKTRDY bit in the PERI_TXCSR register (bit 0) in the same way as for a Bulk Tx endpoint. As the interrupt could occur almost any time within a frame(/microframe), depending on when the host has scheduled the transaction, this may result in irregular timing of FIFO load requests. If the data source for the endpoint is coming from some external hardware, it may be more convenient to wait until the end of each frame(/microframe) before loading the FIFO as this will minimize the requirement for additional buffering. This can be done by using either the SOF interrupt or the external SOF_PULSE signal from the controller to trigger the loading of the next data packet. The SOF_PULSE is generated once per frame(/microframe) when a SOF packet is received. (The controller also maintains an external frame(/microframe) counter so it can still generate a SOF_PULSE when the SOF packet has been lost.) The interrupts may still be used to set the TXPKTRDY bit in PERI_TXCSR (bit 0) and to check for data overruns/underruns.

Starting up a double-buffered Isochronous IN pipe can be a source of problems. Double buffering requires that a data packet is not transmitted until the frame(/microframe) after it is loaded. There is no problem if the function loads the first data packet at least a frame(/microframe) before the host sets up the pipe (and therefore starts sending IN tokens). But if the host has already started sending IN tokens by the time the first packet is loaded, the packet may be transmitted in the same frame(/microframe) as it is loaded, depending on whether it is loaded before, or after, the IN token is received. This potential problem can be avoided by setting the ISOUPDATE bit in the POWER register (bit 7). When this bit is set, any data packet loaded into an Isochronous Tx endpoint FIFO will not be transmitted until after the next SOF packet has been received, thereby ensuring that the data packet is not sent too early.

31.2.7.1.4.1.3 Error Handling

If the endpoint has no data in its FIFO when an IN token is received, it will send a null data packet to the host and set the UNDERRUN bit in the PERI_TXCSR register (bit 2). This is an indication that the software is not supplying data fast enough for the host. It is up to the application to determine how this error condition is handled.

If the software is loading one packet per frame(/microframe) and it finds that the TXPKTRDY bit in the PERI_TXCSR register (bit 0) is set when it wants to load the next packet, this indicates that a data packet has not been sent (perhaps because an IN token from the host was corrupted). It is up to the application how it handles this condition: it may choose to flush the unsent packet by setting the FLUSHFIFO bit in the PERI_TXCSR register (bit 3), or it may choose to skip the current packet.

Copyright © 2013–2016, Texas Instruments Incorporated

31.2.7.1.4.2 Peripheral Mode: Isochronous OUT Transactions

An Isochronous OUT transaction is used to transfer periodic data from the host to the function controller.

Following optional features are available for use with an Rx endpoint used in Peripheral mode for Isochronous OUT transactions:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO on reception from the host. Double packet buffering is enabled by setting the DPB bit of RXFIFOSZ register (bit 4).

  NOTE: Double packet buffering is generally advisable for Isochronous transactions in order to avoid Overrun errors.

- DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint has a packet in its FIFO. This feature can be used to allow the DMA controller to unload packets from the FIFO without processor intervention.

  However, this feature is not particularly useful with Isochronous endpoints because the packets transferred are often not maximum packet size and the PERI_RXCSR register needs to be accessed following every packet to check for Overrun or CRC errors.

  When DMA is enabled, endpoint interrupt will not be generated for completion of packet reception. Endpoint interrupt will be generated only in the error conditions.

31.2.7.1.4.2.1 Setup

In configuring an Rx endpoint for Isochronous OUT transactions, the RXMAXP register must be written with the maximum packet size (in bytes) for the endpoint. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the endpoint. In addition, the relevant interrupt enable bit in the INTRRXE register should be set (if an interrupt is required for this endpoint) and the PERI_RXCSR register should be set as shown in Table 31-7.

Table 31-7. PERI_RXCSR Register Bit Configuration for Isochronous OUT Transactions

Bit Position	Bit Field Name	Configuration
Bit 14	ISO	Set to 1 to enable isochronous protocol.
Bit 13	DMAEN	Set to 1 if a DMA request is required for this endpoint.
Bit 12	DISNYET	Ignored in isochronous transfers.
Bit 11	DMAMODE	Always clear this bit to 0.

31.2.7.1.4.2.2 Operation

An Isochronous endpoint does not support data retries so, if a data overrun is to be avoided, there must be space in the FIFO to accept a packet when it is received. The host will send one packet per frame (or microframe in High-speed mode); however, the time within the frame can vary. If a packet is received near the end of one frame(/microframe) and another arrives at the start of the next frame, there will be little time to unload the FIFO. For this reason, double buffering of the endpoint is usually necessary.

An interrupt is generated whenever a packet is received from the host and the software may use this interrupt to unload the packet from the FIFO and clear the RXPKTRDY bit in the PERI_RXCSR register (bit 0) in the same way as for a Bulk Rx endpoint. As the interrupt could occur almost any time within a frame(/microframe), depending on when the host has scheduled the transaction, the timing of FIFO unload requests will probably be irregular. If the data sink for the endpoint is going to some external hardware, it may be better to minimize the requirement for additional buffering by waiting until the end of each frame(/microframe) before unloading the FIFO. This can be done by using either the SOF interrupt or the external SOF_PULSE signal from the controller to trigger the unloading of the data packet. The SOF_PULSE is generated once per frame(/microframe) when a SOF packet is received. (The controller also maintains an external frame(/microframe) counter so it can still generate a SOF_PULSE when the SOF packet has been lost.) The interrupts may still be used to clear the RXPKTRDY bit in PERI_RXCSR and to check for data overruns/underruns.

Copyright © 2013–2016, Texas Instruments Incorporated

31.2.7.1.4.2.3 Error Handling

If there is no space in the FIFO to store a packet when it is received from the host, the OVERRUN bit in the PERI_RXCSR register (bit 2) will be set. This is an indication that the software is not unloading data fast enough for the host. It is up to the application to determine how this error condition is handled.

If the controller finds that a received packet has a CRC error, it will still store the packet in the FIFO and set the RXPKTRDY bit (bit 0 of PERI_RXCSR) and the DATAERROR bit (bit 3 of PERI_RXCSR). It is left up to the application how this error condition is handled.

31.2.7.2 USB Controller Host Mode Operation

- *Entry into Suspend mode*. When operating as a host, the controller can be prompted to enter Suspend mode by setting the SUSPENDM bit in the POWER register. When this bit is set, the controller will complete the current transaction then stop the transaction scheduler and frame counter. No further transactions will be started and no SOF packets will be generated. If the ENSUSPM bit (bit 0 of POWER register) is set, PHY will go into low-power mode when the controller enters Suspend mode.
- *Sending Resume Signaling.* When the application requires the controller to leave Suspend mode, it must clear the SUSPENDM bit in the POWER register (bit 1), set the RESUME bit (bit 2) and leave it set for 20ms. While the RESUME bit is high, the controller will generate Resume signaling on the bus. After 20 ms, the application should clear the Resume bit, at which point the frame counter and transaction scheduler will be started.
- *Responding to Remote Wake-up.* If Resume signaling is detected from the target while the controller is in Suspend mode, the PHY will be brought out of low-power mode. The controller will then exit Suspend mode and automatically set the RESUME bit in the POWER register (bit 2) to take over generating the Resume signaling from the target. If the Resume interrupt is enabled, an interrupt will be generated.
- *Reset Signaling.* If the RESET bit in the POWER register (bit 3) is set while the controller is in Host mode, it will generate Reset signaling on the bus. If the HSENAB bit in the POWER register (bit 5) was set, it will also try to negotiate for high-speed operation. The software should keep the RESET bit set for at least 20 ms to ensure correct resetting of the target device. After the software has cleared the bit, the controller will start its frame counter and transaction scheduler. Whether high-speed operation is selected will be indicated by HSMODE bit of POWER register (bit 4).

31.2.7.2.1 Control Transactions

Host Control Transactions are conducted through Endpoint 0 and the software is required to handle all the Standard Device Requests that may be sent or received via Endpoint 0 (as described in Universal Serial Bus Specification, Revision 2.0).

As for a USB peripheral device, there are three categories of Standard Device Requests to be handled: Zero Data Requests (in which all the information is included in the command), Write Requests (in which the command will be followed by additional data), and Read Requests (in which the device is required to send data back to the host).

1. Zero Data Requests consist of a SETUP command followed by an IN Status Phase
2. Write Requests consist of a SETUP command, followed by an OUT Data Phase which is in turn followed by an IN Status Phase
3. Read Requests consist of a SETUP command, followed by an IN Data Phase which is in turn followed by an OUT Status Phase

A timeout may be set to limit the length of time for which the controller will retry a transaction which is continually NAKed by the target. This limit can be between 2 and 215 frames/ microframes and is set through the HOST_NAKLIMIT0 register. The following sections describe the CPU actions required for these different types of requests by examining the steps to take in the different Control Transaction phases.

Copyright © 2013–2016, Texas Instruments Incorporated

31.2.7.2.1.1 Setup Phase

For the SETUP Phase of a control transaction (Figure 31-10), the software driving the USB host device needs to:

1. Load the 8 bytes of the required Device request command into the Endpoint 0 FIFO.
2. Set SETUPPKT and TXPKTRDY (bits 3 and 1 of HOST_CSR0, respectively).

   NOTE: These bits must be set together.

   The controller then proceeds to send a SETUP token followed by the 8-byte command to Endpoint 0 of the addressed device, retrying as necessary. (On errors, controller retries the transaction three times.)
3. At the end of the attempt to send the data, the controller will generate an Endpoint 0 interrupt. The software should then read HOST_CSR0 to establish whether the RXSTALL bit (bit 2), the ERROR bit (bit 4) or the NAK_TIMEOUT bit (bit 7) has been set.

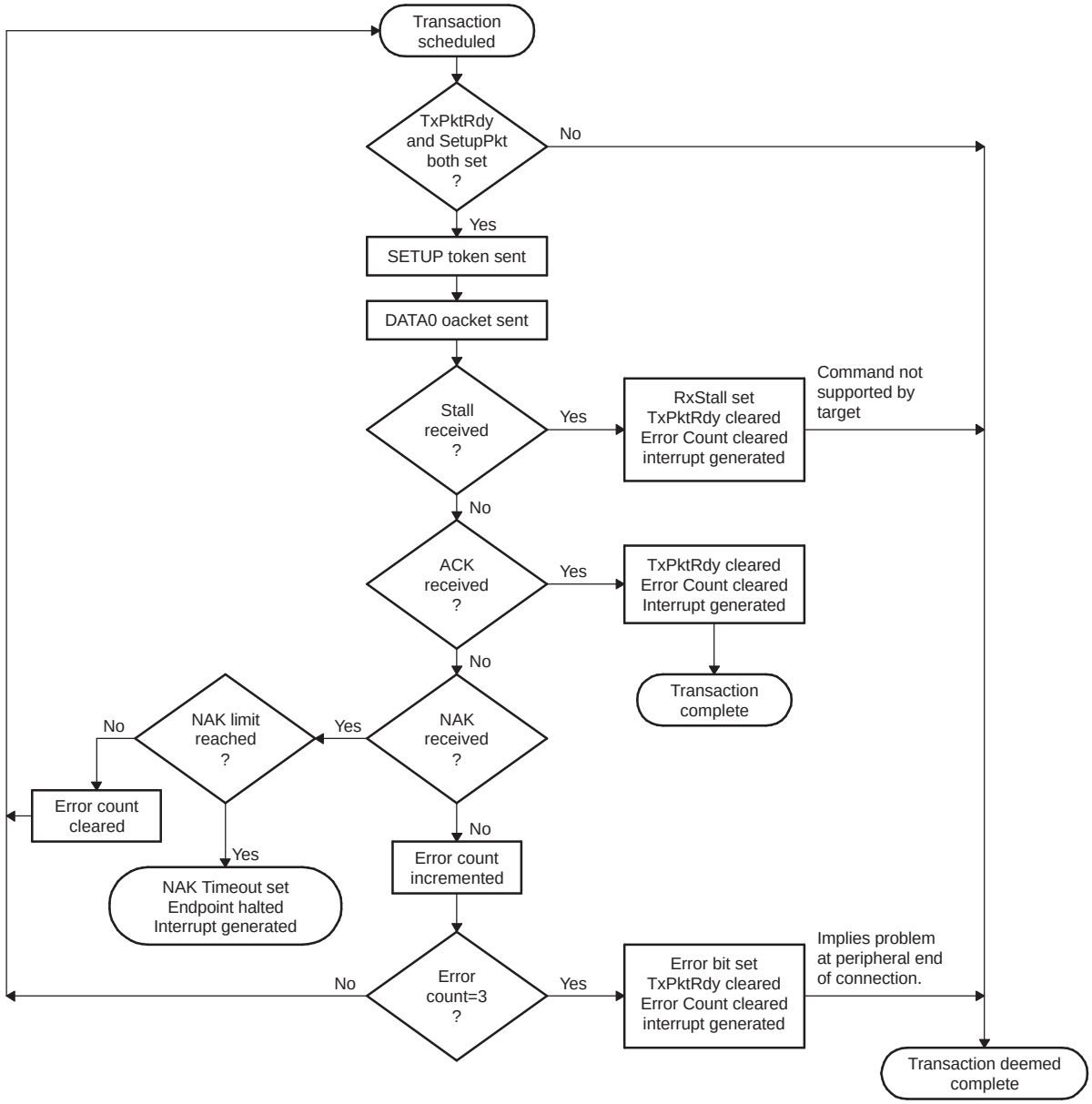
   If RXSTALL is set, it indicates that the target did not accept the command (for example, because it is not supported by the target device) and so has issued a STALL response.

   If ERROR is set, it means that the controller has tried to send the SETUP Packet and the following data packet three times without getting any response.

   If NAK_TIMEOUT is set, it means that the controller has received a NAK response to each attempt to send the SETUP packet, for longer than the time set in HOST_NAKLIMIT0. The controller can then be directed either to continue trying this transaction (until it times out again) by clearing the NAK_TIMEOUT bit or to abort the transaction by flushing the FIFO before clearing the NAK_TIMEOUT bit.
4. If none of RXSTALL, ERROR or NAK_TIMEOUT is set, the SETUP Phase has been correctly ACKed and the software should proceed to the following IN Data Phase, OUT Data Phase or IN Status Phase specified for the particular Standard Device Request.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 31-10. Setup Phase of a Control Transaction Flow Chart



Copyright © 2013–2016, Texas Instruments Incorporated

31.2.7.2.1.2 IN Data Phase

For the IN Data Phase of a control transaction (Figure 31-11), the software driving the USB host device needs to:

1. Set REQPKT bit of HOST_CSR0 (bit 5).
2. Wait while the controller sends the IN token and receives the required data back.
3. When the controller generates the Endpoint 0 interrupt, read HOST_CSR0 to establish whether the RXSTALL bit (bit 2), the ERROR bit (bit 4), the NAK_TIMEOUT bit (bit 7) or RXPKTRDY bit (bit 0) has been set.

   If RXSTALL is set, it indicates that the target has issued a STALL response.

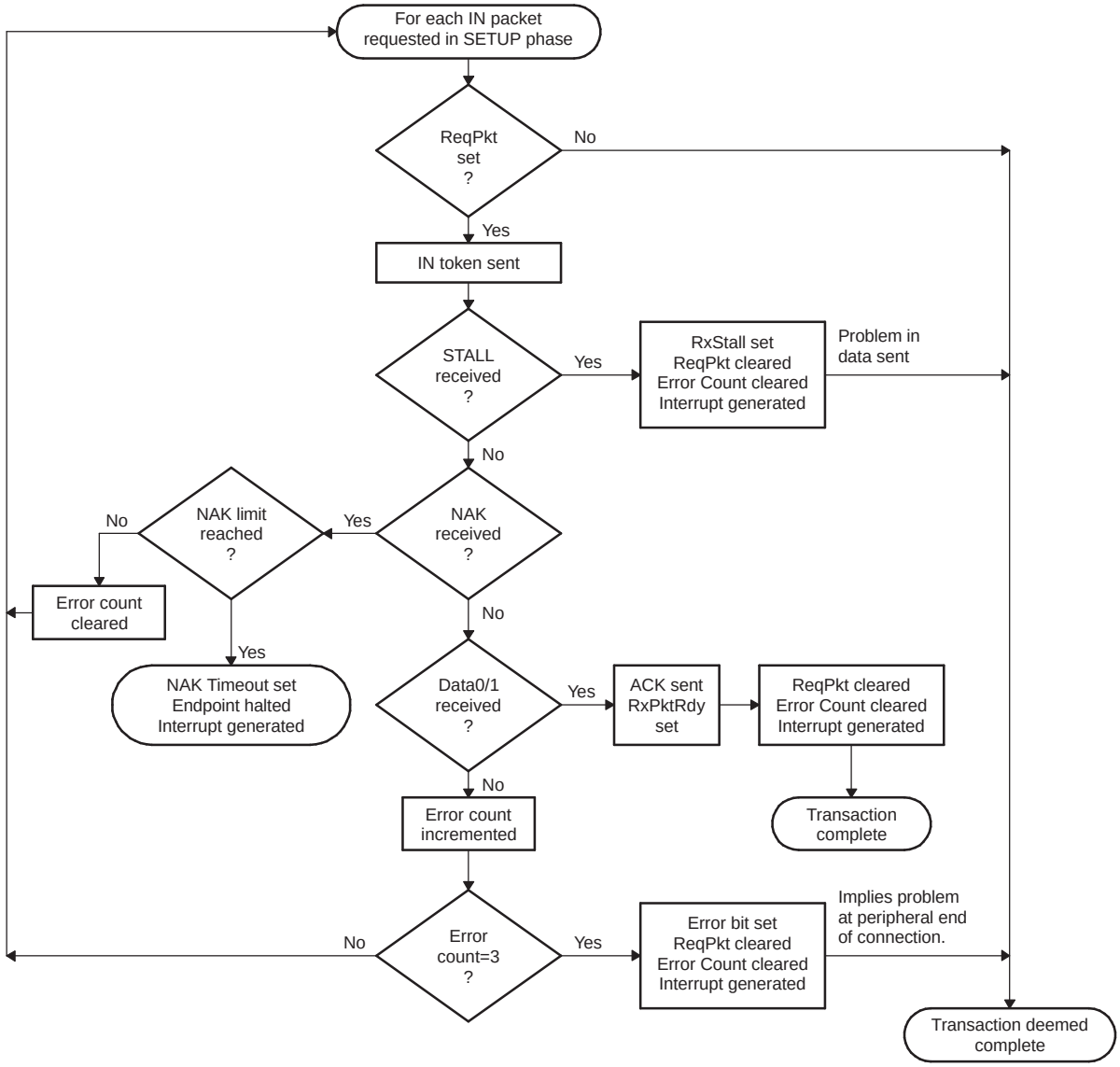
   If ERROR is set, it means that the controller has tried to send the required IN token three times without getting any response.

   If NAK_TIMEOUT bit is set, it means that the controller has received a NAK response to each attempt to send the IN token, for longer than the time set in HOST_NAKLIMIT0. The controller can then be directed either to continue trying this transaction (until it times out again) by clearing the NAK_TIMEOUT bit or to abort the transaction by clearing REQPKT before clearing the NAK_TIMEOUT bit.
4. If RXPKTRDY has been set, the software should read the data from the Endpoint 0 FIFO, then clear RXPKTRDY.
5. If further data is expected, the software should repeat Steps 1-4.

When all the data has been successfully received, the CPU should proceed to the OUT Status Phase of the Control Transaction.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 31-11. IN Data Phase Flow Chart



Copyright © 2013–2016, Texas Instruments Incorporated

31.2.7.2.1.3 OUT Data Phase

For the OUT Data Phase of a control transaction (Figure 31-12), the software driving the USB host device needs to:

1. Load the data to be sent into the endpoint 0 FIFO.
2. Set the TXPKTRDY bit of HOST_CSR0 (bit 1). The controller then proceeds to send an OUT token followed by the data from the FIFO to Endpoint 0 of the addressed device, retrying as necessary.
3. At the end of the attempt to send the data, the controller will generate an Endpoint 0 interrupt. The software should then read HOST_CSR0 to establish whether the RXSTALL bit (bit 2), the ERROR bit (bit 4) or the NAK_TIMEOUT bit (bit 7) has been set.

   If RXSTALL bit is set, it indicates that the target has issued a STALL response.

   If ERROR bit is set, it means that the controller has tried to send the OUT token and the following data packet three times without getting any response.

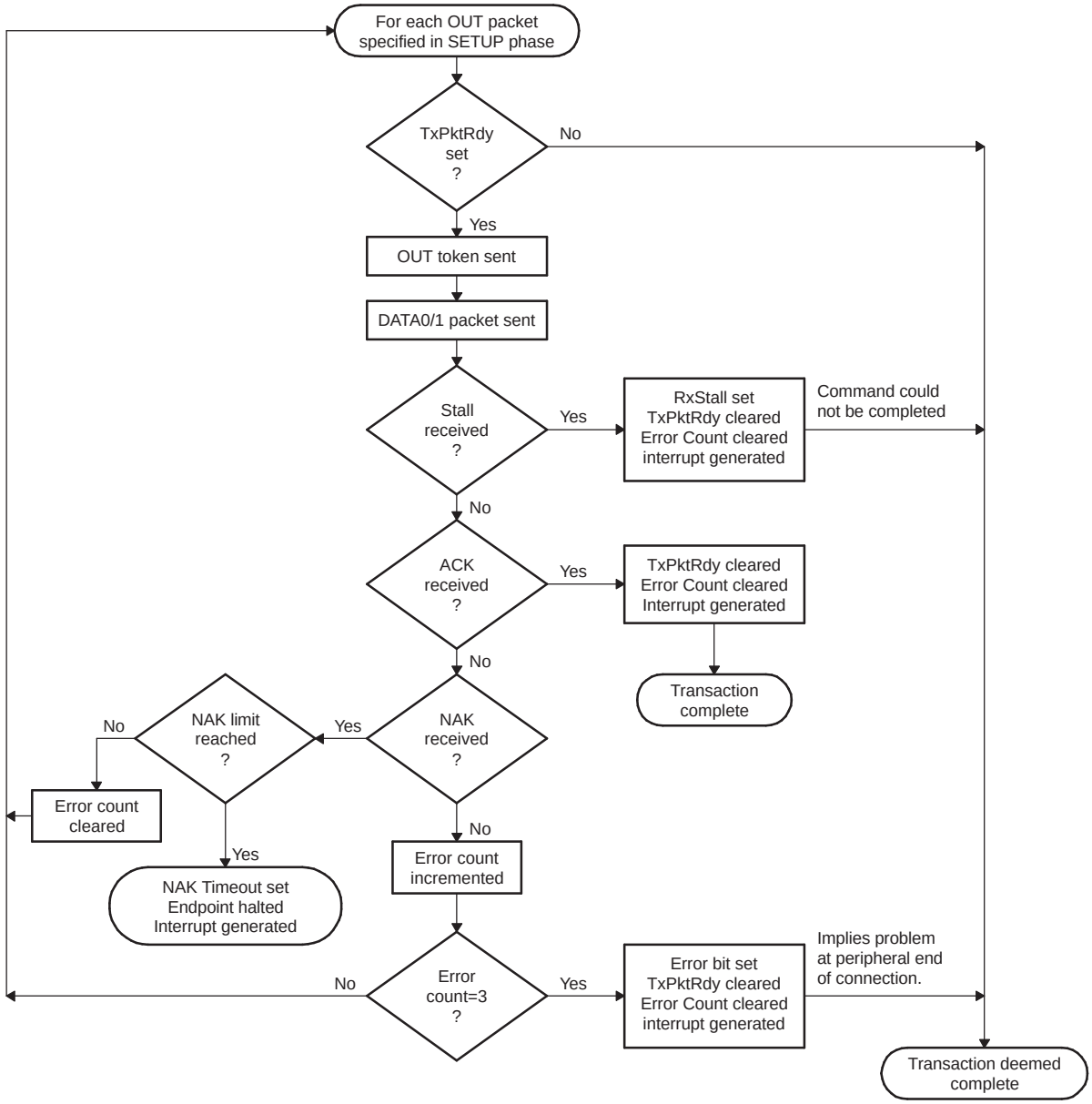
   If NAK_TIMEOUT is set, it means that the controller has received a NAK response to each attempt to send the OUT token, for longer than the time set in the HOST_NAKLIMIT0 register. The controller can then be directed either to continue trying this transaction (until it times out again) by clearing the NAK_TIMEOUT bit or to abort the transaction by flushing the FIFO before clearing the NAK_TIMEOUT bit.

   If none of RXSTALL, ERROR or NAKLIMIT is set, the OUT data has been correctly ACKed.
4. If further data needs to be sent, the software should repeat Steps 1-3.

   When all the data has been successfully sent, the software should proceed to the IN Status Phase of the Control Transaction.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 31-12. OUT Data Phase Flow Chart



Copyright © 2013–2016, Texas Instruments Incorporated

31.2.7.2.1.4 IN Status Phase (following SETUP Phase or OUT Data Phase)

For the IN Status Phase of a control transaction (Figure 31-13), the software driving the USB Host device needs to:

1. Set the STATUSPKT and REQPKT bits of HOST_CSR0 (bit 6 and bit 5, respectively).
2. Wait while the controller sends an IN token and receives a response from the USB peripheral device.
3. When the controller generates the Endpoint 0 interrupt, read HOST_CSR0 to establish whether the RXSTALL bit (bit 2), the ERROR bit (bit 4), the NAK_TIMEOUT bit (bit 7) or RXPKTRDY bit (bit 0) has been set.

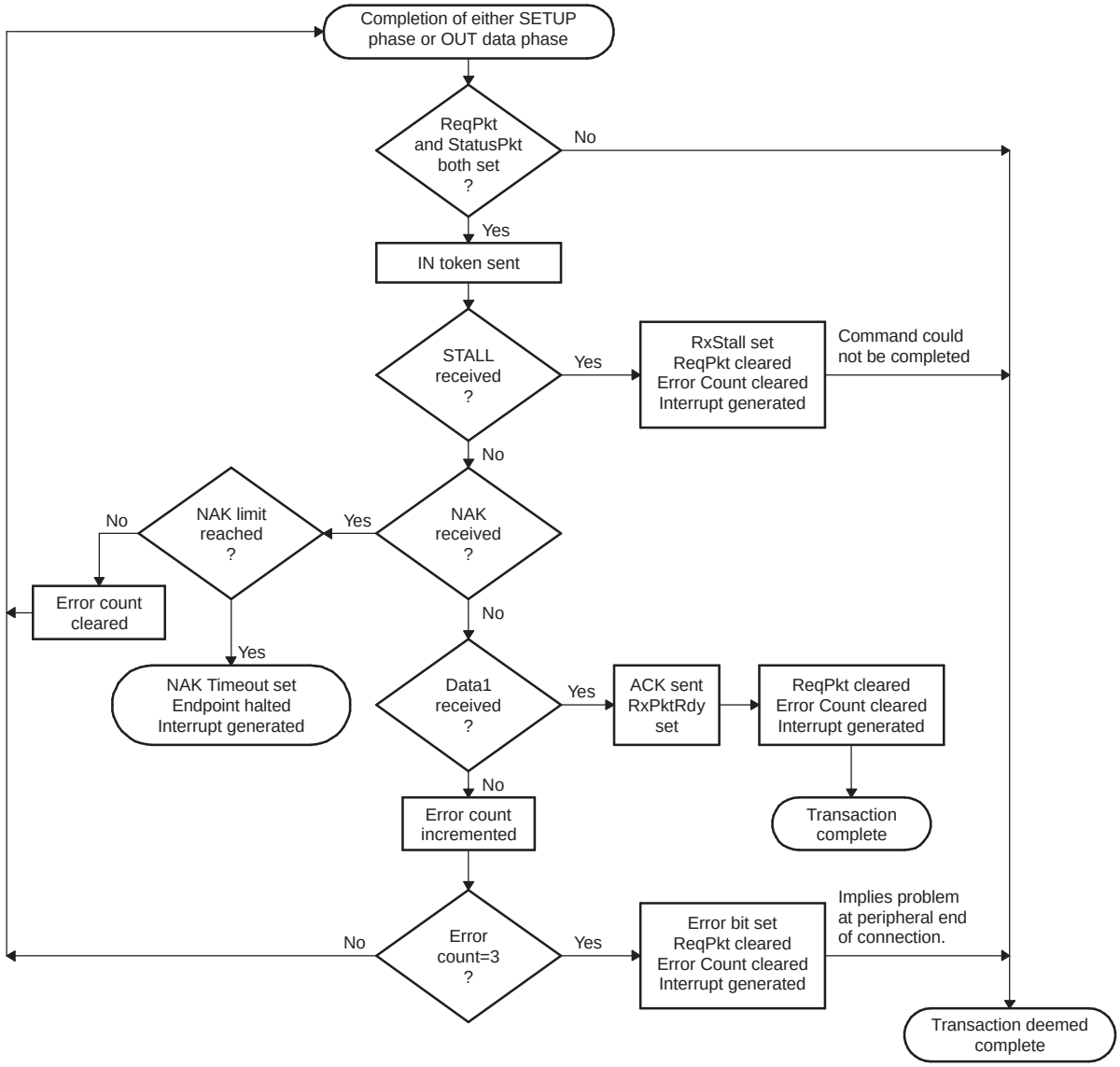
   If RXSTALL bit is set, it indicates that the target could not complete the command and so has issued a STALL response.

   If ERROR bit is set, it means that the controller has tried to send the required IN token three times without getting any response.

   If NAK_TIMEOUT bit is set, it means that the controller has received a NAK response to each attempt to send the IN token, for longer than the time set in the HOST_NAKLIMIT0 register. The controller can then be directed either to continue trying this transaction (until it times out again) by clearing the NAK_TIMEOUT bit or to abort the transaction by clearing REQPKT bit and STATUSPKT bit before clearing the NAK_TIMEOUT bit.
4. If RxPktRdy has been set, the CPU should simply clear RxPktRdy.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 31-13. Completion of SETUP or OUT Data Phase Flow Chart



Copyright © 2013–2016, Texas Instruments Incorporated

31.2.7.2.1.5 OUT Status Phase (following IN Data Phase)

For the OUT Status Phase of a control transaction (Figure 31-14) , the CPU driving the host device needs to:

1. Set STATUSPKT and TXPKTRDY bits of HOST_CSR0 (bit 6 and bit 1, respectively).

   NOTE: These bits need to be set together.

2. Wait while the controller sends the OUT token and a zero-length DATA1 packet.
3. At the end of the attempt to send the data, the controller will generate an Endpoint 0 interrupt. The software should then read HOST_CSR0 to establish whether the RXSTALL bit (bit 2), the ERROR bit (bit 4) or the NAK_TIMEOUT bit (bit 7) has been set.

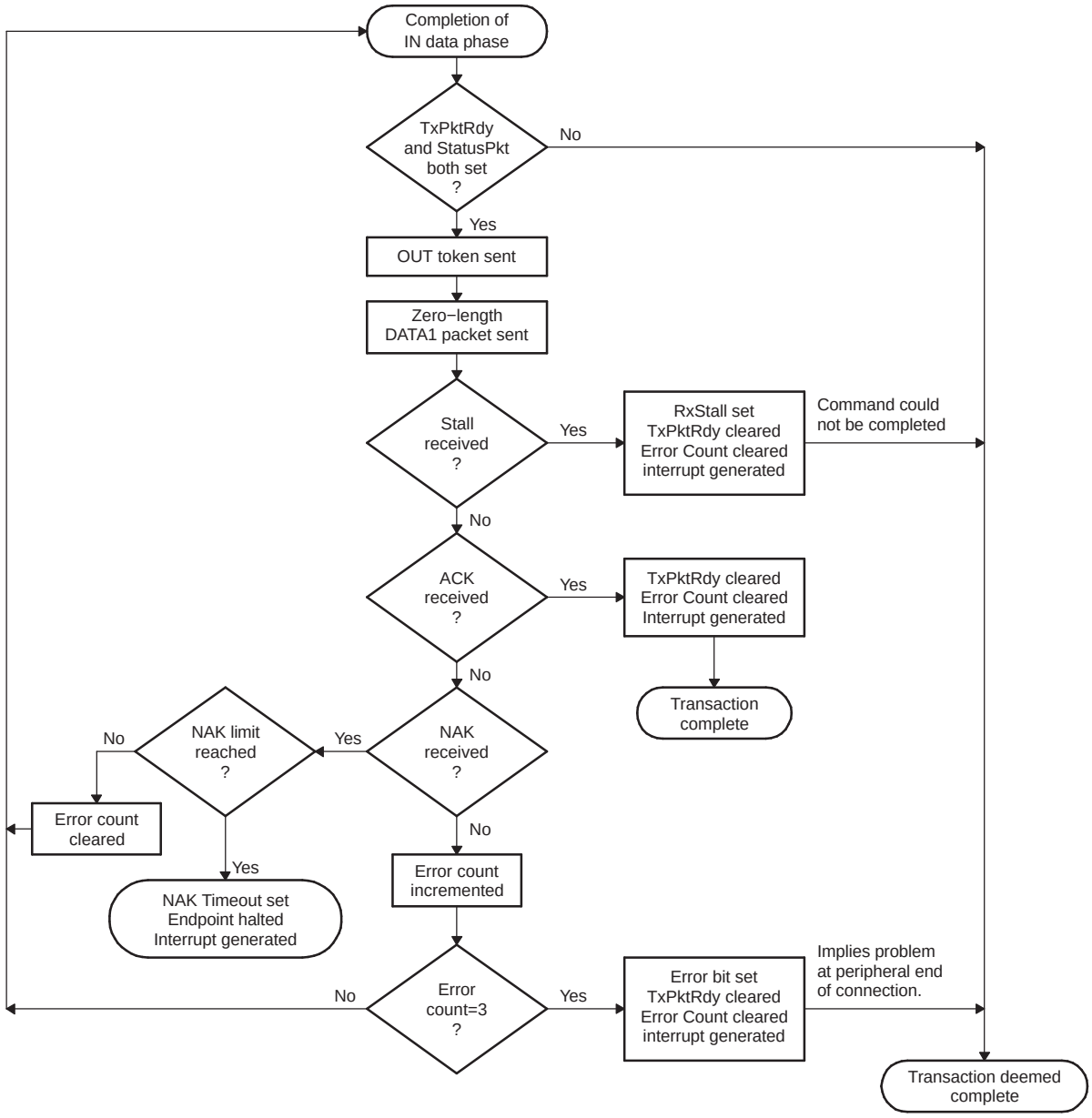
   If RXSTALL bit is set, it indicates that the target could not complete the command and so has issued a STALL response.

   If ERROR bit is set, it means that the controller has tried to send the STATUS Packet and the following data packet three times without getting any response.

   If NAK_TIMEOUT bit is set, it means that the controller has received a NAK response to each attempt to send the IN token, for longer than the time set in the HOST_NAKLIMIT0 register. The controller can then be directed either to continue trying this transaction (until it times out again) by clearing the NAK_TIMEOUT bit or to abort the transaction by flushing the FIFO before clearing the NAK_TIMEOUT bit.
4. If none of RXSTALL, ERROR or NAK_TIMEOUT bits is set, the STATUS Phase has been correctly ACKed.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 31-14. Completion of IN Data Phase Flow Chart



Copyright © 2013–2016, Texas Instruments Incorporated

31.2.7.2.2 Bulk Transactions

31.2.7.2.2.1 Bulk IN Transactions

A Bulk IN transaction may be used to transfer non-periodic data from the external USB peripheral to the host.

The following optional features are available for use with an Rx endpoint used in host mode to receive the data:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO on reception from the host. This allows that one packet can be received while another is being read. Double packet buffering is enabled by setting the DPB bit of RXFIFOSZ register (bit 4).
- DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint has a packet in its FIFO. This feature can be used to allow the DMA controller to unload packets from the FIFO without processor intervention.

  When DMA is enabled, endpoint interrupt will not be generated for completion of packet reception. Endpoint interrupt will be generated only in the error conditions.
- AutoRequest: When the AutoRequest feature is enabled, the REQPKT bit of HOST_RXCSR (bit 5) will be automatically set when the RXPKTRDY bit is cleared.

  This feature is applicable only when DMA is enabled. To enable AutoRequest feature, set the AUTOREQ register for the DMA channel associated for the endpoint.

31.2.7.2.2.1.1 Setup

Before initiating any Bulk IN Transactions in Host mode:

- The target function address needs to be set in the RXFUNCADDR register for the selected controller endpoint. (RXFUNCADDR register is available for all endpoints from EP0 to EP4.)
- The HOST_RXTYPE register for the endpoint that is to be used needs to be programmed as:
  - Operating speed in the SPEED bit field (bits 7 and 6).
  - Set 10 (binary value) in the PROT field for bulk transfer.
  - Endpoint Number of the target device in RENDPN field. This is the endpoint number contained in the Rx endpoint descriptor returned by the target device during enumeration.
- The RXMAXP register for the controller endpoint must be written with the maximum packet size (in bytes) for the transfer. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the target endpoint.
- The HOST_RXINTERVAL register needs to be written with the required value for the NAK limit (2-215 frames/microframes), or set to zero if the NAK timeout feature is not required.
- The relevant interrupt enable bit in the INTRRXE register should be set (if an interrupt is required for this endpoint).
- The following bits of HOST_RXCSR register should be set as:
  - Set DMAEN (bit 13) to 1 if a DMA request is required for this endpoint.
  - Clear DSINYET (bit 12) to 0 to allow normal PING flow control. This will affect only High Speed transactions.
  - Always clear DMAMODE (bit 11) to 0.
- If DMA is enabled, the AUTOREQ register can be set for generating IN tokens automatically after receiving the data. Set the bit field RXn_AUTOREQ (where n is the endpoint number) with binary value 01 or 11.

When the endpoint is first configured, the endpoint data toggle should be cleared to 0 either by using the DATATOGWREN and DATATOG bits of HOST_RXCSR (bit 10 and bit 9) to toggle the current setting or by setting the CLRDATATOG bit of HOST_RXCSR (bit 7). This will ensure that the data toggle (which is handled automatically by the controller) starts in the correct state. Also if there are any data packets in the FIFO (indicated by the RXPKTRDY bit (bit 0 of HOST_RXCSR) being set), they should be flushed by setting the FLUSHFIFO bit of HOST_RXCSR (bit 4).

Copyright © 2013–2016, Texas Instruments Incorporated

NOTE: It may be necessary to set this bit twice in succession if double buffering is enabled.

31.2.7.2.2.1.2 Operation

When Bulk data is required from the USB peripheral device, the software should set the REQPKT bit in the corresponding HOST_RXCSR register (bit 5). The controller will then send an IN token to the selected peripheral endpoint and waits for data to be returned.

If data is correctly received, RXPKTRDY bit of HOST_RXCSR (bit 0) is set. If the USB peripheral device responds with a STALL, RXSTALL bit (bit 6 of HOST_RXCSR) is set. If a NAK is received, the controller tries again and continues to try until either the transaction is successful or the POLINTVL_NAKLIMIT set in the HOST_RXINTERVAL register is reached. If no response at all is received, two further attempts are made before the controller reports an error by setting the ERROR bit of HOST_RXCSR (bit 2).

The controller then generates the appropriate endpoint interrupt, whereupon the software should read the corresponding HOST_RXCSR register to determine whether the RXPKTRDY, RXSTALL, ERROR or DATAERR_NAKTIMEOUT bit is set and act accordingly. If the DATAERR_NAKTIMEOUT bit is set, the controller can be directed either to continue trying this transaction (until it times out again) by clearing the DATAERR_NAKTIMEOUT bit or to abort the transaction by clearing REQPKT bit before clearing the DATAERR_NAKTIMEOUT bit.

The packets received should not exceed the size specified in the RXMAXP register (as this should be the value set in the wMaxPacketSize field of the endpoint descriptor sent to the host).

In the general case, the application software will need to read each packet from the FIFO individually. If large blocks of data are being transferred, the overhead of calling an interrupt service routine to unload each packet can be avoided by using DMA.

31.2.7.2.2.1.3 Error Handling

If the target wants to shut down the Bulk IN pipe, it will send a STALL response to the IN token. This will result in the RXSTALL bit of HOST_RXCSR (bit 6) being set.

Copyright © 2013–2016, Texas Instruments Incorporated

31.2.7.2.2.2 Host Mode: Bulk OUT Transactions

A Bulk OUT transaction may be used to transfer non-periodic data from the host to the USB peripheral.

Following optional features are available for use with a Tx endpoint used in Host mode to transmit this data:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO awaiting transmission to the peripheral device. Double packet buffering is enabled by setting the DPB bit of TXFIFOSZ register (bit 4).
- DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint is able to accept another packet in its FIFO. This feature can be used to allow the DMA controller to load packets into the FIFO without processor intervention.

  When DMA is enabled and DMAMODE bit in HOST_TXCSR register is set, an endpoint interrupt will not be generated for completion of packet reception. An endpoint interrupt will be generated only in the error conditions.

31.2.7.2.2.2.1 Setup

Before initiating any bulk OUT transactions:

- The target function address needs to be set in the TXFUNCADDR register for the selected controller endpoint. (TXFUNCADDR register is available for all endpoints from EP0 to EP4.)
- The HOST_TXTYPE register for the endpoint that is to be used needs to be programmed as:
  - Operating speed in the SPEED bit field (bits 7 and 6).
  - Set 10b in the PROT field for bulk transfer.
  - Endpoint Number of the target device in TENDPN field. This is the endpoint number contained in the OUT(Tx) endpoint descriptor returned by the target device during enumeration.
- The TXMAXP register for the controller endpoint must be written with the maximum packet size (in bytes) for the transfer. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the target endpoint.
- The HOST_TXINTERVAL register needs to be written with the required value for the NAK limit (2-215 frames/microframes), or set to zero if the NAK timeout feature is not required.
- The relevant interrupt enable bit in the INTRTXE register should be set (if an interrupt is required for this endpoint).
- The following bits of HOST_TXCSR register should be set as:
  - Set the MODE bit (bit 13) to 1 to ensure the FIFO is enabled (only necessary if the FIFO is shared with an Rx endpoint).
  - Set the DMAEN bit (bit 12) to 1 if a DMA request is required for this endpoint.
  - Clear the FRCDATATOG bit (bit 11) to 0 to allow normal data toggle operations.
  - Set the DMAMODE bit (bit 10) to 1 when DMA is enabled.

When the endpoint is first configured, the endpoint data toggle should be cleared to 0 either by using the DATATOGWREN bit and DATATOG bit of HOST_TXCSR (bit 9 and bit 8) to toggle the current setting or by setting the CLRDATATOG bit of HOST_TXCSR (bit 6). This will ensure that the data toggle (which is handled automatically by the controller) starts in the correct state. Also, if there are any data packets in the FIFO (indicated by the FIFONOTEMPTY bit of HOST_TXCSR register (bit 1) being set), they should be flushed by setting the FLUSHFIFO bit (bit 3 of HOST_TXCSR).

NOTE: It may be necessary to set this bit twice in succession if double buffering is enabled.

Copyright © 2013–2016, Texas Instruments Incorporated

31.2.7.2.2.2 Operation

When Bulk data is required to be sent to the USB peripheral device, the software should write the first packet of the data to the FIFO (or two packets if double-buffered) and set the TXPKTRDY bit in the corresponding HOST_TXCSR register (bit 0). The controller will then send an OUT token to the selected peripheral endpoint, followed by the first data packet from the FIFO.

If data is correctly received by the peripheral device, an ACK should be received whereupon the controller will clear TXPKTRDY bit of HOST_TXCSR (bit 0). If the USB peripheral device responds with a STALL, the RXSTALL bit (bit 5) of HOST_TXCSR is set. If a NAK is received, the controller tries again and continues to try until either the transaction is successful or the NAK limit set in the HOST_TXINTERVAL register is reached. If no response at all is received, two further attempts are made before the controller reports an error by setting ERROR bit in HOST_TXCSR (bit 2).

The controller then generates the appropriate endpoint interrupt, whereupon the software should read the corresponding HOST_TXCSR register to determine whether the RXSTALL (bit 5), ERROR (bit 2) or NAK_TIMEOUT (bit 7) bit is set and act accordingly. If the NAK_TIMEOUT bit is set, the controller can be directed either to continue trying this transaction (until it times out again) by clearing the NAK_TIMEOUT bit or to abort the transaction by flushing the FIFO before clearing the NAK_TIMEOUT bit.

If large blocks of data are being transferred, then the overhead of calling an interrupt service routine to load each packet can be avoided by using DMA.

31.2.7.2.2.3 Error Handling

If the target wants to shut down the Bulk OUT pipe, it will send a STALL response. This is indicated by the RXSTALL bit of HOST_TXCSR register (bit 5) being set.

31.2.7.2.3 Interrupt Transactions

When the controller is operating as the host, interactions with an Interrupt endpoint on the USB peripheral device are handled in very much the same way as the equivalent Bulk transactions (described in previous sections).

The principal difference as far as operational steps are concerned is that the PROT field of HOST_RXTYPE and HOST_TXTYPE (bits 5-4) need to be set (binary value) to represent an Interrupt transaction. The required polling interval also needs to be set in the HOST_RXINTERVAL and HOST_TXINTERVAL registers.

Copyright © 2013–2016, Texas Instruments Incorporated

31.2.7.2.4 Isochronous Transactions

31.2.7.2.4.1 Isochronous IN Transactions

An Isochronous IN transaction is used to transfer periodic data from the USB peripheral to the host.

The following optional features are available for use with an Rx endpoint used in Host mode to receive this data:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO on reception from the host. This allows that one packet can be received while another is being read. Double packet buffering is enabled by setting the DPB bit of RXFIFOSZ register (bit 4).
- DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint has a packet in its FIFO. This feature can be used to allow the DMA controller to unload packets from the FIFO without processor intervention. However, this feature is not particularly useful with isochronous endpoints because the packets transferred are often not maximum packet size.

  When DMA is enabled, endpoint interrupt will not be generated for completion of packet reception. Endpoint interrupt will be generated only in the error conditions.
- AutoRequest: When the AutoRequest feature is enabled, the REQPKT bit of HOST_RXCSR (bit 5) will be automatically set when the RXPKTRDY bit is cleared.

  This feature is applicable only when DMA is enabled. To enable AutoRequest feature, set the AUTOREQ register for the DMA channel associated for the endpoint.

31.2.7.2.4.1.1 Setup

Before initiating an Isochronous IN Transactions in Host mode:

- The target function address needs to be set in the RXFUNCADDR register for the selected controller endpoint (RXFUNCADDR register is available for all endpoints from EP0 to EP4).
- The HOST_RXTYPE register for the endpoint that is to be used needs to be programmed as:
  - Operating speed in the SPEED bit field (bits 7 and 6).
  - Set 01 (binary value) in the PROT field for isochronous transfer.
  - Endpoint Number of the target device in RENDPN field. This is the endpoint number contained in the Rx endpoint descriptor returned by the target device during enumeration.
- The RXMAXP register for the controller endpoint must be written with the maximum packet size (in bytes) for the transfer. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the target endpoint.
- The HOST_RXINTERVAL register needs to be written with the required transaction interval (usually one transaction per frame/microframe).
- The relevant interrupt enable bit in the INTRRXE register should be set (if an interrupt is required for this endpoint).
- The following bits of HOST_RXCSR register should be set as:
  - Set the DMAEN bit (bit 13) to 1 if a DMA request is required for this endpoint.
  - Clear the DISNYET it (bit 12) to 0 to allow normal PING flow control. This will only affect High Speed transactions.
  - Always clear the DMAMODE bit (bit 11) to 0.
- If DMA is enabled, AUTOREQ register can be set for generating IN tokens automatically after receiving the data. Set the bit field RX*n*_AUTOREQ (where *n* is the endpoint number) with binary value 01 or 11.

31.2.7.2.4.1.2 Operation

The operation starts with the software setting REQPKT bit of HOST_RXCSR (bit 5). This causes the controller to send an IN token to the target.

Copyright © 2013–2016, Texas Instruments Incorporated

When a packet is received, an interrupt is generated which the software may use to unload the packet from the FIFO and clear the RXPKTRDY bit in the HOST_RXCSR register (bit 0) in the same way as for a Bulk Rx endpoint. As the interrupt could occur almost any time within a frame(/microframe), the timing of FIFO unload requests will probably be irregular. If the data sink for the endpoint is going to some external hardware, it may be better to minimize the requirement for additional buffering by waiting until the end of each frame before unloading the FIFO. This can be done by using the SOF_PULSE signal from the controller to trigger the unloading of the data packet. The SOF_PULSE is generated once per frame(/microframe). The interrupts may still be used to clear the RXPKTRDY bit in HOST_RXCSR.

31.2.7.2.4.1.3 Error Handling

If a CRC or bit-stuff error occurs during the reception of a packet, the packet will still be stored in the FIFO but the DATAERR_NAKTIMEOUT bit of HOST_RXCSR (bit 3) is set to indicate that the data may be corrupt.

31.2.7.2.4.2 Isochronous OUT Transactions

An Isochronous OUT transaction may be used to transfer periodic data from the host to the USB peripheral.

Following optional features are available for use with a Tx endpoint used in Host mode to transmit this data:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO awaiting transmission to the peripheral device. Double packet buffering is enabled by setting the DPB bit of TXFIFOSZ register (bit 4).
- DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint is able to accept another packet in its FIFO. This feature can be used to allow the DMA controller to load packets into the FIFO without processor intervention.

  However, this feature is not particularly useful with isochronous endpoints because the packets transferred are often not maximum packet size.

  When DMA is enabled and DMAMODE bit in HOST_TXCSR register is set, endpoint interrupt will not be generated for completion of packet reception. Endpoint interrupt will be generated only in the error conditions.

31.2.7.2.4.2.1 Setup

Before initiating any Isochronous OUT transactions:

- The target function address needs to be set in the TXFUNCADDR register for the selected controller endpoint (TXFUNCADDR register is available for all endpoints from EP0 to EP4).
- The HOST_TXTYPE register for the endpoint that is to be used needs to be programmed as:
  - Operating speed in the SPEED bit field (bits 7 and 6).
  - Set 01 (binary value) in the PROT field for isochronous transfer.
  - Endpoint Number of the target device in TENDPN field. This is the endpoint number contained in the OUT(Tx) endpoint descriptor returned by the target device during enumeration.
- The TXMAXP register for the controller endpoint must be written with the maximum packet size (in bytes) for the transfer. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the target endpoint.
- The HOST_TXINTERVAL register needs to be written with the required transaction interval (usually one transaction per frame/microframe).
- The relevant interrupt enable bit in the INTRTXE register should be set (if an interrupt is required for this endpoint).
- The following bits of HOST_TXCSR register should be set as:
  - Set the MODE bit (bit 13) to 1 to ensure the FIFO is enabled (only necessary if the FIFO is shared with an Rx endpoint).
  - Set the DMAEN bit (bit 12) to 1 if a DMA request is required for this endpoint.
  - The FRCDATATOG bit (bit 12) is ignored for isochronous transactions.

Copyright © 2013–2016, Texas Instruments Incorporated

– Set the DMAMODE bit (bit 10) to 1 when DMA is enabled.

31.2.7.2.4.2.2 Operation

The operation starts when the software writes to the FIFO and sets TXPKTRDY bit of HOST_TXCSR (bit 0). This triggers the controller to send an OUT token followed by the first data packet from the FIFO.

An interrupt is generated whenever a packet is sent and the software may use this interrupt to load the next packet into the FIFO and set the TXPKTRDY bit in the HOST_TXCSR register (bit 0) in the same way as for a Bulk Tx endpoint. As the interrupt could occur almost any time within a frame, depending on when the host has scheduled the transaction, this may result in irregular timing of FIFO load requests. If the data source for the endpoint is coming from some external hardware, it may be more convenient to wait until the end of each frame before loading the FIFO as this will minimize the requirement for additional buffering. This can be done by using the SOF_PULSE signal from the controller to trigger the loading of the next data packet. The SOF_PULSE is generated once per frame(/microframe). The interrupts may still be used to set the TXPKTRDY bit in HOST_TXCSR.

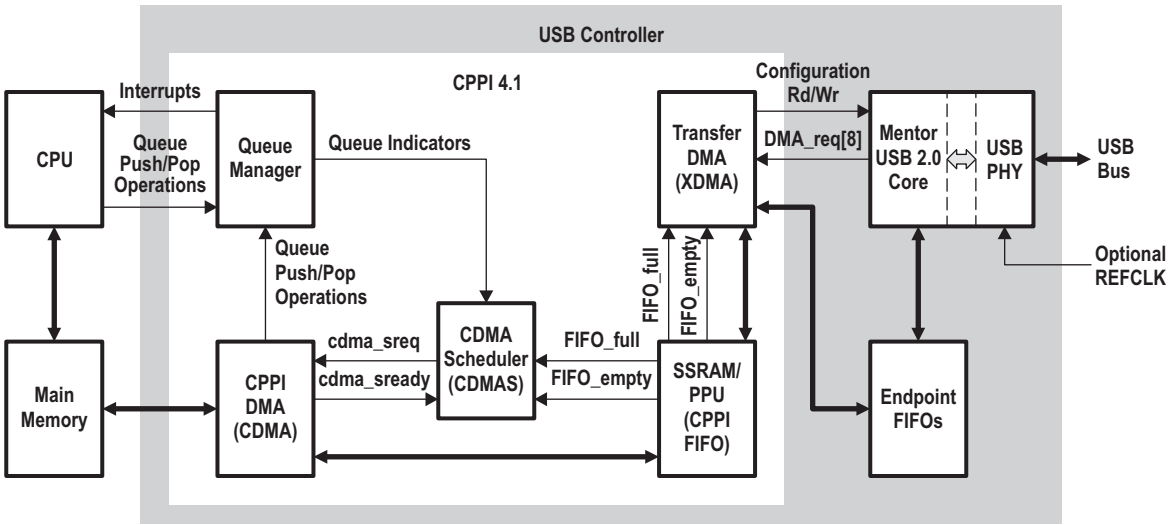
31.2.8 Communications Port Programming Interface (CPPI) 4.1 DMA Overview

The CPPI DMA module supports the transmission and reception of USB packets. The CPPI DMA is designed to facilitate the segmentation and reassembly of CPPI compliant packets to/from smaller data blocks that are natively compatible with the specific requirements of each networking port. Multiple Tx and Rx channels are provided within the DMA which allow multiple segmentation or reassembly operations to be effectively performed in parallel (but not actually simultaneously). The DMA controller maintains state information for each of the ports/channels which allows packet segmentation and reassembly operations to be time division multiplexed between channels in order to share the underlying DMA hardware. A DMA scheduler is used to control the ordering and rate at which this multiplexing occurs.

The CPPI (version 4.1) DMA controller sub-module is a common 4 dual-port DMA controller. It supports 4 Tx and 4 Rx Ports and each port attaches to the associated endpoint in the controller. Port 1 maps to endpoint 1 and Port 2 maps to endpoint 2 and Port 3 maps to endpoint 3 and Port 4 maps to endpoint 4, while endpoint 0 can not utilize the DMA and the firmware is responsible to load or offload the endpoint 0 FIFO via CPU.

Figure 31-15 displays the USB controller block diagram.

Figure 31-15. USB Controller Block Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

Host— The host is an intelligent system resource that configures and manages each communications control module. The host is responsible for allocating memory, initializing all data structures, and responding to port interrupts.

Main Memory— The area of data storage managed by the CPU. The CPPI DMA (CDMA) reads and writes CPPI packets from and to main memory. This memory can exist internal or external from the device.

Queue Manager (QM)— The QM is responsible for accelerating management of a variety of Packet Queues and Free Descriptor / Buffer Queues. It provides status indications to the CDMA Scheduler when queues are empty or full.

CPPI DMA (CDMA)— The CDMA is responsible for transferring data between the CPPI FIFO and Main Memory. It acquires free Buffer Descriptor from the QM (Receive Submit Queue) for storage of received data, posts received packets pointers to the Receive Completion Queue, transmits packets stored on the Transmit Submit Queue (Transmit Queue) , and posts completed transmit packets to the Transmit Completion Queue.

CDMA Scheduler (CDMAS)— The CDMAS is responsible for scheduling CDMA transmit and receive operations. It uses Queue Indicators from the QM and the CDMA to determine the types of operations to schedule.

CPPI FIFO— The CPPI FIFO provides 8 FIFO interfaces (one for each of the 4 transmit and receive endpoints). Each FIFO contains two 64-byte memory storage elements (ping-pong buffer storage).

Transfer DMA (XDMA)— The XDMA receives DMA requests from the Mentor USB 2.0 Core and initiates DMAs to the CPPI FIFO.

Endpoint FIFOs— The Endpoint FIFOs are the USB packet storage elements used by the Mentor USB 2.0 Core for packet transmission or reception. The XDMA transfers data between the CPPI FIFO and the Endpoint FIFOs for transmit operations and between the Endpoint FIFOs and the CPPI FIFO for receive operations.

Mentor USB 2.0 Core— This controller is responsible for processing USB bus transfers (control, bulk, interrupt, and isochronous). It supports 4 transmit and 4 receive endpoints in addition to endpoint 0 (control).

31.2.8.1 CPPI Terminology

The following terms are important in the discussion of DMA CPPI.

Port— A port is the communications module (peripheral hardware) that contains the control logic for Direct Memory Access for a single transmit/receive interface or set of interfaces. Each port may have multiple communication channels that transfer data using homogenous or heterogeneous protocols. A port is usually subdivided into transmit and receive pairs which are independent of each other. Each endpoint, excluding endpoint 0, has its own dedicated port.

Channel— A channel refers to the sub-division of information (flows) that is transported across ports. Each channel has associated state information. Channels are used to segregate information flows based on the protocol used, scheduling requirements (example: CBR, VBR, ABR), or concurrency requirements (that is, blocking avoidance). All four ports have dedicated single channels, channel 0, associated for their use in a USB application.

Data Buffer— A data buffer is a single data structure that contains payload information for transmission to or reception from a port. A data buffer is a byte aligned contiguous block of memory used to store packet payload data. A data buffer may hold any portion of a packet and may be linked together (via descriptors) with other buffers to form packets. Data buffers may be allocated anywhere within the 32-bit memory space. The Buffer Length field of the packet descriptor indicates the number of valid data bytes in the buffer. There may be from 1 to 4M-1 valid data bytes in each buffer.

Host Buffer Descriptor— A buffer descriptor is a single data structure that contains information about one or more data buffers. This type of descriptor is required when more than one descriptor is needed to define an entire packet, i.e., it either defines the middle of a packet or end of a packet.

Copyright © 2013–2016, Texas Instruments Incorporated

Host Packet Descriptor— A packet descriptor is another name for the first buffer descriptor within a packet. Some fields within a data buffer descriptor are only valid when it is a packet descriptor including the tags, packet length, packet type, and flags. This type of descriptor is always used to define a packet since it provides packet level information that is useful to both the ports and the Host in order to properly process the packet. It is the only descriptor used when single descriptor solely defines a packet. When multiple descriptors are needed to define a packet, the packet descriptor is the first descriptor used to define a packet.

Free Descriptor/Buffer Queue— A free descriptor/buffer queue is a hardware managed list of available descriptors with pre-linked empty buffers that are to be used by the receive ports for host type descriptors. Free Descriptor/Buffer Queues are implemented by the Queue Manager.

Teardown Descriptor— Teardown Descriptor is a special structure which is not used to describe either a packet or a buffer but is instead used to describe the completion of a channel halt and teardown event. Channel teardown is an important function because it ensures that when a connection is no longer needed that the hardware can be reliably halted and any remaining packets which had not yet been transmitted can be reclaimed by the Host without the possibility of losing buffer or descriptor references (which results in a memory leak).

Packet Queue— A packet queue is hardware managed list of valid (i.e. populated) packet descriptors that is used for forwarding a packet from one entity to another for any number of purposes.

Queue Manager— The queue manager is a hardware module that is responsible for accelerating management of the packet queues. Packets are added to a packet queue by writing the 32-bit descriptor address to a particular memory mapped location in the Queue Manager module. Packets are de-queued by reading the same location for that particular queue. A single Queue Manager is used for a USB application.

NOTE: All descriptors (regardless of type) must be allocated at addresses that are naturally aligned to the smallest power of 2 that is equal to or greater than the descriptor size.

31.2.8.2 Host Packet Descriptor (SOP Descriptor)

Host Packet Descriptors are designed to be used when USB like application requires support for true, unlimited fragment count scatter/gather type operations. The Host Packet Descriptor is the first descriptor on multiple descriptors setup or the only descriptor in a single descriptors setup. The Host Packet Descriptor contains the following information:

- Indicator which identifies the descriptor as a Host Packet Descriptor (always 10h)
- Source and Destination Tags (Reserved)
- Packet Type
- Packet Length
- Protocol Specific Region Size
- Protocol Specific Control/Status Bits
- Pointer to the first valid byte in the SOP data buffer
- Length of the SOP data buffer
- Pointer to the next buffer descriptor in the packet

Host Packet Descriptors can vary in size of their defined fields from 32 bytes up to 104 bytes. Within this range, Host Packet Descriptors always contain 32 bytes of required information and may also contain 8 bytes of software specific tagging information and up to 64 bytes (indicated in 4 byte increments) of protocol specific information. How much protocol specific information (and therefore the allocated size of the descriptors) is application dependent.

The Host Packet Descriptor layout is shown in Figure 31-16 and described in Table 31-8 to Table 31-15.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 31-16. Host Packet Descriptor Layout

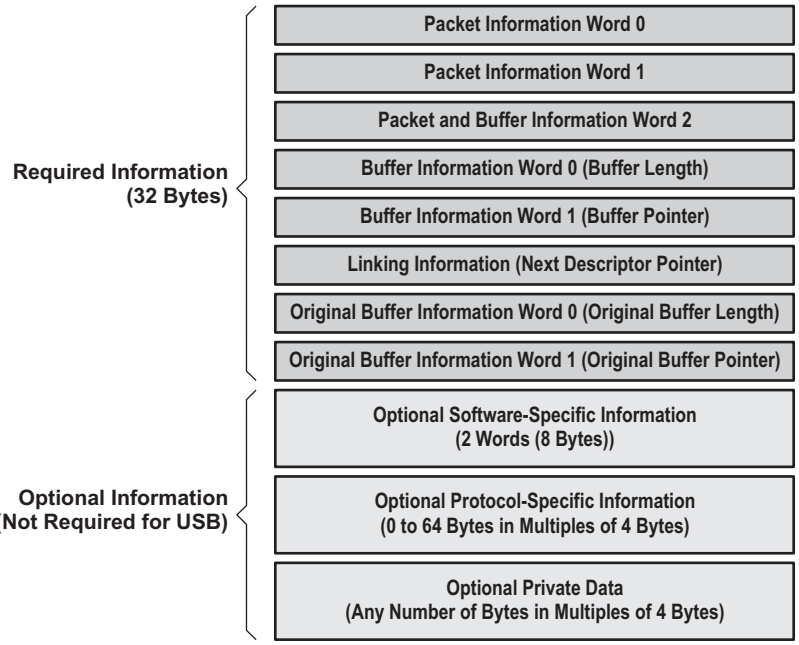


Table 31-8. Host Packet Descriptor Word 0 (HPD Word 0)

Bits	Name	Description
31-27	Descriptor Type	The Host Packet Descriptor Type is 16 decimal (10h). The CPU initializes this field.
26-22	Protocol Specific Valid Word Count	This field indicates the valid numberof 32-bit words in the protocol specific region. The CPU initializes this field. This is encoded in increments of 4 bytes as: 0 = 0 byte 1 = 4 bytes ... 16 = 64 bytes 17-31 = Reserved
21-0	Packet Length	The length of the packet in bytes. If the Packet Length is less than the sum of the buffer lengths, then the packet data will be truncated. A Packet Length greater than the sum of the buffers is an error. The valid range for the packet length is 0 to (4M - 1) bytes. The CPU initializes this field for transmitted packets; the DMA overwrites this field on packet reception.

Table 31-9. Host Packet Descriptor Word 1 (HPD Word 1)

Bits	Name	Description
31-27	Source Tag: Port #	This field indicates the port number (0-31) from which the packet originated. The DMA overwrites this field on packet reception. This is the RX Endpoint number from which the packet originated.
26-21	Source Tag: Channel #	This field indicates the channel number within the port from which the packet originated. The DMA overwrites this field on packet reception. This field is always 0-67.
20-16	Source Tag: Sub-channel #	This field indicates the sub-channel number (0-31) within the channel from which the packet originated. The DMA overwrites this field on packet reception. This field is always 0.
15-0	Destination Tag	This field is application specific. This field is always 0.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 31-10. Host Packet Descriptor Word 2 (HPD Word 2)

Bits	Name	Description
31	Packet Error	This bit indicates if an error occurred during reception of this packet (0 = No error occurred, 1 = Error occurred). The DMA overwrites this field on packet reception. Additional information about different errors may be encoded in the protocol specific fields in the descriptor.
30-26	Packet Type	This field indicates the type of this packet. The CPU initializes this field for transmitted packets; the DMA overwrites this field on packet reception. This field is encoded as: 5 = USB 8-31 = Reserved
25-20	Reserved	Reserved
19	Zero-length packet indicator	If a zero-length USB packet is received, the XDMA will send the CDMA a data block with a byte count of 0 and this bit is set. The CDMA will then perform normal EOP termination of the packet without transferring data. For transmit, if a packet has this bit set, the XDMA will ignore the CPPI packet size and send a zero-length packet to the USB controller.
18-16	Protocol Specific	This field contains protocol specific flags/information that can be assigned based on the packet type. Not used for USB.
15	Return Policy	This field indicates the return policy for this packet. The CPU initializes this field. 0 = Entire packet (still linked together) should be returned to the queue specified in bits 13-0. 1 = Each buffer should be returned to the queue specified in bits 13-0 of Word 2 in their respective descriptors. The Tx DMA will return each buffer in sequence.
14	On-chip	This field indicates whether or not this descriptor is in a region which is in on-chip memory space (1) or in external memory (0).
13-12	Packet Return Queue Mgr #	This field indicates which queue manager in the system the descriptor is to be returned to after transmission is complete. This field is not altered by the DMA during transmission or reception and is initialized by the CPU. There is only 1 Queue Manager in the USB HS/FS Device Controller, this field must always be 0.
11-0	Packet Return Queue #	This field indicates the queue number within the selected queue manager that the descriptor is to be returned to after transmission is complete. This field is not altered by the DMA during transmission or reception and is initialized by the CPU.

Table 31-11. Host Packet Descriptor Word 3 (HPD Word 3)

Bits	Name	Description
31-22	Reserved	Reserved
21-0	Buffer 0 Length	The Buffer Length field indicates how many valid data bytes are in the buffer. The CPU initializes this field for transmitted packets; the DMA overwrites this field on packet reception.

Table 31-12. Host Packet Descriptor Word 4 (HPD Word 4)

Bits	Name	Description
31-0	Buffer 0 Pointer	The Buffer Pointer is the byte aligned memory address of the buffer associated with the buffer descriptor. The CPU initializes this field for transmitted packets; the DMA overwrites this field on packet reception.

Table 31-13. Host Packet Descriptor Word 5 (HPD Word 5)

Bits	Name	Description
31-0	Next Descriptor Pointer	The 32-bit word aligned memory address of the next buffer descriptor in the packet. If the value of this pointer is zero, then the current buffer is the last buffer in the packet. The CPU initializes this field for transmitted packets; the DMA overwrites this field on packet reception.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 31-14. Host Packet Descriptor Word 6 (HPD Word 6)

Bits	Name	Description
31-22	Reserved	Reserved
21-0	Original Buffer 0 Length	The Buffer Length field indicates the original size of the buffer in bytes. This value is not overwritten during reception. This value is read by the Rx DMA to determine the actual buffer size as allocated by the CPU at initialization. Since the buffer length in Word 3 is overwritten by the Rx port during reception, this field is necessary to permanently store the buffer size information.

Table 31-15. Host Packet Descriptor Word 7 (HPD Word 7)

Bits	Name	Description
31-22	Reserved	Reserved
21-0	Original Buffer 0 Pointer	The Buffer Pointer is the byte aligned memory address of the buffer associated with the buffer descriptor. This value is not overwritten during reception. This value is read by the Rx DMA to determine the actual buffer location as allocated by the CPU at initialization. Since the buffer pointer in Word 4 is overwritten by the Rx port during reception, this field is necessary to permanently store the buffer pointer information.

31.2.8.3 Host Buffer Descriptor (Non-SOP Descriptor)

The Host Buffer Descriptor is identical in size and organization to a Host Packet Descriptor but does not include valid information in the packet level fields and does not include a populated region for protocol specific information. The packet level fields is not needed since the SOP descriptor contain this information and additional copy of this data is not needed/necessary.

Host Buffer Descriptors are designed to be linked onto a Host Packet Descriptor or another Host Buffer Descriptor to provide support for unlimited scatter / gather type operations. Host Buffer Descriptors provide information about a single corresponding data buffer. Every Host buffer descriptor stores the following information:

- Pointer to the first valid byte in the data buffer
- Length of the data buffer
- Pointer to the next buffer descriptor in the packet

Host Buffer Descriptors always contain 32 bytes of required information. Since it is a requirement that it is possible to convert a Host descriptor between a Buffer Descriptor and a Packet Descriptor (by filling in the appropriate fields) in practice, Host Buffer Descriptors will be allocated using the same sizes as Host Packet Descriptors. In addition, since the 5 LSBs of the Descriptor Pointers are used in CPPI 4.1 for the purpose of indicating the length of the descriptor, the minimum size of a descriptor is always 32 bytes. (For more information on Descriptor Size, see Section 31.4.87).

The Host Buffer Descriptor layout is shown in Figure 31-17 and described in Table 31-16 to Table 31-23.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 31-17. Host Buffer Descriptor Layout

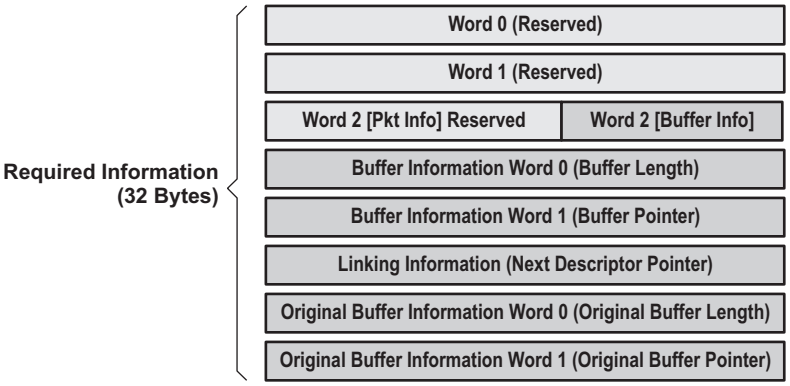


Table 31-16. Host Buffer Descriptor Word 0 (HBD Word 0)

Bits	Name	Description
31-0	Reserved	Reserved

Table 31-17. Host Buffer Descriptor Word 1 (HBD Word 1)

Bits	Name	Description
31-0	Reserved	Reserved

Table 31-18. Host Buffer Descriptor Word 2 (HBD Word 2)

Bits	Name	Description
31-15	Reserved	Reserved
14	On-chip	This field indicates whether or not this descriptor is in a region which is in on-chip memory space (1) or in external memory (0).
13-12	Packet Return Queue Mgr #	This field indicates which queue manager in the system the descriptor is to be returned to after transmission is complete. This field is not altered by the DMA during transmission or reception and is initialized by the CPU. There is only 1 Queue Manager in the USB HS/FS Device Controller, this field must always be 0.
11-0	Packet Return Queue #	This field indicates the queue number within the selected queue manager that the descriptor is to be returned to after transmission is complete. This field is not altered by the DMA during transmission or reception and is initialized by the CPU.

Table 31-19. Host Buffer Descriptor Word 3 (HBD Word 3)

Bits	Name	Description
31-22	Reserved	Reserved
21-0	Buffer 0 Length	The Buffer Length field indicates how many valid data bytes are in the buffer. The CPU initializes this field for transmitted packets; the DMA overwrites this field on packet reception.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 31-20. Host Buffer Descriptor Word 4 (HBD Word 4)

Bits	Name	Description
31-0	Buffer 0 Pointer	The Buffer Pointer is the byte aligned memory address of the buffer associated with the buffer descriptor. The CPU initializes this field for transmitted packets; the DMA overwrites this field on packet reception.

Table 31-21. Host Buffer Descriptor Word 5 (HBD Word 5)

Bits	Name	Description
31-0	Next Descriptor Pointer	The 32-bit word aligned memory address of the next buffer descriptor in the packet. If the value of this pointer is zero, then the current descriptor is the last descriptor in the packet. The CPU initializes this field for transmitted packets; the DMA overwrites this field on packet reception.

Table 31-22. Host Buffer Descriptor Word 6 (HBD Word 6)

Bits	Name	Description
31-22	Reserved	Reserved
21-0	Original Buffer 0 Length	The Buffer Length field indicates the original size of the buffer in bytes. This value is not overwritten during reception. This value is read by the Rx DMA to determine the actual buffer size as allocated by the CPU at initialization. Since the buffer length in Word 3 is overwritten by the Rx port during reception, this field is necessary to permanently store the buffer size information.

Table 31-23. Host Buffer Descriptor Word 7 (HBD Word 7)

Bits	Name	Description
31-0	Original Buffer 0 Pointer	The Buffer Pointer is the byte aligned memory address of the buffer associated with the buffer descriptor. This value is not overwritten during reception. This value is read by the Rx DMA to determine the actual buffer location as allocated by the CPU at initialization. Since the buffer pointer in Word 4 is overwritten by the Rx port during reception, this field is necessary to permanently store the buffer pointer information.

31.2.8.4 Teardown Descriptor

The Teardown Descriptor is not like the Host Packet or Buffer Descriptors since it is not used to describe either a packet or a buffer. The Teardown Descriptor is always 32 bytes long and is comprised of 4 bytes of actual teardown information and 28 bytes of pad. The Teardown Descriptor layout is shown in Figure 31-18 and described in Table 31-24 and Table 31-25. Since the 5 LSBs of the Descriptor Pointers are used in CPPI 4.1 for the purpose of indicating the length of the descriptor, the minimum size of a descriptor is 32 bytes.

The Teardown Descriptor is used to describe a channel halt and teardown event. Channel teardown ensures that when a connection is no longer needed that the hardware can be reliably halted and any remaining packets which had not yet been transmitted can be reclaimed by the Host without the possibility of losing buffer or descriptor references (which results in a memory leak).

The Teardown Descriptor contains the following information:

- Indicator which identifies the descriptor as a Teardown Packet Descriptor
- DMA Controller Number where teardown occurred
- Channel number within DMA where teardown occurred
- Indicator of whether this teardown was for the Tx or Rx channel

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 31-18. Teardown Descriptor Layout

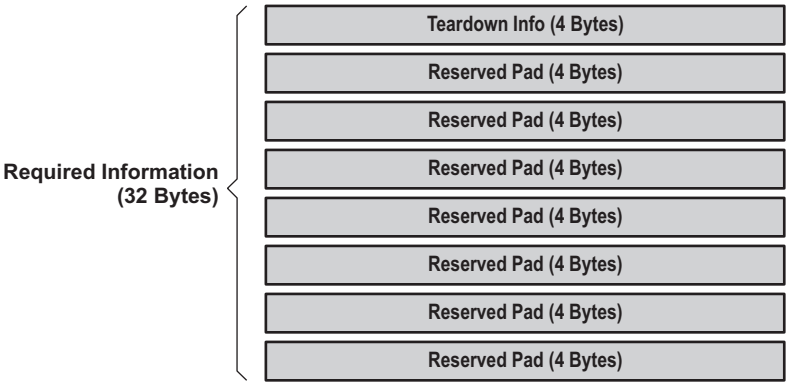


Table 31-24. Teardown Descriptor Word 0

Bits	Name	Description
31-27	Descriptor Type	The teardown descriptor type is 19 decimal (13h).
26-17	Reserved	Reserved
16	TX_RX	Indicates whether teardown is a TX (0) or RX (1).
15-10	DMA Number	Indicates the DMA number for this teardown.
9-6	Reserved	Reserved
5-0	Channel Number	Indicates the channel number within the DMA that was torn down.

Table 31-25. Teardown Descriptor Words 1-7

Bits	Name	Description
31-0	Reserved	Reserved

Teardown operation of an endpoint requires three operations. The teardown register in the CPPI DMA must be written, the corresponding endpoint bit in TEARDOWN of the USB module must be set, and the FlushFIFO bit in the Mentor USB controller Tx/RxCSR register must be set.

The following is the Transmit teardown procedure highlighting the steps required to be followed.

1. Set the TX_TEARDOWN bit in the CPPI DMA TX channel *n* global configuration register (TXGCR*n*).
2. Set the appropriate TX_TDOWN bit in the USBOTG controller's USB teardown register (TEARDOWN). Write Tx Endpoint Number to teardown to TEARDOWN[TX_TDOWN] fileld.
3. Check if the teardown descriptor has been received on the teardown queue: The completion queue (Queues 24 or 25) is usually used as the Teardown queue when the Teardown descriptor has been received, the descriptor address will be loaded onto CTRLD[24/25] register:
   (a) If not, go to step 2.
   (b) If so, go to step 4.
4. Set the appropriate TX_TDOWN bit in the USBOTG controller's USB teardown register (TEARDOWN). Set the bit corresponding to the Channel Number within TEARDOWN[TX_TDOWN] field.
5. Flush the TX FIFO in the Mentor OTG core: Set PERI_TXCSR[FLUSHFIFO] for the corresponding Endpoint.

Copyright © 2013–2016, Texas Instruments Incorporated

6. Re-enable the Tx DMA channel:
   (a) Clear TXGCR*n*[TX_TEARDOWN and TX_ENABLE] bit.
   (b) Set TXGCR*n*[TX_ENABLE] bit.

31.2.8.5 Queues

Several types of queues exist (a total of 64 queues) within the CPPI 4.1 DMA. Regardless of the type of queue a queue is, queues are used to hold pointers to host or buffer packet descriptors while they are being passed between the Host and / or any of the ports in the system. All queues are maintained within the Queue Manager module.

The following type of Queues exist:

- Receive Free Descriptor/Buffer Queue
- Receive Completion (Return) Queue
- Transmit Submit Queue (also referred as Transmit Queue)
- Transmit Completion (Return) Queue
- Free Descriptor Queue (Unassigned: Can be used for Completion or Application Specific purposes)

Table 31-26 displays the allocation (partition) of the available Queues.

Table 31-26. Allocation of Queues

Starting Queue Number	Number of Queues	Function
0	16	RX + Free Descriptor/Buffer (submit) queues
16	2	USB Endpoint 1 TX (submit) queues
18	2	USB Endpoint 2 TX (submit) queues
20	2	USB Endpoint 3 TX (submit) queues
22	2	USB Endpoint 4 TX (submit) queues
24	2	TX Completion (return) queues
26	2	RX Completion (return) queues
28	36	Unassigned (application-defined) queues

31.2.8.5.1 Queuing Packets

Prior to queuing packets, the host/firmware should construct data buffer as well host packet/buffer descriptors within memory that is external to the CPPI 4.1 DMA module.

Queuing of packets onto a packet queue is accomplished by writing a pointer to the Packet Descriptor into a specific address within the selected queue (Register D of Queue N). Packet is always queued onto the tail of the queue. The Queue Manager provides a unique set of addresses for adding packets for each queue that it manages.

31.2.8.5.2 De-Queuing Packets

De-queuing of packets from a packet queue is accomplished by reading the head packet pointer from a specific address within the selected queue (Register D of Queue N). After the head pointer has been read, the Queue Manager will invalidate the head pointer and will replace it with the next packet pointer in the queue. This functionality which is implemented in the Queue Manager prevents the ports from needing to traverse linked lists and allows for certain optimizations to be performed within the Queue Manager.

31.2.8.5.3 Type of Queues

Several types of queues exist and all are managed by the Queue Manager which is part of the CPPI 4.1 DMA. All accesses to the queues are through memory mapped registers and no external memory setup is required by the firmware.

Copyright © 2013–2016, Texas Instruments Incorporated

31.2.8.5.3.1 Receive Free Descriptor/Buffer (Submit) Queue

Receive ports use queues referred to as "receive free descriptor / buffer queues" to forward completed receive packets to the host or another peer port entity. The entries on the Free Descriptor / Buffer Queues have pre-attached empty buffers whose size and location are described in the "original buffer information" fields in the descriptor. The host is required to allocate both the descriptor and buffer and pre-link them prior to adding (submitting) a descriptor to one of the available receive free descriptor / buffer queue. The first 16 queues (Queue 0 up to Queue 15) are reserved for all four receive ports to handle incoming packets.

31.2.8.5.3.2 Transmit (Submit) Queue

Transmit ports use packet queues referred to as "transmit (submit) queues" to store the packets that are waiting to be transmitted. Each port has dedicated queues (2 queues per port) that are reserved exclusively for a use by a single port. Multiple queues per port/channel are allocated to facilitate Quality of Service (QoS) for applications that require QoS. Queue 16 and 17 are allocated for port 1, Queue 18 and 19 are allocated for port 2 and Queue 20 and Queue 21 are allocated for port 3 and Queue 22 and 23 are allocated for port 4.

31.2.8.5.3.3 Transmit Completion Queue

Transmit ports also use packet queues referred to as "transmit completion queues" to return packets to the host after they have been transmitted. Even though, non-allocated queues can be used for this purpose, a total of two dedicated queues (Queue 24 and Queue 25), that is to be shared amongst all four transmit ports, have been reserved for returning transmit packets after end of transmit operation when the firmware desires to receive interrupt when transmission completes.

31.2.8.5.3.4 Receive Completion Queue

Receive ports also use packet queues referred to as "receive completion queues" to return packets to the port after they have been received. Even though, non-allocated queues can be used for this purpose, a total of two dedicated queues (Queue 26 and Queue 27), that is to be shared amongst all four transmit ports, have been reserved for returning received packets to the receive ports after end of receive operation when the firmware desires to receive interrupt when transmission completes.

31.2.8.5.3.5 Unassigned (Application Defined) Queue

Thirty-six additional queues (Queue 28 to Queue 63) exist that have not been dedicated for exclusive use. The user can use these queues as a Completion Queues or Free Descriptor/Buffer queue.

When these queues are used as Completion Queues, interrupt will not be generated. However, the queues will have the list of descriptor pointers for the packets that have completed transmission or reception. The firmware can use polling method by continually performing the de-queuing technique onto the particular unassigned queue used to identify if the reception or transmission has completed.

When unassigned queues are used as free descriptor/buffer queue, the user can use these queues to queue/store available descriptors for future receive and transmit operations by the firmware popping the respective assigned queue and retrieving and populating descriptor prior to submitting the updated descriptor.

31.2.8.5.3.6 Teardown Queue

The Teardown Queue is used by the DMA to communicate a completion of a channel teardown after a channel teardown is invoked on to a channel. The pointer to the teardown descriptor is written to the teardown queue, which is also the Completion Queue, when the channel teardown completes.

31.2.8.5.3.7 Diverting Queue Packets from one Queue to Another

The host can move the entire contents of one queue to another queue by writing the source queue number and the destination queue number to the Queue Diversion Register. When diverting packets, the host can choose whether the source queue contents should be pushed onto the tail of the destination queue.

Copyright © 2013–2016, Texas Instruments Incorporated

31.2.8.6 Memory Regions and Linking RAM

In addition to allocating memory for raw data, the host is responsible for allocating additional memory for exclusive use of the CPPI DMA Queue Manager to be used as a scratch PAD RAM. The Queue Manager uses this memory to manage states of Descriptors submitted within the submit queues. In other words, this memory needs not to be managed by your software and your software responsibility is only for allocation of memory. The allocated memory can be a single block of memory that is contiguous or two blocks of memory that are not contiguous. These two blocks of memory are referred as a Linking RAM Regions and should not be confused with Memory Regions that are used to store Descriptors and the use of the term Region should be used in the context of its use.

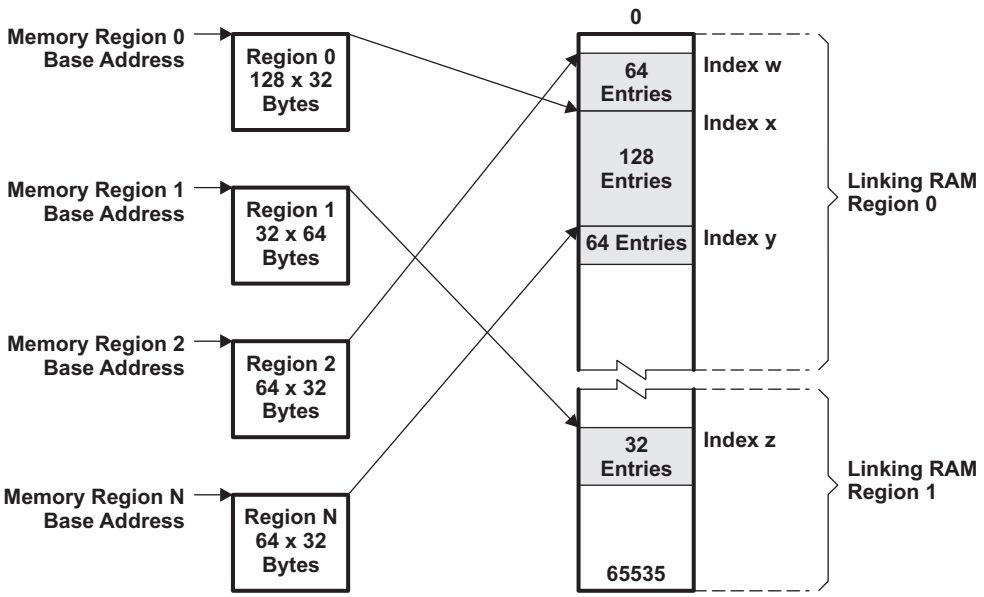
The physical size of the Linking RAM region(s) to be allocated depends on the total number of Descriptors defined within all memory regions. A minimum of four bytes of memory needs to be allocated for each Descriptor defined within all 16 Memory Regions.

The Queue Manager has the capability of managing up to 16 Memory Regions. These Memory Regions are used to store descriptors of variable sizes. The total number of Descriptors that can be managed by the Queue Manager should not exceed 16K. Each Memory Region has Descriptors of one configurable size, that is, Descriptors with different sizes cannot be housed within a single Memory Region. These 16K Descriptors are referenced internally in the Queue Manager by a 16-bit quantity index.

The information about the Linking RAM regions and the size that are allocated is communicated to the CPPI DMA via three registers dedicated for this purpose. Two of the three registers are used to store the 32-bit aligned start addresses of the Linking RAM regions. The remaining one register is used to store the size of the first Linking RAM. The size value stored here is the number of Descriptors that is to be managed by the Queue Manager within that region not the physical size of the buffer, which is four times the number of descriptors.

Note that you are not required to use both Linking RAM Regions, if the size of the Linking RAM for the first Region is large enough to accommodate all Descriptors defined. No Linking RAM size register for Linking RAM Region 2 exists. The size of the second Linking RAM, when used, is indirectly computed from the total number of Descriptors defined less the number of Descriptors managed by the first Linking RAM.

Figure 31-19. Relationship Between Memory Regions and Linking RAM



Copyright © 2013–2016, Texas Instruments Incorporated

31.2.8.7 Zero Length Packets

A special case is the handling of null packets with the CPPI 4.1 compliant DMA controller. Upon receiving a zero length USB packet, the XFER DMA will send a data block to the DMA controller with byte count of zero and the zero byte packet bit of INFO Word 2 set. The DMA controller will then perform normal End of Packet termination of the packet, without transferring data.

If a zero-length USB packet is received, the XDMA will send the CDMA a data block with a byte count of 0 and this bit set. The CDMA will then perform normal EOP termination of the packet without transferring data. For transmit, if a packet has this bit set, the XDMA will ignore the CPPI packet size and send a zero-length packet to the USB controller.

31.2.8.8 CPPI DMA Scheduler

The CPPI DMA scheduler is responsible for controlling the rate and order between the different Tx and Rx threads that are provided in the CPPI DMA controller. The scheduler table RAM exists within the scheduler.

31.2.8.8.1 CPPI DMA Scheduler Initialization

Before the scheduler can be used, the host is required to initialize and enable the block. This initialization is performed as follows:

1. The Host initializes entries within an internal memory array in the scheduler. This array contains up to 256 entries (4 entries per table word *n*) and each entry consists of a DMA channel number and a bit indicating if this is a Tx or Rx opportunity. These entries represent both the order and frequency that various Tx and Rx channels will be processed. A table size of 256 entries allows channel bandwidth to be allocated with a maximum precision of 1/256th of the total DMA bandwidth. The more entries that are present for a given channel, the bigger the slice of the bandwidth that channel will be given. Larger tables can be accommodated to allow for more precision. This array can only be written by the Host, it cannot be read.
2. If the application does not need to use the entire 256 entries, firmware can initialize the portion of the 256 entries and indicate the size of the entries used by writing onto an internal register in the scheduler which sets the actual size of the array (it can be less than 256 entries).
3. The host writes an internal register bit to enable the scheduler. The scheduler is not required to be disabled in order to change the scheduler array contents.

31.2.8.8.2 Example of Scheduler Programming

Consider a three endpoints use on a system with the following configurations: EP1-Tx, EP2-Rx, and EP2-Tx. Two assumptions are considered:

Case 1: Assume that you would like to service each enabled endpoints (EP1-Tx, EP2-Rx, and EP2-Tx) with equal priority.

The scheduler handles the rate at which an endpoint is serviced by the number of credits programmed (entries) for that particular endpoint within the scheduler Table Words. The scheduler has up to 256 credits that it can grant and for this example the number of entries/credits could be anywhere from 3 to 256. However, the optimum and direct programming for this scenario would be programming only the first three entries of the scheduler via scheduler Table WORD[0]. Since this case expects the Scheduler to use only the first three entries, you communicate that by programming DMA_SCHED_CTRL.LAST_ENTRY with 2 (that is, 3 - 1). The Enabled Endpoint numbers and the data transfer direction is then communicated by programming the first three entries of WORD[0] (ENTRY0_CHANNEL = 1: ENTRY0_RXTX = 0; ENTRY1_CHANNEL = 2: ENTRY1_RXTX = 1;ENTRY2_CHANNEL = 2: ENTRY2_RXTX = 0). With this programming, the Scheduler will only service the first three entries in a round-robin fashion, checking each credited endpoint for transfer one after the other, and servicing the endpoint that has data to transfer.

Copyright © 2013–2016, Texas Instruments Incorporated

Case 2: Enabled endpoint EP1-Tx is serviced at twice the rate as the other enabled endpoints (EP2-Rx and EP2-Tx).

The number of entries/credit that has to be awarded to EP1-Tx has to be twice as much of the others. Since only four entries/credits are required, two for EP1-Tx, one for EP2-Rx, and one for EP2-Tx, the use of scheduler Table WORD[0] would still suffice. Even though several scenarios exist to programming the order of service for this case, one scenario would be to allow EP1-Tx to be serviced back-to-back followed by the other enabled endpoints. Program DMA_SCHED_CTRL.LAST_ENTRY with 3 (that is, 4 - 1). Program WORD[0] (ENTRY0_CHANNEL = 1: ENTRY0_RXTX = 0; ENTRY1_CHANNEL = 1: ENTRY1_RXTX = 0; ENTRY2_CHANNEL = 2: ENTRY2_RXTX = 1; ENTRY3_CHANNEL = 2: ENTRY3_RXTX = 0).

31.2.8.8.3 Scheduler Operation

Once the scheduler is enabled it will begin processing the entries in the table and when appropriate passing credits to the DMA controller to perform a Tx or Rx operation. The operation of the DMA controller is as follows:

1. After the DMA scheduler is enabled it begins with the table index set to 0.
2. The scheduler reads the entry pointed to by the index and checks to see if the channel in question is currently in a state where a DMA operation can be accepted. The following must both be true:
   - The DMA channel must be enabled.
   - The CPPI FIFO that the channel talks to has free space on TX (FIFO full signal is not asserted) or a valid block on Rx (FIFO empty signal is not asserted).
3. If the DMA channel is capable of processing a credit to transfer a block, the DMA scheduler will issue that credit via the DMA scheduling interface. These are the steps:
   (a) The DMA controller may not be ready to accept the credit immediately and is provided a sched_ready signal which is used to stall the scheduler until it can accept the credit. The DMA controller only asserts the sched_ready signal when it is in the IDLE state.
   (b) Once a credit has been accepted (indicated by sched_req and sched_ready both asserted), the scheduler will increment the index to the next entry and will start at step 2.
4. If the channel in question is not currently capable of processing a credit, the scheduler will increment the index in the scheduler table to the next entry and will start at step 2.
5. When the scheduler attempts to increment its index to the value programmed in the table size register, the index will reset to 0.

31.2.8.9 CPPI DMA Transfer Interrupt Handling

The CPPI DMA 4.1 Interrupt handling mechanism does not go through the PDR interrupt handler built into the core. The DMA interrupt line is directly routed to the Interrupt Dispatcher in a PDR compliant manner. The DMA interrupt is not maskable. The firmware needs to use queues reserved by hardware as Completion Queues, if required for the DMA interrupt to be generated on a completion of a transfer.

Queues 24 and 25 are reserved by hardware for DMA transmit operations and queues 26 and 27 are reserved by hardware for DMA receive operations. If firmware uses these queues as Completion Queues, an interrupt will be generated when the transfer completes. If you need not to generate an interrupt, firmware is required to use queues that are not reserved as Completion Queues.

Copyright © 2013–2016, Texas Instruments Incorporated

31.2.8.10 DMA State Registers

The port must store and maintain state information for each transmit and receive port/channel. The state information is referred to as the Tx DMA State and Rx DMA State.

31.2.8.10.1 Transmit DMA State Registers

The Tx DMA State is a combination of control fields and protocol specific port scratchpad space used to manipulate data structures and transmit packets. Each transmit channel has two queues. Each queue has a one head descriptor pointer and one completion pointer. There are four Tx DMA State registers; one for each port/channel.

The following information is stored in the Tx DMA State:

- Tx Queue Head Descriptor Pointer(s)
- Tx Completion Pointer(s)
- Protocol specific control/status (port scratchpad)

31.2.8.10.2 Receive DMA State Registers

The Rx DMA State is a combination of control fields and protocol specific port scratchpad space used to manipulate data structures in order to receive packets. Each receive channel has only one queue. Each channel queue has one head descriptor pointer and one completion pointer. There are four Rx DMA State registers; one for each port/channel.

The following information is stored in the Rx DMA State:

- Rx Queue Head Descriptor Pointer
- Rx Queue Completion Pointer
- Rx Buffer Offset

Copyright © 2013–2016, Texas Instruments Incorporated

31.2.8.11 USB DMA Protocols Supported

Four different type of DMA transfers are supported by the CPPI 4.1 DMA; Transparent, RNDIS, Generic RNDIS, and Linux CDC. The following sections will outline the details on these DMA transfer types.

31.2.8.11.1 Transparent DMA

Transparent Mode DMA operation is the default DMA mode where DMA interrupt is generated whenever a DMA packet is transferred. In the transparent mode, DMA packet size cannot be greater than USB MaxPktSize for the endpoint. This transfer type is ideal for transfer (not packet) sizes that are less than a max packet size.

Transparent DMA Transfer Setup

The following will configure all four ports/channels for Transparent DMA Transfer type.

- Make sure that RNDIS Mode is disabled globally. RNDIS bit in the control register (CTRLR) is cleared to 0.
- Configure the DMA Mode Register (MODE) for the Endpoint field in use is programmed for Transparent Mode. MODE = 0000 0000h

31.2.8.11.2 RNDIS

RNDIS mode DMA is used for large transfers (i.e., total data size to be transferred is greater than USB MaxPktSize where the MzxPktSize is a multiple of 64 bytes) that requires multiple USB packets. This is accomplished by breaking the larger packet into smaller packets, where each packet size being USB MaxPktSize except the last packet where its size is less than USB MaxPktSize, including zero bytes. This implies that multiple USB packets of MaxPktSize will be received and transferred together as a single large DMA transfer and the DMA interrupt is generated only at the end of the complete reception of DMA transfer. The protocol defines the end of the complete transfer by receiving a short USB packet (smaller than USB MaxPktSize as mentioned in USB specification 2.0). If the DMA packet size is an exact multiple of USB MaxPktSize, the DMA controller waits for a zero byte packet at the end of complete transfer to signify the completion of the transfer.

NOTE: RNDIS Mode DMA is supported only when USB MaxPktSize is an integral multiple of 64 bytes.

RNDIS DMA Transfer Setup

The following will configure all four ports/channels for RNDIS DMA Transfer type. If all endpoints are to be configured with the same RNDIS DMA transfer type, then you can enable for RNDIS mode support from the Control Register and the content of the Mode Register will be ignored.

If you need to enable RNDIS support globally.

- Enable RNDIS Mode globally. RNDIS bit in the control register (CTRLR) is set to 1.

If you need to enable RNDIS support at the port/channel (endpoint) level.

- Disable RNDIS Mode globally. RNDIS bit in the control register (CTRLR) is cleared to 0.
- Configure the DMA Mode Register (MODE) for the Endpoint field in use is programmed for RNDIS Mode. MODE = 1111 1111h

The above two setups yield the same result.

Copyright © 2013–2016, Texas Instruments Incorporated

31.2.8.11.3 Generic RNDIS

Generic RNDIS DMA transfer mode is identical to the normal RNDIS mode in nearly all respects, except for the exception case where the last packet of the transfer can either be a short packet or the MaxPktSize. Generic RNDIS transfer makes use of a RNDIS EP Size register (there exists a register for each endpoint) that must be programmed with a value that is an integer multiple of the endpoint size for the DMA to know the end of the transfer when the last packet size is equal to the USB MaxPktSize. For example, it the Tx/RxMaxP is programmed with a value of 64, the Generic RNDIS EP Size register for that endpoint must be programmed with a value that is an integer multiple of 64 (for example, 64, 128, 192, 256, etc.).

In other words, when using Generic RNDIS mode and the DMA is tasked to transfer data transfer size that is less than a value programmed within the RNDIS EP Size register and this transfer will be resulting with a short packet, the DMA will terminate the transfer when encountering the short packet behaving exactly as the RNDIS DMA transfer type.

This means that Generic RNDIS mode will perform data transfer in the same manner as RNDIS mode, closing the CPPI packet when a USB packet is received that is less than the USB MaxPktSize size. Otherwise, the packet will be closed when the value in the Generic RNDIS EP Size register is reached.

Using RNDIS EP Size register, a packet of up to 64K bytes can be transferred. This is to allow the host software to program the USB module to transfer data that is an exact multiple of the USB MaxPktSize (Tx/RxMaxP programmed value) without having to send an additional short packet to terminate.

NOTE: As in RNDIS mode, the USB max packet size of any Generic RNDIS mode enabled endpoints must be a multiple of 64 bytes. Generic RNDIS acceleration should not be enabled for endpoints where the max packet size is not a multiple of 64 bytes. Only transparent mode should be used for such endpoints.

Generic RNDIS DMA Transfer Setup

The following will configure all four ports/channels for Generic RNDIS DMA Transfer type.

- Disable RNDIS Mode globally. RNDIS bit in the control register (CTRLR) is cleared to 0.
- Configure the DMA Mode Register (MODE) for the Endpoint field in use is programmed for Generic RNDIS Mode. MODE = 3333 3333h

31.2.8.11.4 Linux CDC

Linux CDC DMA transfer mode acts in the same manner as RNDIS packets, except for the case where the last data matches the max USB packet size. If the last data packet of a transfer is a short packet where the data size is greater than zero and less the USB MaxPktSize, then the behavior of the Linux CDC DMA transfer type is identical with the RNDIS DMA transfer type. The only exception is when the short packet length terminating the transfer is a Null Packet. In this case, instead of transferring the Null Packet, it will transfer a data packet of size 1 byte with the data value of 0x00.

In transmit operation, if an endpoint is configured or CDC Linux mode, upon receiving a Null Packet from the CPPI DMA, the XFER DMA will then generate a packet containing 1 byte of data, whose value is 0x00, indicating the end of the transfer. During receive operation, the XFER DMA will recognize the one byte zero packet as a termination of the data transfer, and sends a block of data with the EOP indicator set and a byte count of one to the CPPI DMA controller. The CPPI DMA realizing the end of the transfer termination will not update/increase the packet size count of the Host Packet Descriptor.

Linux CDC DMA Transfer Setup

The following will configure all four ports/channels for Linux CDC DMA Transfer type.

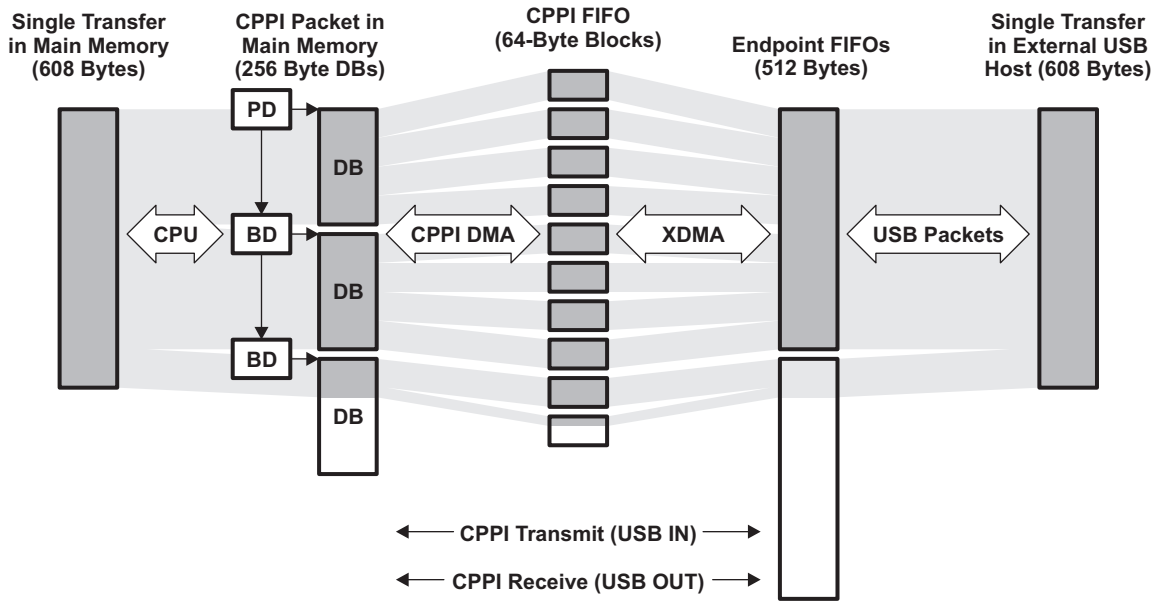
- Disable RNDIS Mode globally. RNDIS bit in the control register (CTRLR) is cleared to 0.
- Configure the DMA Mode Register (MODE) for the Endpoint field in use is programmed for Linux CDC Mode. MODE = 2222 2222h

Copyright © 2013–2016, Texas Instruments Incorporated

31.2.8.12 USB Data Flow Using DMA

The necessary steps required to perform a USB data transfer using the CPPI 4.1 DMA is expressed using an example for both transmit and receive cases. Assume a device is ready to perform a data transfer of size 608 bytes (see Figure 31-20).

Figure 31-20. High-Level Transmit and Receive Data Transfer Example



Example assumptions:

- The CPPI data buffers are 256 bytes in length.
- The USB endpoint 1 Tx and Rx endpoint 1 size are 512 bytes.
- A single transfer length is 608 bytes.
- The SOP offset is 0.

This translates to the following:

- Transmit Case:
  - 1 Host Packet Descriptor with Packet Length field of 608 bytes and a Data Buffer of size 256 Bytes linked to the 1st Host Buffer Descriptor.
  - First Host Buffer Descriptor with a Data Buffer size of 256 Bytes linked to the 2nd Buffer Descriptor.
  - Second Host Buffer Descriptor with a Data Buffer size of 96 bytes (can be greater, the Packet Descriptor contain the size of the packet) linked with its link word set to Null.
- Receive Case:
  - Two Host Buffer Descriptors with 256 bytes of Data Buffer Size
  - One Host Buffer Descriptor with 96 bytes (can be greater) of Data Buffer size

Within the rest of this section, the following nomenclature is used.

BD— Host Buffer Descriptor

DB— Data Buffer Size of 256 Bytes

PBD— Pointer to Host Buffer Descriptor

PD— Host Packet Descriptor

Copyright © 2013–2016, Texas Instruments Incorporated

PPD— Pointer to Host Packet Descriptor

RXCQ— Receive Completion Queue or Receive Return Queue (for all Rx EPs, use 26 or 27)

RXSQ— Receive Free Packet/Buffer Descriptor Queue or Receive Submit Queue. (for all Rx EPs, use 0 to 15)

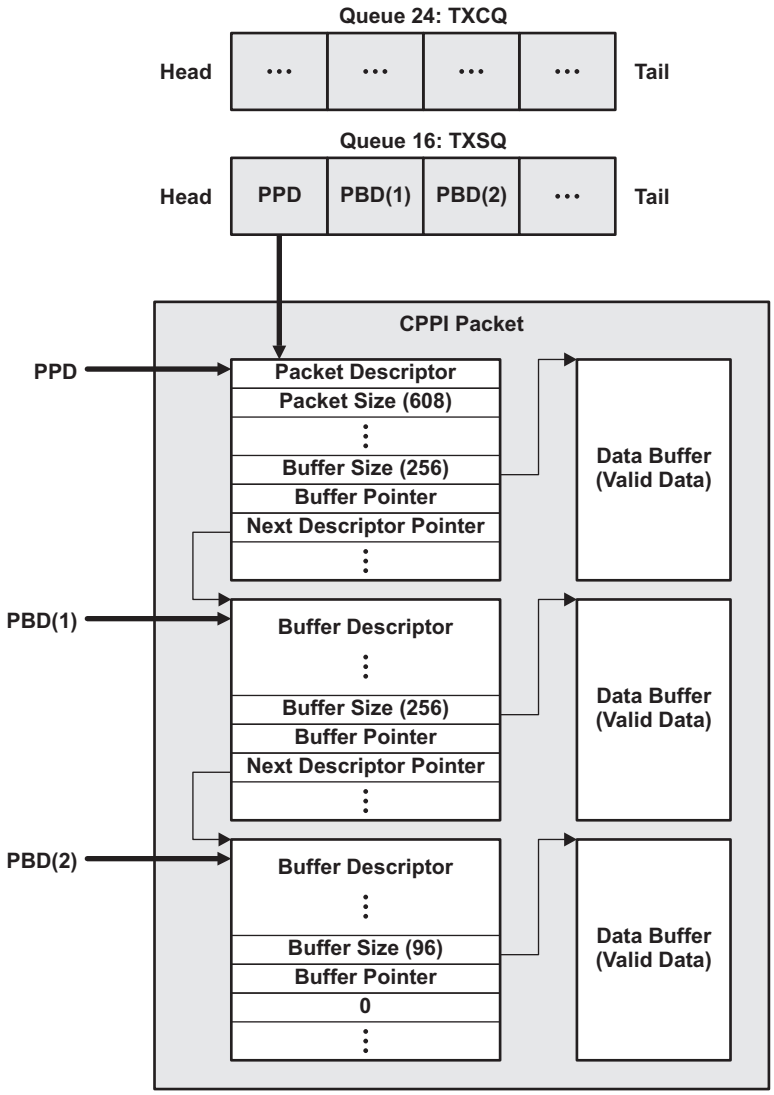
TXCQ— Transmit Completion Queue or Transmit Return Queue (for all Tx EPs, use 24 or 25)

TXSQ— Transmit Queue or Transmit Submit Queue (for EP1, use 16 or 17)

31.2.8.12.1 Transmit USB Data Flow Using DMA

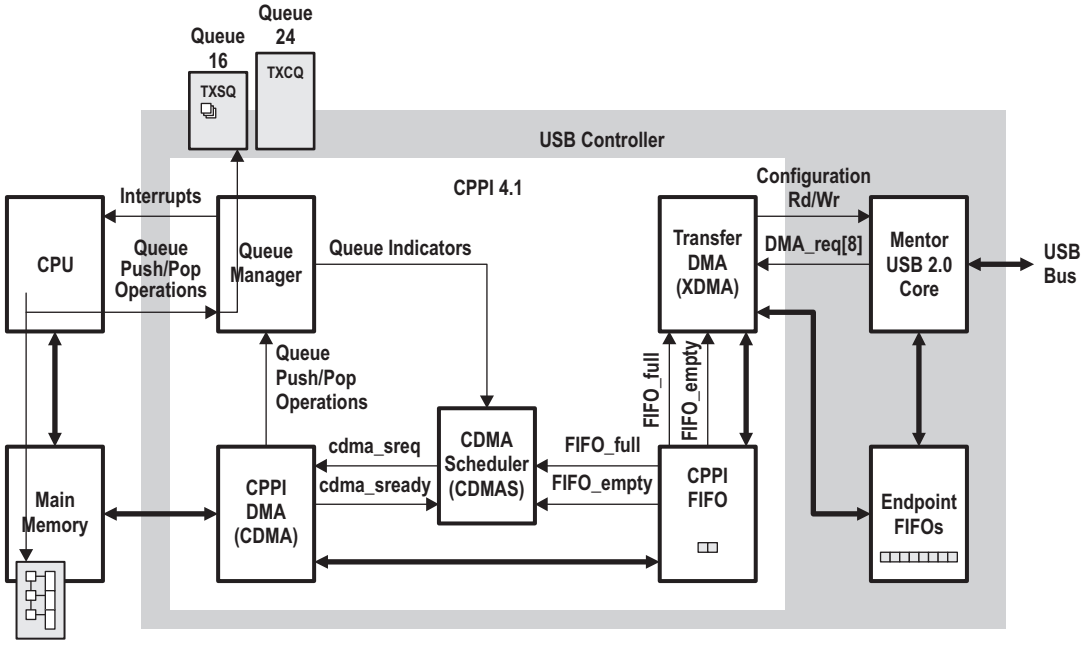
The transmit descriptors and queue status configuration prior to the transfer taking place is shown in Figure 31-21. An example of initialization for a transmit USB data flow is shown in Figure 31-22.

Figure 31-21. Transmit Descriptors and Queue Status Configuration



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 31-22. Transmit USB Data Flow Example (Initialization)



Step 1 (Initialization for Tx):

1. The CPU initializes Queue Manager with the Memory Region 0 base address and Memory Region 0 size, Link RAM0 Base address, Link RAM0 data size, and Link RAM1 Base address.
2. The CPU creates PD, BDs, and DBs in main memory and link as indicated in Figure 31-22.
3. It then initializes and configures the Queue Manager, Channel Setup, DMA Scheduler, and Mentor USB 2.0 Core.
4. It then adds (pushes) the PPD and the two PBDs to the TXSQ

NOTE: You can create more BD/DB pairs and push them on one of the unassigned queues. The firmware can pop a BD/DP pair from this chosen queue and can create its HPD or HBDs and pre link them prior to submitting the pointers to the HPD and HBD on to the TXSQ.

Step 2 (CDMA and XDMA transfers packet data into Endpoint FIFO for Tx):

1. The Queue Manager informs the CDMAS that the TXSQ is not empty.
2. CDMAS checks that the CPPI FIFO FIFO_full is not asserted, then issues a credit to the CDMA.
3. CDMA reads the packet descriptor pointer and descriptor size hint from the Queue Manager.
4. CMDA reads the packet descriptor from memory.
5. For each 64-byte block of data in the packet data payload:
   (a) The CDMA transfers a max burst of 64-byte block from the data to be transferred in main memory to the CPPI FIFO.
   (b) The XDMA sees FIFO_empty not asserted and transfers 64-byte block from CPPI FIFO to Endpoint FIFO.
   (c) The CDMA performs the above 2 steps 3 more times since the data size of the HPD is 256 bytes.
6. The CDMA reads the first buffer descriptor pointer.
7. CDMA reads the buffer descriptor from memory.
8. For each 64-byte block of data in the packet data payload:
   (a) The CDMA transfers a max burst of 64-byte block from the data to be transferred in main memory to the CPPI FIFO.

Copyright © 2013–2016, Texas Instruments Incorporated

(b) The XDMA sees FIFO_empty not asserted and transfers 64-byte block from CPPI FIFO to Endpoint FIFO.

(c) The CDMA performs the above 2 steps 2 more times since data size of the HBD is 256 bytes.

9. The CDMA reads the second buffer descriptor pointer.
10. CDMA reads the buffer descriptor from memory.
11. For each 64-byte block of data in the packet data payload:

(a) The CDMA transfers a max burst of 64-byte block from the data to be transferred in main memory to the CPPI FIFO.

(b) The XDMA sees FIFO_empty not asserted and transfers 64-byte block from CPPI FIFO to Endpoint FIFO.

(c) The CDMA transfers the last remaining 32-byte from the data to be transferred in main memory to the CPPI FIFO.

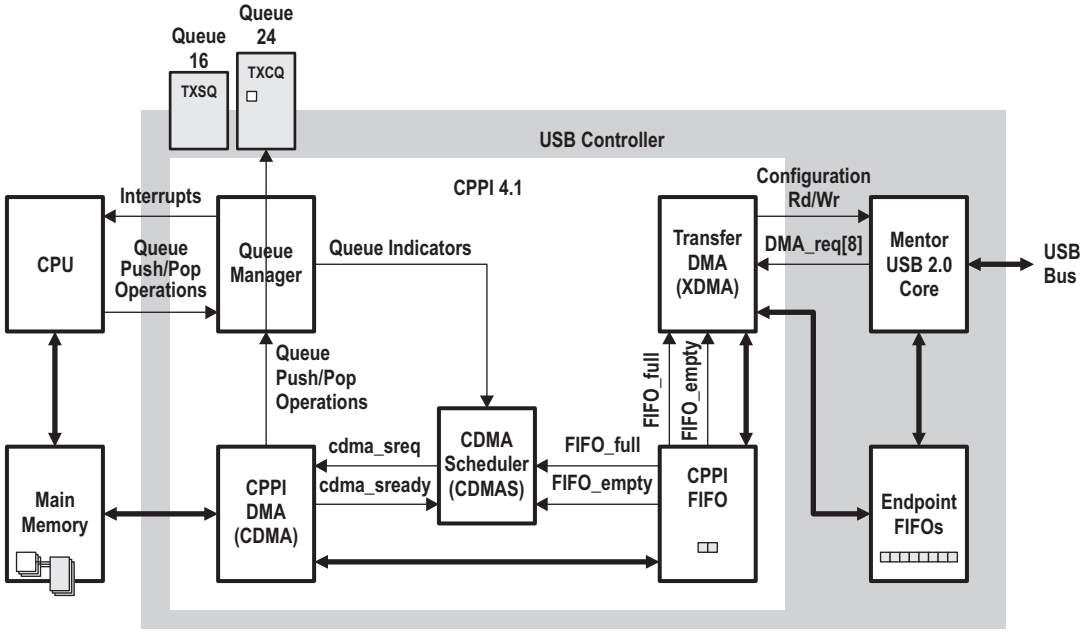
(d) The XDMA sees FIFO_empty not asserted and transfers 32-byte block from CPPI FIFO to Endpoint FIFO.

Step 3 (Mentor USB 2.0 Core transmits USB packets for Tx):

1. Once the XDMA has transferred enough 64-byte blocks of data from the CPPI FIFO to fill the Endpoint FIFO, it signals the Mentor USB 2.0 Core that a TX packet is ready (sets the endpoint's TxPktRdy bit).
2. The Mentor USB 2.0 Core will transmit the packet from the Endpoint FIFO out on the USB BUS when it receives a corresponding IN request from the attached USB Host.
3. After the USB packet is transferred, the Mentor USB 2.0 Core issues a TX DMA_req to the XDMA.
4. This process is repeated until the entire packet has been transmitted. The XDMA will also generate the required termination packet depending on the termination mode configured for the endpoint.

An example of the completion for a transmit USB data flow is shown in Figure 31-23.

Figure 31-23. Transmit USB Data Flow Example (Completion)



Step 4 (Return packet to completion queue and interrupt CPU for Tx):

1. After all data for the packet has been transmitted (as specified by the packet size field), the CDMA will write the pointer to the packet descriptor to the TX Completion Queue specified in the return queue manager / queue number fields of the packet descriptor.

Copyright © 2013–2016, Texas Instruments Incorporated

2. The Queue Manager then indicates the status of the TXSQ (empty) to the CDMAS and the TXCQ to the CPU via an interrupt.

31.2.8.12.2 Receive USB Data Flow Using DMA

The receive descriptors and queue status configuration prior to the transfer taking place is shown in Figure 31-24. An example of initialization for a receive USB data flow is shown in Figure 31-25.

Figure 31-24. Receive Descriptors and Queue Status Configuration

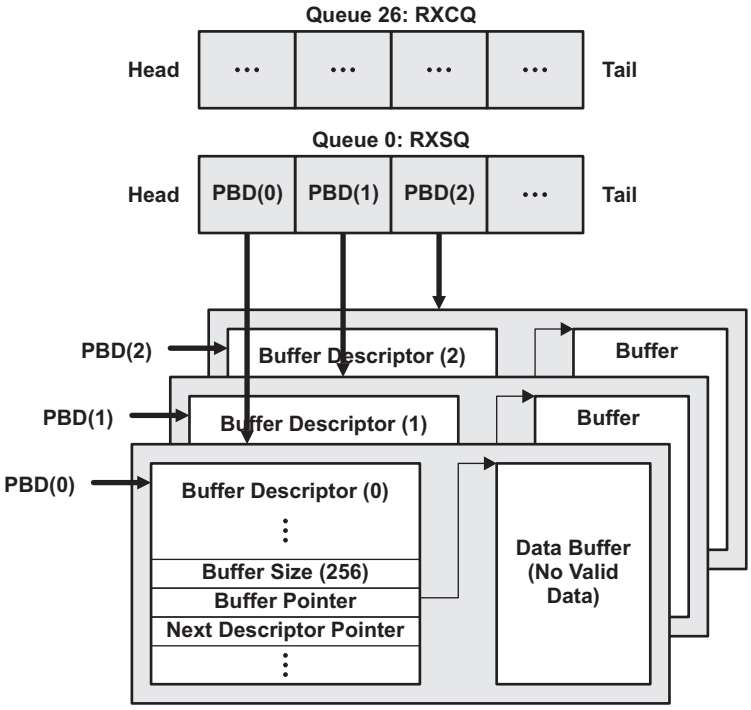
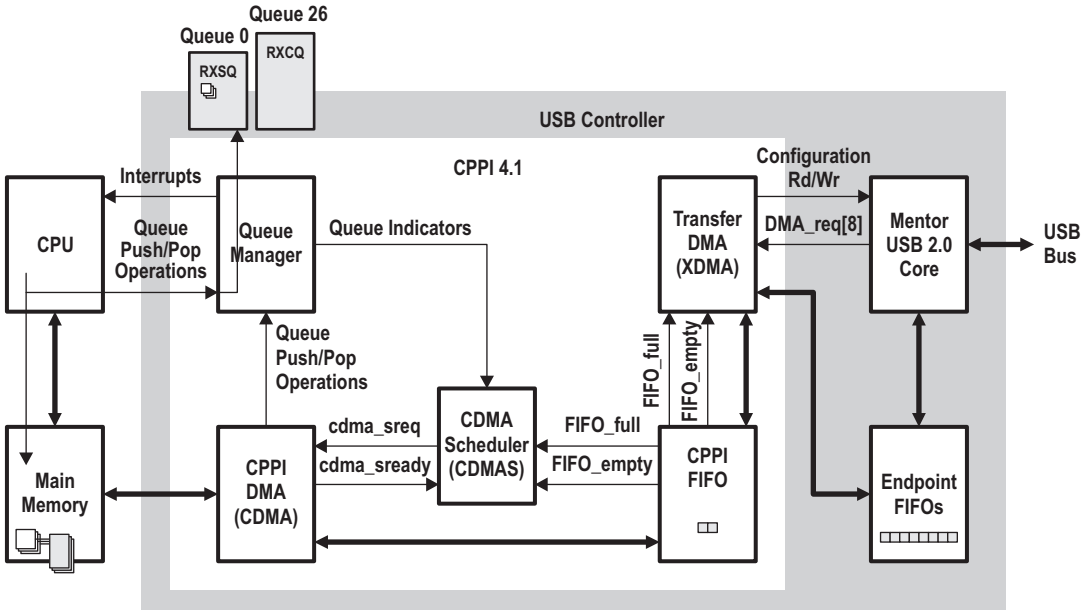


Figure 31-25. Receive USB Data Flow Example (Initialization)



Copyright © 2013–2016, Texas Instruments Incorporated

Step 1 (Initialization for Rx):

1. The CPU initializes Queue Manager with the Memory Region 0 base address and Memory Region 0 size, Link RAM0 Base address, Link RAM0 data size, and Link RAM1 Base address.
2. The CPU creates BDs, and DBs in main memory and link them as indicated in Figure 31-25.
3. It then initializes the RXCQ queue and configures the Queue Manager, Channel Setup, DMA Scheduler, and Mentor USB 2.0 Core.
4. It then adds (pushes) the address of the three PHDs into the RXSQ.

Step 2 (Mentor USB 2.0 Core receives a packet, XDMA starts data transfer for Receive):

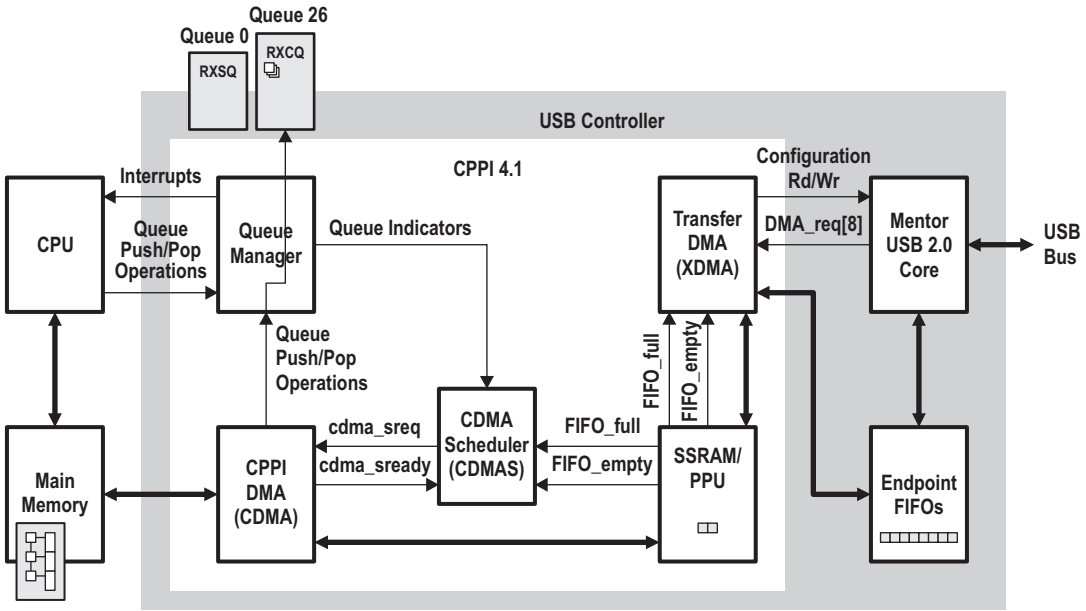
1. The Mentor USB 2.0 Core receives a USB packet from the USB Host and stores it in the Endpoint FIFO.
2. It then asserts a DMA_req to the XDMA informing it that data is available in the Endpoint FIFO.
3. The XDMA verifies the corresponding CPPI FIFO is not full via the FIFO_full signal, then starts transferring 64-byte data blocks from the Endpoint FIFO into the CPPI FIFO.

Step 3 (CDMA transfers data from SSRAM / PPU to main memory for Receive):

1. The CDMAS see FIFO_empty de-asserted (there is RX data in the FIFO) and issues a transaction credit to the CDMA.
2. The CDMA begins packet reception by fetching the first PBD from the Queue Manager using the Free Descriptor / Buffer Queue 0 (Rx Submit Queue) index that was initialized in the RX port DMA state for that channel.
3. The CDMA will then begin writing the 64-byte block of packet data into this DB.
4. The CDMA will continue filling the buffer with additional 64-byte blocks of data from the CPPI FIFO and will fetch additional PBD as needed using the Free Descriptor / Buffer Queue 1, 2, and 3 indexes for the 2nd, 3rd, and remaining buffers in the packet. After each buffer is filled, the CDMA writes the buffer descriptor to main memory.

An example of the completion for a receive USB data flow is shown in Figure 31-26.

Figure 31-26. Receive USB Data Flow Example (Completion)



Copyright © 2013–2016, Texas Instruments Incorporated

Step 4 (CDMA completes the packet transfer for Receive):

1. After the entire packet has been received, the CDMA writes the packet descriptor to main memory.
2. The CDMA then writes the packet descriptor to the RXCQ specified in the Queue Manager / Queue Number fields in the RX Global Configuration Register.
3. The Queue Manager then indicates the status of the RXCQ to the CPU via an interrupt.
4. The CPU can then process the received packet by popping the received packet information from the RXCQ and accessing the packet's data from main memory.

31.2.8.13 Interrupt Handling

Table 31-27 lists the interrupts generated by the USB controller.

Table 31-27. Interrupts Generated by the USB Controller

Interrupt	Description
Tx Endpoint [4-0]	Tx endpoint ready or error condition. For endpoints 4 to 0. (Rx and Tx for endpoint 0)
Rx Endpoint [4-1]	Rx endpoint ready or error condition. For endpoints 4 to 1. (Endpoint 0 has interrupt status in Tx interrupt)
USB Core[8-0]	Interrupts for 9 USB conditions

Whenever any of these interrupt conditions are generated, the host processor is interrupted. The software needs to read the different interrupt status registers (discussed in later section) to determine the source of the interrupt.

The nine USB interrupt conditions are listed in Table 31-28.

Table 31-28. USB Interrupt Conditions

Interrupt	Description
USB[8]	DRVVBUS level change
USB[7]	VBus voltage < VBus Valid Threshold (VBus error)
USB[6]	SRP detected
USB[5]	Device Disconnected (Valid in Host Mode)
USB[4]	Device Connected (Valid in Host Mode)
USB[3]	SOF started
USB[2]	Reset Signaling detected (In Peripheral Mode) Babble detected (In Host Mode)
USB[1]	Resume signaling detected
USB[0]	Suspend Signaling detected

31.2.8.13.1 USB Core Interrupts

Interrupt status can be determined using the INTSRCR (interrupt source) register. This register is non-masked. To clear the interrupt source, set the corresponding interrupt bit in INTCLRR register. For debugging purposes, interrupt can be set manually through INTSETR register.

The interrupt controller provides the option of masking the interrupts. A mask can be set using INTMSKSETR register and can be cleared by setting the corresponding bit in the INTMSKCLRR register. The mask can be read from INTMSKR register. The masked interrupt status is determined using the INTMASKEDR register.

The host processor software should write to the End Of Interrupt Register (EOIR) to acknowledge the completion of an interrupt.

Copyright © 2013–2016, Texas Instruments Incorporated

NOTE: While EOIR is not written, the interrupt from the USB controller remains asserted.

31.2.9 Test Modes

The USB2.0 controller supports the four USB 2.0 test modes defined for high-speed functions. It also supports an additional “FIFO access” test mode that can be used to test the operation of the CPU interface, the DMA controller (if configured), and the RAM block.

The test modes are entered by writing to the TESTMODE register. A test mode is usually requested by the host sending a SET_FEATURE request to Endpoint 0. When the software receives the request, it should wait until the Endpoint 0 transfer has completed (when it receives the Endpoint 0 interrupt indicating the status phase has completed) then write to the TESTMODE register.

NOTE: These test modes have no purpose in normal operation.

31.2.9.1 TEST_SE0_NAK

To enter the Test_SE0_NAK test mode, the software should set the TEST_SE0_NAK bit in the TESTMODE register to 1. The USB controller will then go into a mode in which it responds to any valid IN token with a NAK.

31.2.9.2 TEST_J

To enter the Test_J test mode, the software should set the TEST_J bit in the TESTMODE register to 1. The USB controller will then go into a mode in which it transmits a continuous J on the bus.

31.2.9.3 TEST_K

To enter the Test_K test mode, the software should set the TEST_K bit in the TESTMODE register to 1. The USB controller will then go into a mode in which it transmits a continuous K on the bus.

31.2.9.4 TEST_PACKET

To execute the Test_Packet, the software should:

1. Start a session (if the core is being used in Host mode).
2. Write the standard test packet (shown below) to the Endpoint 0 FIFO.
3. Write 8h to the TESTMODE register (TEST_PACKET = 1) to enter Test_Packet test mode.
4. Set the TxPktRdy bit in the CSR0 register (D1).

The 53 by test packet to load is as follows (all bytes in hex). The test packet only has to be loaded once; the USB controller will keep re-sending the test packet without any further intervention from the software.

This data sequence is defined in Universal Serial Bus Specification Revision 2.0, Section 7.1.20. The USB controller will add the DATAA0 PID to the head of the data sequence and the CRC to the end.

00	00	00	00	00	00	00	00
00	AA	AA	AA	AA	AA	AA	AA
AA	EE	EE	EE	EE	EE	EE	EE
EE	FE	FF	FF	FF	FF	FF	FF
FF	FF	FF	FF	FF	7F	BF	DF
EF	F7	FB	FD	FC	7E	BF	DF
EF	F7	FB	FD	7E			

Copyright © 2013–2016, Texas Instruments Incorporated

31.2.9.5 FIFO_ACCESS

The FIFO Access test mode allows you to test the operation of CPU Interface, the DMA controller (if configured), and the RAM block by loading a packet of up to 64 bytes into the Endpoint 0 FIFO and then reading it back out again. Endpoint 0 is used because it is a bidirectional endpoint that uses the same area of RAM for its Tx and Rx FIFOs.

NOTE: The core does not need to be connected to the USB bus to run this test. If it is connected, then no session should be in progress when the test is run.

The test procedure is as follows:

1. Load a packet of up to 64 bytes into the Endpoint 0 Tx FIFO.
2. Set CSR0.TxPktRdy.
3. Write 40h to the TESTMODE register (FIFO_ACCESS = 1).
4. Unload the packet from the Endpoint Rx FIFO, again.
5. Set CSR0.ServicedRxPktRdy.

Writing 40h to the TESTMODE register causes the following sequence of events:

1. The Endpoint 0 CPU pointer (that records the number of bytes to be transmitted) is copied to the Endpoint 0 USB pointer (that records the number of bytes received).
2. The Endpoint 0 CPU pointer is reset.
3. CSR0.TxPktRdy is cleared.
4. CSR0.RxPktRdy is set.
5. An Endpoint 0 interrupt is generated (if enabled).

The effect of these steps is to make the Endpoint 0 controller act as if the packet loaded into the Tx FIFO has flushed and the same packet received over the USB. The data that was loaded in the Tx FIFO can now be read out of the Rx FIFO.

31.2.9.6 FORCE_HOST

The Force Host test mode enables you to instruct the core to operate in Host mode, regardless of whether it is actually connected to any peripheral; that is, the state of the CID input and the LINESTATE and HOSTDISCON signals are ignored. (While in this mode, the state of the HOSTDISCON signal can be read from the BDEVICE bit in the device control register (DEVCTL).)

This mode, which is selected by writing 80h to the TESTMODE register (FORCE_HOST = 1), allows implementation of the USB Test_Force_Enable (7.1.20). It can also be used for debugging PHY problems in hardware.

While the FORCE_HOST bit remains set, the core enters the Host mode when the SESSION bit in DEVCTL is set to 1 and remains in the Host mode until the SESSION bit is cleared to 0 even if a connected device is disconnected during the session. The operating speed while in this mode is determined by the FORCE_HS and FORCE_FS bits in the TESTMODE register.

Copyright © 2013–2016, Texas Instruments Incorporated

31.2.10 Reset Considerations

The USB controller has two reset sources: hardware reset and the soft reset.

31.2.10.1 Software Reset Considerations

When the RESET bit in the control register (CTRLR) is set, all the USB controller registers and DMA operations are reset. The bit is cleared automatically.

A software reset on the CPU does not affect the register values and operation of the USB controller.

31.2.10.2 Hardware Reset Considerations

When a hardware reset is asserted, all the registers are set to their default values.

31.2.11 Interrupt Support

The USB peripheral provides the interrupts listed in Table 31-29 to the interrupt distributor module (INTD). For information on the mapping of interrupts, see your device-specific data manual.

Table 31-29. USB Interrupts

Event	Acronym	Source

31.2.12 DMA Event Support

The USB is an internal bus master peripheral and does not utilize EDMA events. The USB has its own dedicated DMA, CPPI 4.1 DMA, that it utilizes for DMA driven data transfer.

31.2.13 Power Management

The USB controller can be placed in reduced power modes to conserve power during periods of low activity. The power management of the peripheral is controlled by the processor Power and Sleep Controller (PSC). The PSC acts as a master controller for power management for all of the peripherals on the device. For detailed information on power management procedures using the PSC, see the *Power and Sleep Controller (PSC)* chapter.

Copyright © 2013–2016, Texas Instruments Incorporated

31.3 Use Cases

The USB supports the following use cases.

31.3.1 User Case 1: Example of How to Initialize the USB Controller

Example 31-1. Initializing the USB2.0 Controller

```
void usb_init()
{
       Uint16 I;

// ****************************************************************************
// Configure DRVVBUS Pin to be used for USB
// ****************************************************************************
// MAKE SURE WRITE ACCESS KEY IS INITIALIZED PRIOR TO ACCESSING ANY OF THE
// BOOTCFG REGISTERS.


   BootCfg->KICK0R = KICK0KEY;      // Write Access Key 0
   BootCfg->KICK1R = KICK1KEY;      // Write Access Key 1
   /* CONFIGURE THE DRVVBUS PIN HERE.*/
   /* See the System Configuration (SYSCFG) Module chapter for more information on how to set up
the pinmux. */


// Reset the USB controller:
        usbRegs->CTRLR |= 0x00000001;


//Wait until controller is finished with Reset. When done, it will clear the RESET bit field.
        while ((usbRegs->CTRLR & 0x1) == 1);


        // RESET: Hold PHY in Reset
        BootCfg->CFGCHIP2 |= 0x00008000;   // Hold PHY in Reset
        // Drive Reset for few clock cycles
        for (I=0; i<50; I++);
        // RESET: Release PHY from Reset
        BootCfg->CFGCHIP2 &= 0xFFFF7FFF;   // Release PHY from Reset


/* Configure PHY with the Desired Operation */
              // OTGMODE
              BootCfg->CFGCHIP2 &= 0xFFFF9FFF;   // 00=> Do Not Override PHY Values

              // PHYPWDN
              BootCfg->CFGCHIP2 &= 0xFFFFFBFF;   // 1/0 => PowerdDown/ NormalOperation
              // OTGPWRDN
              BootCfg->CFGCHIP2 &= 0xFFFFFDFF;   // 1/0 => PowerDown/ NormalOperation
              // DATAPOL
              BootCfg->CFGCHIP2 |= 0x00000100;   // 1/0 => Normal/ Reversed
              // SESNDEN
              BootCfg->CFGCHIP2 |= 0x00000020;   // 1/0 => NormalOperation/ SessionEnd
              // VBDTCTEN
              BootCfg->CFGCHIP2 |= 0x00000010;   // 1/0 => VBUS Comparator Enable/ Disable



/* Configure PHY PLL use and Select Source */
        // REF_FREQ[3:0]
        BootCfg->CFGCHIP2 |= 0x00000002;        // 0010b => 24MHz Input Source

        // USB2PHYCLKMUX: Select External Source
        BootCfg->CFGCHIP2 &= 0xFFFFF7FF;        // 1/0 => Internal/External(Pin)
```


Copyright © 2013–2016, Texas Instruments Incorporated

Example 31-1. Initializing the USB2.0 Controller (continued)

```
        // PHY_PLLON: On Simulation PHY PLL is OFF
        BootCfg->CFGCHIP2 |= 0x00000040;        // 1/0 => On/ Off



/* Wait Until PHY Clock is Good */
        while ((BootCfg->CFGCHIP2 & 0x00020000) == 0); // Wait Until PHY Clock is Good.

        #ifndef HS_ENABLE
                // Disable high-speed
                CSL_FINS(usbRegs->POWER,USB_OTG_POWER_HSEN,0);
        #else
                // Enable high-speed
                CSL_FINS(usbRegs->POWER,USB_OTG_POWER_HSEN,1);
        #endif


// Enable Interrupts
        // Enable interrupts in OTG block
        usbRegs->CTRLR &= 0xFFFFFFF7; // Enable PDR2.0 Interrupt

        usbRegs-
>INTRTXE = 0x1F;              // Enable All Core Tx Endpoints Interrupts + EP0 Tx/Rx interrupt
        usbRegs->INTRRXE = 0x1E;              // Enable All Core Rx Endpoints Interrupts

        // Enable all interrupts in OTG block
        usbRegs->INTMSKSETR = 0x01FF1E1F;

        // Enable all USB interrupts in MUSBMHDRC
        usbRegs->INTRUSBE = 0xFF;

        // Enable SUSPENDM so that suspend can be seen UTMI signal
        CSL_FINS(usbRegs->POWER,USB_OTG_POWER_ENSUSPM,1);

        //Clear all pending interrupts
        usbRegs->INTCLRR = usbRegs->INTSRCR;

        #if (_USB_PERIPHERAL_)   // defined within project file if need to function as Peripheral.
                // Set softconn bit
                CSL_FINS(usbRegs->POWER,USB_OTG_POWER_SOFTCONN,1);
                while ((usbRegs-
>DEVCTL & 0x01) == 0);  //Stay here until controller goes in Session.
        #else
                // Start a session
                CSL_FINS(usbRegs->DEVCTL,USB_OTG_DEVCTL_SESSION,1);
        #endif

}
```


Copyright © 2013–2016, Texas Instruments Incorporated

Example 31-2. Initializing the CPPI 4.1 DMA Controller

```
void cppiDmaInit() {

       switch (DMAmode) {
              case RNDIS:
                     usbRegs->CTRLR |=0x00000010; // Enable RNDIS from Global Level
                     usbRegs->RNDISR=0x11111111;
                     break;

              case GENERIC_RNDIS:
                     usbRegs->CTRLR &= ~0x00000010; // Disable RNDIS from Global Level
                     usbRegs->RNDISR=0x33333333;
                     usbRegs->GENRNDISSZ[chan_num].SIZE=descPacketLength;
                     break;

              case LINUX_CDC:
                     usbRegs->CTRLR &= ~0x00000010; // Disable RNDIS from Global Level
                     usbRegs->RNDISR=0x22222222;
                     break;

              case TRANSPARENT:
                     usbRegs->CTRLR &= ~0x00000010; // Disable RNDIS from Global Level
                     usbRegs->RNDISR=0x00000000;
                     break;

              default:
                     usbRegs->CTRLR |=0x00000010; // Enable RNDIS from Global Level
                     break;
       }
       #ifndef _USB_PERIPHERAL_            // If Controller is assuming Host Role
              #ifdef TRANSPARENT
                     usbRegs->AUTOREQ=00;      // No Auto Req
              #else
                     usbRegs->AUTOREQ=01;      // Auto Req on all but EOP
             #endif
       #endif

       A Single Queue Manager (00b) exist and 16 Regions; no particular assignment exists.

       Program Link Ram0 and Link Ram1, Base & Size.
       No Link Ram1 Size Register exists. Most likely is using the same Size Register used
       //    Link Ram1 Base
       usbRegs->QMGR.LRAM0BASE = (Uint32)queueMgrLinkRam0;
       usbRegs->QMGR.LRAM0SIZE = LINKRAM0SIZE/4;
       usbRegs->QMGR.LRAM1BASE = (Uint32)queueMgrLinkRam1;

// Allocate Resource to Region 0 (can use any of the 16 available) (memory location for Host
Packet Descriptors)
// DESC_SIZE value should be [1-8]. Values of 9 - 15 are reserved.
// Since a minimum of 32 Bytes is required, only program values above 32.
// Host Packet Descriptor sizes: Min/Max = 32/(104 + Opt S/W Data)
// REG_SIZE is the total # of Descriptors the Region can accommodate. At the minimum should be
capable of handling 32 Descriptors.

// This example is allocating specific regions for each port
       usbRegs->QMEMREGION[chan_num].QMEMRBASE=((Uint32)region0DescriptorSpace);
       usbRegs->QMEMREGION[chan_num].QMEMRCTRL=(REG0START_INDEX<<16) | (DESC_SIZE<<8) | REG_SIZE;

Configure the Scheduler
// Configure the Tx/Rx Word[x=0,1] and Scheduler Configuration Register
// Priorities are handled by programming more of the channel number wanting to see serviced
// 64 Words exist for a total of 256 entries.

       // Credit can be given to both Tx and Rx Channel within the same Register.
       // Here Rx Credit is given first for the single Rx Channel defined by chan_num.
```


Copyright © 2013–2016, Texas Instruments Incorporated

Example 31-2. Initializing the CPPI 4.1 DMA Controller (continued)

```
        // Here we are using the first 8 credits and is assigned to the Channel/Endpoint
        //  being serviced.
        usbRegs-
>DMA_SCHED.ENTRY[0]= (0x80808080 | (chan_num | (chan_num << 8) | (chan_num << 16) | (chan_num <<
24)));
        // Corresponds to WORD0 Offset 0x2800

        usbRegs-
>DMA_SCHED.ENTRY[1]=(0x00000000 | (chan_num | (chan_num << 8) | (chan_num << 16) | (chan_num <<
24)));
        // Corresponds to WORD1 Offset 0x2804, etc

        // Scheduler is Enabled and number of Credits entered (since 8 of the 256 credits are used
program 8-1=7 and enable Scheduler)
        usbRegs->DMA_SCHED.DMA_SCHED_CTRL=0x80000007; // Scheduler Control Register Offset 0x2000

// Configure Tx and Rx DMA State Registers
// The Rx Channel Global Configuration Registers are used to initialize the global
// (non descriptor type specific) behavior of each of the Rx DMA channels.  If the
// enable bit is being set, the Rx/Tx Channel Global Configuration Register SHOULD ONLY
// BE WRITTEN AFTER ALL OF THE OTHER Rx/Tx CONFIGURATION REGISTERS HAVE BEEN INITIALIZED.
// Only a Single Queue Manager exists & its value is 0.


// RxHPCRA/B requires for Queue Manager (have only one and is 00b) and Receive Submit Queues for
the
// first 4 Host Buffer Descriptors to be programmed. Since Queues 0 to 15 are alloted for Receive
// operations and there exist no dedicated queue assignments for each channel, a user can
// associate any Queue with any Channel and this association is not fixed for the Receive
Operations.
// Queue[15:0]<==>Any Rx Channel

// However, for Tx Operations, Dedicated Submit Queues have been assigned for Each Channel,
Endpoints.
// Queue[17:16]<==>TxCh[0], Queue[19:18]<==>TxCh[1], Queue[21:20]<==>TxCh[2],
Queue[23:22]<==>TxCh[3]

// CDMA Rx Chanel x Host Packet Configuration Registers A & B
// For a Single Descriptor setup, all you will be needing is RXHPCRA[13,12 & 11:00]

// Assumed that all Descriptors are going to be using the same Queue, but this is configurable.

        usbRegs-
>DMA_CTRL[chan_num].RXHPCRA=(rxSubmitQ | (rxSubmitQ<<16)); // Rx Channel 0 Host Pkt Cfg Reg A
Offset 0x180C
        usbRegs-
>DMA_CTRL[chan_num].RXHPCRB=(rxSubmitQ | (rxSubmitQ<<16)); // Rx Channel 0 Host Pkt Cfg Reg B
Offset 0x1810

        // For Loopback purposes, the same data buffer will be used for receiving and transmitting.
        //  However, different descriptors (tx) and queues will be used to process the same data
from the same buffer
        // Rx Submit Queues are assigned by H/W but they are not port specific. Queues 0-
15 are available for any channel.
        usbRegs->DMA_CTRL[chan_num].RXGCR=(0x81004000 | rxCompQ);
        // Rx Channel 0 Host Pkt Cfg Register Offset 0x1808 (Use Queue 26 for Completion/Return
Queue)

        // Tx Submit Queues are assigned by H/W and are Channel Specific. 2 Submit Queues for each
port. Queues 16, 17 for port 0, etc
        usbRegs-
>DMA_CTRL[chan_num].TXGCR=(0x80000000 | txCompQ); // Tx Channel 0 Host Pkt Cfg Register Offset
0x1800
}
```


Copyright © 2013–2016, Texas Instruments Incorporated

31.3.2 User Case 2: Example of How to Program the USB Endpoints in Peripheral Mode

Example 31-3. Programming the USB Endpoints in Peripheral Mode

```
// DMA channel number.  Valid values are 0, 1, 2, or 3.
int CHAN_NUM = 0;

// Fifo sizes: uncomment the desired size.
// This example uses 64-byte fifo.
// int fifosize = 0;    // 8 bytes
// int fifosize = 1;    // 16 bytes
// int fifosize = 2;    // 32 bytes
   int fifosize = 3;    // 64 bytes
// int fifosize = 4;    // 128 bytes
// int fifosize = 5;    // 256 bytes
// int fifosize = 6;    // 512 bytes
// int fifosize = 7;    // 1024 bytes
// int fifosize = 8;    // 2048 bytes
// int fifosize = 9;    // 4096 bytes

// FIFO address.  Leave 64-bytes for endpoint 0.
int fifo_start_address = 8;

// Uncomment the desired buffering.  If double-buffer is selected, actual
// FIFO space will be twice the value listed above for fifosize.
// This example uses single buffer.
   int double_buffer = 0;  // Single-buffer
// int double_buffer = 1;  // Double-buffer

// For maximum packet size this formula will usually work, but it can also be
// set to another value if needed.  If non power of 2 value is needed (such as
// 1023) set it explicitly.
#define FIFO_MAXP 8*(1<<fifosize);

// Set the following variable to the device address.
int device_address = 0;


// The following code should be run after receiving a USB reset from the host.


// Initialize the endpoint FIFO.  RX and TX will be allocated the same sizes.
usbRegs->INDEX = CHAN_NUM+1;
usbRegs->RXFIFOSZ = fifosize | ((double_buffer & 1)<<4);
usbRegs->RXFIFOADDR = fifo_start_address;
usbRegs->TXFIFOSZ =    fifosize | ((double_buffer & 1)<<4);
usbRegs->TXFIFOADDR = fifo_start_address + (1<<(fifosize+double_buffer));
usbRegs->RXMAXP = FIFO_MAXP;
usbRegs->TXMAXP = FIFO_MAXP;

// Force Data Toggle is optional for interrupt traffic.  Uncomment if needed.
// CSL_FINS(usbRegs->PERI_TXCSR,USB_PERI_TXCSR_FRCDATATOG,1);

// Uncomment below to configure the endpoint for ISO and not respond with a
// handshake packet.
// CSL_FINS(usbRegs->PERI_RXCSR,USB_PERI_RXCSR_ISO,1);
// CSL_FINS(usbRegs->PERI_TXCSR,USB_PERI_TXCSR_ISO,1);

// After receiving a successful set-address command, set the following register
// to the specified address immediately following the status stage.
usbRegs->FADDR = device_address;
```


Copyright © 2013–2016, Texas Instruments Incorporated

31.3.3 User Case 3: Example of How to Program the USB Endpoints in Host Mode

Example 31-4. Programming the USB Endpoints in Host Mode

```
// DMA channel number.  Valid values are 0, 1, 2, or 3.
int CHAN_NUM = 0;

// Fifo sizes: uncomment the desired size.
// This example uses 64-byte fifo.
// int fifosize = 0;    // 8 bytes
// int fifosize = 1;    // 16 bytes
// int fifosize = 2;    // 32 bytes
   int fifosize = 3;    // 64 bytes
// int fifosize = 4;    // 128 bytes
// int fifosize = 5;    // 256 bytes
// int fifosize = 6;    // 512 bytes
// int fifosize = 7;    // 1024 bytes
// int fifosize = 8;    // 2048 bytes
// int fifosize = 9;    // 4096 bytes

// FIFO address.  Leave 64-bytes for endpoint 0.
int fifo_start_address = 8;

// Uncomment the desired buffering.  If double-buffer is selected, actual
// FIFO space will be twice the value listed above for fifosize.
// This example uses single buffer.
   int double_buffer = 0;  // Single-buffer
// int double_buffer = 1;  // Double-buffer

// Set the following variable to the device endpoint type: CONTROL ISO BULK or IN
  int device_protocol = BULK;
//int device_protocol = ISO;
//int device_protocol = INT;

// USB speeds
#define LOW_SPEED 0
#define FULL_SPEED 1
#define HIGH_SPEED 2

// TXTYPE protocol
#define CONTROL 0
#define ISO 1
#define BULK 2
#define INT 3

// For maximum packet size this formula will usually work, but it can also be
// set to another value if needed.  If non power of 2 value is needed (such as
// 1023) set it explicitly.
#define FIFO_MAXP 8*(1<<fifosize);

// Set the following variable to the device address.
int device_address = 1;

// Set the following variable to the device endpoint number.
int device_ep = 1;

// Variable used for endpoint configuration
Uint8 type = 0;

// Variable to keep track of errors
int error = 0;


// The following code should be run after resetting the attached device
```


Copyright © 2013–2016, Texas Instruments Incorporated

Example 31-4. Programming the USB Endpoints in Host Mode (continued)

```
// Initialize the endpoint FIFO.  RX and TX will be allocated the same sizes.
usbRegs->INDEX = CHAN_NUM+1;
usbRegs->RXFIFOSZ = fifosize | ((double_buffer & 1)<<4);
usbRegs->RXFIFOADDR = fifo_start_address;
usbRegs->TXFIFOSZ =    fifosize | ((double_buffer & 1)<<4);
usbRegs->TXFIFOADDR = fifo_start_address + (1<<(fifosize+double_buffer));
usbRegs->RXMAXP = FIFO_MAXP;
usbRegs->TXMAXP = FIFO_MAXP;


//Configure the endpoint
switch (device_speed) {
    case LOW_SPEED : type = (3<<6) | ((device_protocol & 3) << 4) | (device_ep & 0xf); break;
    case FULL_SPEED: type = (2<<6) | ((device_protocol & 3) << 4) | (device_ep & 0xf); break;
    case HIGH_SPEED: type = (1<<6) | ((device_protocol & 3) << 4) | (device_ep & 0xf); break;
    default:error++;
}
usbRegs->EPCSR[CHAN_NUM+1].HOST_TYPE0 = type;  // TXTYPE
usbRegs->EPCSR[CHAN_NUM+1].HOST_RXTYPE = type;

// Set NAK limit / Polling interval (Interrupt & Iso protocols)
if ((device_protocol == INT) || (device_protocol == ISO)) {
    usbRegs->EPCSR[CHAN_NUM+1].HOST_NAKLIMIT0 = TXINTERVAL;    // TX Polling interval
    usbRegs->EPCSR[CHAN_NUM+1].HOST_RXINTERVAL = RXINTERVAL;    // RX Polling interval
} else {
    usbRegs->EPCSR[CHAN_NUM+1].HOST_NAKLIMIT0 = 2;    // Frames to timeout from NAKs
    usbRegs->EPCSR[CHAN_NUM+1].HOST_RXINTERVAL = 2;    // Frames to timeout from NAKs
}

//Set the address for transactions after SET ADDRESS successfully completed
usbRegs->EPTRG[CHAN_NUM+1].TXFUNCADDR = device_address;
usbRegs->EPTRG[CHAN_NUM+1].RXFUNCADDR = device_address;
```


Copyright © 2013–2016, Texas Instruments Incorporated

31.3.4 User Case 4: Example of How to Program the USB DMA Controller

Example 31-5. Programming the USB DMA Controller

```
typedef struct {
        Uint32 PktLength:22;
        Uint32 ProtSize:5;
        Uint32 HostPktType:5; // This should be 16
}HPDWord0;

typedef struct {
        Uint32 DstTag:16;        //bits[15:0] always Zero
        Uint32 SrcSubChNum:5;    //bits[20:16] always Zero
        Uint32 SrcChNum:6;       //bits[26:21]
        Uint32 SrcPrtNum:5;      //bits[31:27]
}HPDWord1;

typedef struct {
        Uint32 PktRetQueue:12;   //bits[11:0]
        Uint32 PktRetQM:2;       //bits[13:12]
        Uint32 OnChip:1;         //bit[14]
        Uint32 RetPolicy:1;      //bit[15]
        Uint32 ProtoSpecific:4;  //bits[19:16]
        Uint32 Rsv:6;            //bits[25:20]
        Uint32 PktType:5;        //bits[30:26]
        Uint32 PktErr:1;         //bit[31]
}HPDWord2;

typedef struct hostPacketDesc {
        HPDWord0 HPDword0;
        HPDWord1 HPDword1;
        HPDWord2 HPDword2;
        Uint32 HPDword3buffLength;
        Uint32 HPDword4buffAdd;
        Uint32 HPDword5nextHBDptr;
        Uint32 HPDword6orgBuffLength;
        Uint32 HPDword7orgBuffAdd;
} HostPacketDesc;


// The following sample code uses region 0 for all of the ports. Ports/Channels/EPs are not
limited to a single region


void initSingleHPDorHBD(Uint16 descNum, dataDir dir, descType desc, Uint16 returnQueue) {

/*

*****************************************************************************************

 Initialize a Single Transmit and Receive Host Packet or Buffer Descriptor

*/

            if ((desc==PACKET_DESC) & (dir==TRANSMIT))
                        region0DescriptorSpace[descNum].HPDword0.HostPktType=16;   //This value
is always fixed. For Packet Type Decriptors=16
            else
                        region0DescriptorSpace[descNum].HPDword0.HostPktType=0;     //Word0,
Word1, and Half of Word2 are Reserved for Buffer Descriptors.


            region0DescriptorSpace[descNum].HPDword0.ProtSize=0;


            if ((dir==TRANSMIT) & (desc==PACKET_DESC)) {
```


Copyright © 2013–2016, Texas Instruments Incorporated

Example 31-5. Programming the USB DMA Controller (continued)

```
                        if (DMAmode==TRANSPARENT)

region0DescriptorSpace[descNum].HPDword0.PktLength=singlePktLength;           //Packet Length
(For Transparent Mode this is <= Tx/RxMaxP Value. For RNDIS or like it is = Tx/RxMaxP Value).
                        else

region0DescriptorSpace[descNum].HPDword0.PktLength=descPacketLength;       //Actual Packet
Length: This is the size of the Packet noted at descriptor level to be Transmitted.



                                                              // This is different from the USB Max
Packet Size. This is the total data length.

            }



            else      // This and other Packet related info will be updated by the PORT for
Receive and can be any value.
                        region0DescriptorSpace[descNum].HPDword0.PktLength=0;
// This is actual Packet Length. It will be populated by the Rx Port of the CPPI DMA


            region0DescriptorSpace[descNum].HPDword1.DstTag=0;             //Always programmed
to ZERO.
            region0DescriptorSpace[descNum].HPDword1.SrcSubChNum=0;  //Always programmed to ZERO.
            region0DescriptorSpace[descNum].HPDword1.SrcChNum=0;                    //Always
programmed to ZERO.


            if(desc==BUFFER_DESC)
                        region0DescriptorSpace[descNum].HPDword1.SrcPrtNum=0;        //Word1 is
Reserved for Buffer DESC.
            else
                        region0DescriptorSpace[descNum].HPDword1.SrcPrtNum=chan_num+1;
//Ports[1,2,3,4] is associated with Endpoints[1,2,3,4] respectively.



            region0DescriptorSpace[descNum].HPDword2.PktRetQueue=returnQueue; //24 and 25 for Tx -
 26 and 27 for Rx Completion
            region0DescriptorSpace[descNum].HPDword2.PktRetQM=0;
            region0DescriptorSpace[descNum].HPDword2.OnChip=1;             // Descriptor is
located On-Chip
            region0DescriptorSpace[descNum].HPDword2.RetPolicy=0;
            region0DescriptorSpace[descNum].HPDword2.ProtoSpecific=0;
            region0DescriptorSpace[descNum].HPDword2.Rsv=0;


            if(desc==BUFFER_DESC)
                        region0DescriptorSpace[descNum].HPDword2.PktType=0;
 // Half of Word 3 is Reserved for Buffer DESC.
            else
                        region0DescriptorSpace[descNum].HPDword2.PktType=5;
 // USB Packet ID is 5


            region0DescriptorSpace[descNum].HPDword2.PktErr=0;


            if(DESCsetup==SINGLE_DESC_SETUP)
                        region0DescriptorSpace[descNum].HPDword3buffLength=descPacketLength;
            else
```


Copyright © 2013–2016, Texas Instruments Incorporated

Example 31-5. Programming the USB DMA Controller (continued)

```
                        region0DescriptorSpace[descNum].HPDword3buffLength=singlePktLength;


            if ((dir==TRANSMIT) & (prevDescTxNum==0)) {
                        region0DescriptorSpace[descNum].HPDword4buffAdd=(Uint32)rxBuffer;
                        prevDescTxNum++;
            } else if ((dir==TRANSMIT) & (prevDescTxNum!=0)) {

region0DescriptorSpace[descNum].HPDword4buffAdd=region0DescriptorSpace[descNum-
1].HPDword4buffAdd + region0DescriptorSpace[descNum-1].HPDword3buffLength;
                        prevDescTxNum++;
            }



            if ((dir==RECEIVE) & (prevDescRxNum==0)) {
                        region0DescriptorSpace[descNum].HPDword4buffAdd=(Uint32)rxBuffer;
                        prevDescRxNum++;
            } else if ((dir==RECEIVE) & (prevDescRxNum!=0)) {

region0DescriptorSpace[descNum].HPDword4buffAdd=region0DescriptorSpace[descNum-
1].HPDword4buffAdd + region0DescriptorSpace[descNum-1].HPDword3buffLength;
                        prevDescRxNum++;
            }


            if ((dir==TRANSMIT) & ((prevDescTxNum-1)>0))
                        region0DescriptorSpace[descNum-
1].HPDword5nextHBDptr=(Uint32)&region0DescriptorSpace[descNum];            //Modify Previous Link
Address


            region0DescriptorSpace[descNum].HPDword5nextHBDptr=0;        //Current Descriptor is
the Last Descriptor: Null Value is used as the Next Buffer Descriptor Address
region0DescriptorSpace[descNum].HPDword6orgBuffLength=region0DescriptorSpace[descNum].HPDword3buff
Length;

region0DescriptorSpace[descNum].HPDword7orgBuffAdd=region0DescriptorSpace[descNum].HPDword4buffAdd
;;
}




// usage: qDesc2SubmitQ(rx/txSubmitQ,HPD/HBD);  //Queue Number, HPDdescriptorNumber
void qDesc2SubmitQ(Uint16 queueNum, Uint16 hpdDescriptorNum) {
        usbRegs-
>QCTRL[queueNum].CTRLD=(Uint32)&region0DescriptorSpace[hpdDescriptorNum] | 0x2; //
bits[4:0]=dec_size=[0-31]=[24,28,32,...,148]
}

void enableCoreTxDMA(Uint16 endPoint) {
        Uint16 index_save;
        index_save=usbRegs->INDEX;
        usbRegs->INDEX=endPoint;
        usbRegs->TXCSR.PERI_TXCSR&=0x7FFF; // Clear AUTOSET
        usbRegs->TXCSR.PERI_TXCSR|=0x1400; // Set DMAReqEnab & DMAReqMode
        usbRegs->INDEX=index_save;
}

void enableCoreRxDMA(Uint16 endPoint) {
        Uint16 index_save;
        index_save=usbRegs->INDEX;
        usbRegs->INDEX=endPoint;
```


Copyright © 2013–2016, Texas Instruments Incorporated

Example 31-5. Programming the USB DMA Controller (continued)

```
        usbRegs->RXCSR.PERI_RXCSR&=0x77FF;  // Clear AUTOCLEAR and DMAReqMode
        usbRegs->RXCSR.PERI_RXCSR|=0x2000;  // Set DMAReqEnab
        usbRegs->INDEX=index_save;
}

Uint32 readCompletionQueue(Uint16 queueNum) {
        Uint32 DescAddress;
        DescAddress=(Uint32)usbRegs->QCTRL[queueNum].CTRLD;
        DescAddress&=0xFFFFFFE0;
        return(DescAddress);
}

void disableCoreRxDMA(Uint16 endPoint) {
        Uint16 index_save;
        index_save=usbRegs->INDEX;
        usbRegs->INDEX=endPoint;
        usbRegs->RXCSR.PERI_RXCSR &= 0xDFFF;  // Clear DMAReqEnab
        usbRegs->INDEX=index_save;
}

void disableCoreTxDMA(Uint16 endPoint) {
        Uint16 index_save;
        index_save=usbRegs->INDEX;
        usbRegs->INDEX=endPoint;
        usbRegs->TXCSR.PERI_TXCSR&=0x7FFF;  // Clear AUTOSET
        usbRegs->TXCSR.PERI_TXCSR&=0xEBFF;  // Clear DMAReqEnab & DMAReqMode
        usbRegs->INDEX=index_save;
}

// sample peripheral code
Uint16 usbMain(void) {
        usb_init();
        usb_device_init();  // initialize usb core related vars: index, fifo, etc and DMA related
vars, data buff, rx/txSubmitQ, etc
        wait_for_reset();   // wait here until host performs a reset.
        cppiDmaInit();  // Init CPPI 4.1 DMA
// ****************** Initialize Receive Buffer Descriptors ***********************************
// Host is performing a transfer and the device is going to loop it back out.
// The example below (non commented part of the code) applies for a data transfer made of two 64
bytes packet.
// These two packets are treated as part of a single transfer for Non-
Transparent DMA modes and as two transfers for Transparent
//   DMA mode. This needs to be understood for the below to make sense, especially the last
Descriptor shown below.
// Initialize receive descriptors (HBDs)
        initSingleHPDorHBD(0,RECEIVE,BUFFER_DESC,rxCompQ);  //Usage:
initSingleHPDorHBD(descNum,dir,TypeOfDesc,returnQueue)
        if (DESCsetup==MULTIPLE_DESC_SETUP) {
              initSingleHPDorHBD(1,RECEIVE,BUFFER_DESC,rxCompQ);  //Usage:
initSingleHPDorHBD(descNum,dir,TypeOfDesc,returnQueue)
        //    initSingleHPDorHBD(2,RECEIVE,BUFFER_DESC,rxCompQ);  //Usage:
initSingleHPDorHBD(descNum,dir,TypeOfDesc,returnQueue)
        //    initSingleHPDorHBD(3,RECEIVE,BUFFER_DESC,rxCompQ);  //Usage:
initSingleHPDorHBD(descNum,dir,TypeOfDesc,returnQueue)

        //       :
        //       :
        }

// Last Receive Descriptor creation (this is the case where two or three Buffer Descriptors are
needed).
// Note again that this example is for a two packet data transfer.
        if (DMAmode!=TRANSPARENT) {   // Need to Create the Descriptor to be used for the Null
Packet.
```


Copyright © 2013–2016, Texas Instruments Incorporated

Example 31-5. Programming the USB DMA Controller (continued)

```
            if (DESCsetup==SINGLE_DESC_SETUP)     // One additonal Rx Desc Needs to be Queued for
Handling Null Packet
                initSingleHPDorHBD(1,RECEIVE,BUFFER_DESC,rxCompQ);         //Usage:
initSingleHPDorHBD(descNum,dir,TypeOfDesc,returnQueue)
                        else
                                    initSingleHPDorHBD(2,RECEIVE,BUFFER_DESC,rxCompQ);
//Usage: initSingleHPDorHBD(descNum,dir,TypeOfDesc,returnQueue)
            }

// ****************** Initialize Transmit Packet and Buffer Descriptors************************
// See comment above to understand the reasons for the total number of descriptors.
// Initialize transmit descriptors (HPD and HBDs)
        //Initialize HPD Descriptor 16 for Transmit.
        initSingleHPDorHBD(16,TRANSMIT,PACKET_DESC,txCompQ);   //Usage:
initSingleHPDorHBD(descNum,dir,TypeOfDesc,returnQueue)
        if (DESCsetup==MULTIPLE_DESC_SETUP) {
                if(DMAmode==TRANSPARENT) {
                        initSingleHPDorHBD(17,TRANSMIT,PACKET_DESC,txCompQ);   //Usage:
initSingleHPDorHBD(descNum,dir,TypeOfDesc,returnQueue)
//                      initSingleHPDorHBD(18,TRANSMIT,PACKET_DESC,txCompQ);   //Usage:
initSingleHPDorHBD(descNum,dir,TypeOfDesc,returnQueue)
//                      initSingleHPDorHBD(19,TRANSMIT,PACKET_DESC,txCompQ);   //Usage:
initSingleHPDorHBD(descNum,dir,TypeOfDesc,returnQueue)
                }
                else {
                        initSingleHPDorHBD(17,TRANSMIT,BUFFER_DESC,txCompQ);    //Usage:
initSingleHPDorHBD(descNum,dir,TypeOfDesc,returnQueue)
//                      initSingleHPDorHBD(18,TRANSMIT,BUFFER_DESC,txCompQ);    //Usage:
initSingleHPDorHBD(descNum,dir,TypeOfDesc,returnQueue)
//                      initSingleHPDorHBD(19,TRANSMIT,BUFFER_DESC,txCompQ);    //Usage:
initSingleHPDorHBD(descNum,dir,TypeOfDesc,returnQueue)
                }
            }
// ****************** Submit Receive Buffer Descriptors ***************************************
        //Submit Receive Descriptors
        qDesc2SubmitQ(rxSubmitQ,0);            //Queue Number, HPDdescriptorNumber
        if (DESCsetup==MULTIPLE_DESC_SETUP)
                qDesc2SubmitQ(rxSubmitQ,1);            //Queue Number, HPDdescriptorNumber

        if (DMAmode!=TRANSPARENT) {   // Need to Submit the Descriptor to be used for the Null
Packet.
                if (DESCsetup==SINGLE_DESC_SETUP)     // One additonal Rx Desc Needs to be Queued
for Handling Null Packet
                        qDesc2SubmitQ(rxSubmitQ,1);            //Queue Number, HPDdescriptorNumber
                else
                        qDesc2SubmitQ(rxSubmitQ,2);            //Queue Number, HPDdescriptorNumber
        }

        //Enable Rx DMA
        enableCoreRxDMA(endpoint);
// ****************** Submit Transmit Buffer Descriptors ************************************

// wait till all data is received here <<<<<<<


        //Submit Transmit Descriptors
        qDesc2SubmitQ(txSubmitQ,16);  //Queue Number, HPDdescriptorNumber
        if (DESCsetup==MULTIPLE_DESC_SETUP)
                qDesc2SubmitQ(txSubmitQ,17);  //Queue Number, HPDdescriptorNumber
//      qDesc2SubmitQ(txSubmitQ,18);  //Queue Number, HPDdescriptorNumber
        enableCoreTxDMA(endpoint);
// ***************************************************************************************

// wait till all data is received here <<<<<<<
```


Copyright © 2013–2016, Texas Instruments Incorporated

31.4 Registers

Table 31-30 lists the memory-mapped registers for the universal serial bus OTG controller (USB0). See your device-specific data manual for the memory address of these registers. The base address is 01E0 0000h.

NOTE: In some cases, a single register address can have different names or meanings depending on the mode (host/peripheral) or the setting of the index register. The meaning of some bit fields varies with the mode.

Table 31-30. Universal Serial Bus OTG (USB0) Registers

VBUS Slave Address Offset	Acronym	Register Description	Section
0h	REVID	Revision Identification Register	Section 31.4.1
4h	CTRLR	Control Register	Section 31.4.2
8h	STATR	Status Register	Section 31.4.3
Ch	EMUR	Emulation Register	Section 31.4.4
10h	MODE	Mode Register	Section 31.4.5
14h	AUTOREQ	Autorequest Register	Section 31.4.6
18h	SRPFIXTIME	SRP Fix Time Register	Section 31.4.7
1Ch	TEARDOWN	Teardown Register	Section 31.4.8
20h	INTSRCR	USB Interrupt Source Register	Section 31.4.9
24h	INTSETR	USB Interrupt Source Set Register	Section 31.4.10
28h	INTCLRR	USB Interrupt Source Clear Register	Section 31.4.11
2Ch	INTMSKR	USB Interrupt Mask Register	Section 31.4.12
30h	INTMSKSETR	USB Interrupt Mask Set Register	Section 31.4.13
34h	INTMSKCLRR	USB Interrupt Mask Clear Register	Section 31.4.14
38h	INTMASKEDR	USB Interrupt Source Masked Register	Section 31.4.15
3Ch	EOIR	USB End of Interrupt Register	Section 31.4.16
50h	GENRNDISSZ1	Generic RNDIS Size EP1	Section 31.4.17
54h	GENRNDISSZ2	Generic RNDIS Size EP2	Section 31.4.18
58h	GENRNDISSZ3	Generic RNDIS Size EP3	Section 31.4.19
5Ch	GENRNDISSZ4	Generic RNDIS Size EP4	Section 31.4.20
		Common USB Registers	
400h	FADDR	Function Address Register	Section 31.4.21
401h	POWER	Power Management Register	Section 31.4.22
402h	INTRTX	Interrupt Register for Endpoint 0 plus Transmit Endpoints 1 to 4	Section 31.4.23
404h	INTRRX	Interrupt Register for Receive Endpoints 1 to 4	Section 31.4.24
406h	INTRTXE	Interrupt Enable Register for INTRTX	Section 31.4.25
408h	INTRRXE	Interrupt Enable Register for INTRRX	Section 31.4.26
40Ah	INTRUSB	Interrupt Register for Common USB Interrupts	Section 31.4.27
40Bh	INTRUSBE	Interrupt Enable Register for INTRUSB	Section 31.4.28
40Ch	FRAME	Frame Number Register	Section 31.4.29
40Eh	INDEX	Index Register for Selecting the Endpoint Status and Control Registers	Section 31.4.30
40Fh	TESTMODE	Register to Enable the USB 2.0 Test Modes	Section 31.4.31

Copyright © 2013–2016, Texas Instruments Incorporated

Table 31-30. Universal Serial Bus OTG (USB0) Registers (continued)

VBUS Slave Address Offset	Acronym	Register Description	Section
		Indexed Registers	
		These registers operate on the endpoint selected by the INDEX register	
410h	TXMAXP	Maximum Packet Size for Peripheral/Host Transmit Endpoint (Index register set to select Endpoints 1-4 only)	Section 31.4.32
412h	PERI_CSR0	Control Status Register for Endpoint 0 in Peripheral Mode. (Index register set to select Endpoint 0)	Section 31.4.33
	HOST_CSR0	Control Status Register for Endpoint 0 in Host Mode. (Index register set to select Endpoint 0)	Section 31.4.34
	PERI_TXCSR	Control Status Register for Peripheral Transmit Endpoint. (Index register set to select Endpoints 1-4)	Section 31.4.35
	HOST_TXCSR	Control Status Register for Host Transmit Endpoint. (Index register set to select Endpoints 1-4)	Section 31.4.36
414h	RXMAXP	Maximum Packet Size for Peripheral/Host Receive Endpoint (Index register set to select Endpoints 1-4 only)	Section 31.4.37
416h	PERI_RXCSR	Control Status Register for Peripheral Receive Endpoint. (Index register set to select Endpoints 1-4)	Section 31.4.38
	HOST_RXCSR	Control Status Register for Host Receive Endpoint. (Index register set to select Endpoints 1-4)	Section 31.4.39
418h	COUNT0	Number of Received Bytes in Endpoint 0 FIFO. (Index register set to select Endpoint 0)	Section 31.4.40
	RXCOUNT	Number of Bytes in Host Receive Endpoint FIFO. (Index register set to select Endpoints 1- 4)	Section 31.4.41
41Ah	HOST_TYPE0	Defines the speed of Endpoint 0	Section 31.4.42
	HOST_TXTYPE	Sets the operating speed, transaction protocol and peripheral endpoint number for the host Transmit endpoint. (Index register set to select Endpoints 1-4 only)	Section 31.4.43
41Bh	HOST_NAKLIMIT0	Sets the NAK response timeout on Endpoint 0. (Index register set to select Endpoint 0)	Section 31.4.44
	HOST_TXINTERVAL	Sets the polling interval for Interrupt/ISOC transactions or the NAK response timeout on Bulk transactions for host Transmit endpoint. (Index register set to select Endpoints 1-4 only)	Section 31.4.45
41Ch	HOST_RXTYPE	Sets the operating speed, transaction protocol and peripheral endpoint number for the host Receive endpoint. (Index register set to select Endpoints 1-4 only)	Section 31.4.46
41Dh	HOST_RXINTERVAL	Sets the polling interval for Interrupt/ISOC transactions or the NAK response timeout on Bulk transactions for host Receive endpoint. (Index register set to select Endpoints 1-4 only)	Section 31.4.47
41Fh	CONFIGDATA	Returns details of core configuration. (Index register set to select Endpoint 0)	Section 31.4.48
		FIFOn	
420h	FIFO0	Transmit and Receive FIFO Register for Endpoint 0	Section 31.4.49
424h	FIFO1	Transmit and Receive FIFO Register for Endpoint 1	Section 31.4.50
428h	FIFO2	Transmit and Receive FIFO Register for Endpoint 2	Section 31.4.51
42Ch	FIFO3	Transmit and Receive FIFO Register for Endpoint 3	Section 31.4.52
430h	FIFO4	Transmit and Receive FIFO Register for Endpoint 4	Section 31.4.53
		OTG Device Control	
460h	DEVCTL	Device Control Register	Section 31.4.54

Copyright © 2013–2016, Texas Instruments Incorporated

Table 31-30. Universal Serial Bus OTG (USB0) Registers (continued)

VBUS Slave Address Offset	Acronym	Register Description	Section
		Dynamic FIFO Control	
462h	TXFIFOSZ	Transmit Endpoint FIFO Size (Index register set to select Endpoints 1-4 only)	Section 31.4.55
463h	RXFIFOSZ	Receive Endpoint FIFO Size (Index register set to select Endpoints 1-4 only)	Section 31.4.56
464h-465h	TXFIFOADDR	Transmit Endpoint FIFO Address (Index register set to select Endpoints 1-4 only)	Section 31.4.57
466h-467h	RXFIFOADDR	Receive Endpoint FIFO Address (Index register set to select Endpoints 1-4 only)	Section 31.4.58
46Ch-46Dh	HWVERS	Hardware Version Register	Section 31.4.59
		Target Endpoint 0 Control Registers, Valid Only in Host Mode	
480h	TXFUNCADDR	Address of the target function that has to be accessed through the associated Transmit Endpoint.	Section 31.4.60
482h	TXHUBADDR	Address of the hub that has to be accessed through the associated Transmit Endpoint. This is used only when full speed or low speed device is connected via a USB2.0 high-speed hub.	Section 31.4.61
483h	TXHUBPORT	Port of the hub that has to be accessed through the associated Transmit Endpoint. This is used only when full speed or low speed device is connected via a USB2.0 high-speed hub.	Section 31.4.62
484h	RXFUNCADDR	Address of the target function that has to be accessed through the associated Receive Endpoint.	Section 31.4.63
486h	RXHUBADDR	Address of the hub that has to be accessed through the associated Receive Endpoint. This is used only when full speed or low speed device is connected via a USB2.0 high-speed hub.	Section 31.4.64
487h	RXHUBPORT	Port of the hub that has to be accessed through the associated Receive Endpoint. This is used only when full speed or low speed device is connected via a USB2.0 high-speed hub.	Section 31.4.65
		Target Endpoint 1 Control Registers, Valid Only in Host Mode	
488h	TXFUNCADDR	Address of the target function that has to be accessed through the associated Transmit Endpoint.	Section 31.4.60
48Ah	TXHUBADDR	Address of the hub that has to be accessed through the associated Transmit Endpoint. This is used only when full speed or low speed device is connected via a USB2.0 high-speed hub.	Section 31.4.61
48Bh	TXHUBPORT	Port of the hub that has to be accessed through the associated Transmit Endpoint. This is used only when full speed or low speed device is connected via a USB2.0 high-speed hub.	Section 31.4.62
48Ch	RXFUNCADDR	Address of the target function that has to be accessed through the associated Receive Endpoint.	Section 31.4.63
48Eh	RXHUBADDR	Address of the hub that has to be accessed through the associated Receive Endpoint. This is used only when full speed or low speed device is connected via a USB2.0 high-speed hub.	Section 31.4.64
48Fh	RXHUBPORT	Port of the hub that has to be accessed through the associated Receive Endpoint. This is used only when full speed or low speed device is connected via a USB2.0 high-speed hub.	Section 31.4.65

Copyright © 2013–2016, Texas Instruments Incorporated

Table 31-30. Universal Serial Bus OTG (USB0) Registers (continued)

VBUS Slave Address Offset	Acronym	Register Description	Section
		Target Endpoint 2 Control Registers, Valid Only in Host Mode	
490h	TXFUNCADDR	Address of the target function that has to be accessed through the associated Transmit Endpoint.	Section 31.4.60
492h	TXHUBADDR	Address of the hub that has to be accessed through the associated Transmit Endpoint. This is used only when full speed or low speed device is connected via a USB2.0 high-speed hub.	Section 31.4.61
493h	TXHUBPORT	Port of the hub that has to be accessed through the associated Transmit Endpoint. This is used only when full speed or low speed device is connected via a USB2.0 high-speed hub.	Section 31.4.62
494h	RXFUNCADDR	Address of the target function that has to be accessed through the associated Receive Endpoint.	Section 31.4.63
496h	RXHUBADDR	Address of the hub that has to be accessed through the associated Receive Endpoint. This is used only when full speed or low speed device is connected via a USB2.0 high-speed hub.	Section 31.4.64
497h	RXHUBPORT	Port of the hub that has to be accessed through the associated Receive Endpoint. This is used only when full speed or low speed device is connected via a USB2.0 high-speed hub.	Section 31.4.65
		Target Endpoint 3 Control Registers, Valid Only in Host Mode	
498h	TXFUNCADDR	Address of the target function that has to be accessed through the associated Transmit Endpoint.	Section 31.4.60
49Ah	TXHUBADDR	Address of the hub that has to be accessed through the associated Transmit Endpoint. This is used only when full speed or low speed device is connected via a USB2.0 high-speed hub.	Section 31.4.61
49Bh	TXHUBPORT	Port of the hub that has to be accessed through the associated Transmit Endpoint. This is used only when full speed or low speed device is connected via a USB2.0 high-speed hub.	Section 31.4.62
49Ch	RXFUNCADDR	Address of the target function that has to be accessed through the associated Receive Endpoint.	Section 31.4.63
49Eh	RXHUBADDR	Address of the hub that has to be accessed through the associated Receive Endpoint. This is used only when full speed or low speed device is connected via a USB2.0 high-speed hub.	Section 31.4.64
49Fh	RXHUBPORT	Port of the hub that has to be accessed through the associated Receive Endpoint. This is used only when full speed or low speed device is connected via a USB2.0 high-speed hub.	Section 31.4.65
		Target Endpoint 4 Control Registers, Valid Only in Host Mode	
4A0h	TXFUNCADDR	Address of the target function that has to be accessed through the associated Transmit Endpoint.	Section 31.4.60
4A2h	TXHUBADDR	Address of the hub that has to be accessed through the associated Transmit Endpoint. This is used only when full speed or low speed device is connected via a USB2.0 high-speed hub.	Section 31.4.61
4A3h	TXHUBPORT	Port of the hub that has to be accessed through the associated Transmit Endpoint. This is used only when full speed or low speed device is connected via a USB2.0 high-speed hub.	Section 31.4.62
4A4h	RXFUNCADDR	Address of the target function that has to be accessed through the associated Receive Endpoint.	Section 31.4.63

Copyright © 2013–2016, Texas Instruments Incorporated

Table 31-30. Universal Serial Bus OTG (USB0) Registers (continued)

VBUS Slave Address Offset	Acronym	Register Description	Section
4A6h	RXHUBADDR	Address of the hub that has to be accessed through the associated Receive Endpoint. This is used only when full speed or low speed device is connected via a USB2.0 high-speed hub.	Section 31.4.64
4A7h	RXHUBPORT	Port of the hub that has to be accessed through the associated Receive Endpoint. This is used only when full speed or low speed device is connected via a USB2.0 high-speed hub.	Section 31.4.65
		Control and Status Register for Endpoint 0	
502h	PERI_CSR0	Control Status Register for Endpoint 0 in Peripheral Mode	Section 31.4.33
	HOST_CSR0	Control Status Register for Endpoint 0 in Host Mode	Section 31.4.34
508h	COUNT0	Number of Received Bytes in Endpoint 0 FIFO	Section 31.4.40
50Ah	HOST_TYPE0	Defines the Speed of Endpoint 0	Section 31.4.42
50Bh	HOST_NAKLIMIT0	Sets the NAK Response Timeout on Endpoint 0	Section 31.4.44
50Fh	CONFIGDATA	Returns details of core configuration	Section 31.4.48
		Control and Status Register for Endpoint 1	
510h	TXMAXP	Maximum Packet Size for Peripheral/Host Transmit Endpoint	Section 31.4.32
512h	PERI_TXCSR	Control Status Register for Peripheral Transmit Endpoint (peripheral mode)	Section 31.4.35
	HOST_TXCSR	Control Status Register for Host Transmit Endpoint (host mode)	Section 31.4.36
514h	RXMAXP	Maximum Packet Size for Peripheral/Host Receive Endpoint	Section 31.4.37
516h	PERI_RXCSR	Control Status Register for Peripheral Receive Endpoint (peripheral mode)	Section 31.4.38
	HOST_RXCSR	Control Status Register for Host Receive Endpoint (host mode)	Section 31.4.39
518h	RXCOUNT	Number of Bytes in Host Receive endpoint FIFO	Section 31.4.41
51Ah	HOST_TXTYPE	Sets the operating speed, transaction protocol and peripheral endpoint number for the host Transmit endpoint.	Section 31.4.43
51Bh	HOST_TXINTERVAL	Sets the polling interval for Interrupt/ISOC transactions or the NAK response timeout on Bulk transactions for host Transmit endpoint.	Section 31.4.45
51Ch	HOST_RXTYPE	Sets the operating speed, transaction protocol and peripheral endpoint number for the host Receive endpoint.	Section 31.4.46
51Dh	HOST_RXINTERVAL	Sets the polling interval for Interrupt/ISOC transactions or the NAK response timeout on Bulk transactions for host Receive endpoint.	Section 31.4.47
		Control and Status Register for Endpoint 2	
520h	TXMAXP	Maximum Packet Size for Peripheral/Host Transmit Endpoint	Section 31.4.32
522h	PERI_TXCSR	Control Status Register for Peripheral Transmit Endpoint (peripheral mode)	Section 31.4.35
	HOST_TXCSR	Control Status Register for Host Transmit Endpoint (host mode)	Section 31.4.36
524h	RXMAXP	Maximum Packet Size for Peripheral/Host Receive Endpoint	Section 31.4.37
526h	PERI_RXCSR	Control Status Register for Peripheral Receive Endpoint (peripheral mode)	Section 31.4.38
	HOST_RXCSR	Control Status Register for Host Receive Endpoint (host mode)	Section 31.4.39
528h	RXCOUNT	Number of Bytes in Host Receive endpoint FIFO	Section 31.4.41
52Ah	HOST_TXTYPE	Sets the operating speed, transaction protocol and peripheral endpoint number for the host Transmit endpoint.	Section 31.4.43

Copyright © 2013–2016, Texas Instruments Incorporated

Table 31-30. Universal Serial Bus OTG (USB0) Registers (continued)

VBUS Slave Address Offset	Acronym	Register Description	Section
52Bh	HOST_TXINTERVAL	Sets the polling interval for Interrupt/ISOC transactions or the NAK response timeout on Bulk transactions for host Transmit endpoint.	Section 31.4.45
52Ch	HOST_RXTYPE	Sets the operating speed, transaction protocol and peripheral endpoint number for the host Receive endpoint.	Section 31.4.46
52Dh	HOST_RXINTERVAL	Sets the polling interval for Interrupt/ISOC transactions or the NAK response timeout on Bulk transactions for host Receive endpoint.	Section 31.4.47
		Control and Status Register for Endpoint 3	
530h	TXMAXP	Maximum Packet Size for Peripheral/Host Transmit Endpoint	Section 31.4.32
532h	PERI_TXCSR	Control Status Register for Peripheral Transmit Endpoint (peripheral mode)	Section 31.4.35
	HOST_TXCSR	Control Status Register for Host Transmit Endpoint (host mode)	Section 31.4.36
534h	RXMAXP	Maximum Packet Size for Peripheral/Host Receive Endpoint	Section 31.4.37
536h	PERI_RXCSR	Control Status Register for Peripheral Receive Endpoint (peripheral mode)	Section 31.4.38
	HOST_RXCSR	Control Status Register for Host Receive Endpoint (host mode)	Section 31.4.39
538h	RXCOUNT	Number of Bytes in Host Receive endpoint FIFO	Section 31.4.41
53Ah	HOST_TXTYPE	Sets the operating speed, transaction protocol and peripheral endpoint number for the host Transmit endpoint.	Section 31.4.43
53Bh	HOST_TXINTERVAL	Sets the polling interval for Interrupt/ISOC transactions or the NAK response timeout on Bulk transactions for host Transmit endpoint.	Section 31.4.45
53Ch	HOST_RXTYPE	Sets the operating speed, transaction protocol and peripheral endpoint number for the host Receive endpoint.	Section 31.4.46
53Dh	HOST_RXINTERVAL	Sets the polling interval for Interrupt/ISOC transactions or the NAK response timeout on Bulk transactions for host Receive endpoint.	Section 31.4.47
		Control and Status Register for Endpoint 4	
540h	TXMAXP	Maximum Packet Size for Peripheral/Host Transmit Endpoint	Section 31.4.32
542h	PERI_TXCSR	Control Status Register for Peripheral Transmit Endpoint (peripheral mode)	Section 31.4.35
	HOST_TXCSR	Control Status Register for Host Transmit Endpoint (host mode)	Section 31.4.36
544h	RXMAXP	Maximum Packet Size for Peripheral/Host Receive Endpoint	Section 31.4.37
546h	PERI_RXCSR	Control Status Register for Peripheral Receive Endpoint (peripheral mode)	Section 31.4.38
	HOST_RXCSR	Control Status Register for Host Receive Endpoint (host mode)	Section 31.4.39
548h	RXCOUNT	Number of Bytes in Host Receive endpoint FIFO	Section 31.4.41
54Ah	HOST_TXTYPE	Sets the operating speed, transaction protocol and peripheral endpoint number for the host Transmit endpoint.	Section 31.4.43
54Bh	HOST_TXINTERVAL	Sets the polling interval for Interrupt/ISOC transactions or the NAK response timeout on Bulk transactions for host Transmit endpoint.	Section 31.4.45
54Ch	HOST_RXTYPE	Sets the operating speed, transaction protocol and peripheral endpoint number for the host Receive endpoint.	Section 31.4.46
54Dh	HOST_RXINTERVAL	Sets the polling interval for Interrupt/ISOC transactions or the NAK response timeout on Bulk transactions for host Receive endpoint.	Section 31.4.47

Copyright © 2013–2016, Texas Instruments Incorporated

Table 31-30. Universal Serial Bus OTG (USB0) Registers (continued)

VBUS Slave Address Offset	Acronym	Register Description	Section
		CDMA Registers	
1000h	DMAREVID	CDMA Revision Identification Register	Section 31.4.66
1004h	TDFDQ	CDMA Teardown Free Descriptor Queue Control Register	Section 31.4.67
1008h	DMAEMU	CDMA Emulation Control Register	Section 31.4.68
1800h	TXGCR[0]	Transmit Channel 0 Global Configuration Register	Section 31.4.69
1808h	RXGCR[0]	Receive Channel 0 Global Configuration Register	Section 31.4.70
180Ch	RXHPCRA[0]	Receive Channel 0 Host Packet Configuration Register A	Section 31.4.71
1810h	RXHPCRB[0]	Receive Channel 0 Host Packet Configuration Register B	Section 31.4.72
1820h	TXGCR[1]	Transmit Channel 1 Global Configuration Register	Section 31.4.69
1828h	RXGCR[1]	Receive Channel 1 Global Configuration Register	Section 31.4.70
182Ch	RXHPCRA[1]	Receive Channel 1 Host Packet Configuration Register A	Section 31.4.71
1830h	RXHPCRB[1]	Receive Channel 1 Host Packet Configuration Register B	Section 31.4.72
1840h	TXGCR[2]	Transmit Channel 2 Global Configuration Register	Section 31.4.69
1848h	RXGCR[2]	Receive Channel 2 Global Configuration Register	Section 31.4.70
184Ch	RXHPCRA[2]	Receive Channel 2 Host Packet Configuration Register A	Section 31.4.71
1850h	RXHPCRB[2]	Receive Channel 2 Host Packet Configuration Register B	Section 31.4.72
1860h	TXGCR[3]	Transmit Channel 3 Global Configuration Register	Section 31.4.69
1868h	RXGCR[3]	Receive Channel 3 Global Configuration Register	Section 31.4.70
186Ch	RXHPCRA[3]	Receive Channel 3 Host Packet Configuration Register A	Section 31.4.71
1870h	RXHPCRB[3]	Receive Channel 3 Host Packet Configuration Register B	Section 31.4.72
2000h	DMA_SCHED_CTRL	CDMA Scheduler Control Register	Section 31.4.73
2800h-28FCh	WORD[0]-WORD[63]	CDMA Scheduler Table Word 0-63 Registers	Section 31.4.74
		Queue Manager (QMGR) Registers	
4000h	QMGRREVID	QMGR Revision Identification Register	Section 31.4.75
4008h	DIVERSION	QMGR Queue Diversion Register	Section 31.4.76
4020h	FDBSC0	QMGR Free Descriptor/Buffer Starvation Count Register 0	Section 31.4.77
4024h	FDBSC1	QMGR Free Descriptor/Buffer Starvation Count Register 1	Section 31.4.78
4028h	FDBSC2	QMGR Free Descriptor/Buffer Starvation Count Register 2	Section 31.4.79
402Ch	FDBSC3	QMGR Free Descriptor/Buffer Starvation Count Register 3	Section 31.4.80
4080h	LRAM0BASE	QMGR Linking RAM Region 0 Base Address Register	Section 31.4.81
4084h	LRAM0SIZE	QMGR Linking RAM Region 0 Size Register	Section 31.4.82
4088h	LRAM1BASE	QMGR Linking RAM Region 1 Base Address Register	Section 31.4.83
4090h	PEND0	QMGR Queue Pending Register 0	Section 31.4.84
4094h	PEND1	QMGR Queue Pending Register 1	Section 31.4.85
5000h + 16 × *R*	QMEMRBASE[*R*]	QMGR Memory Region *R* Base Address Register (*R* = 0 to 15)	Section 31.4.86
5004h + 16 × *R*	QMEMRCTRL[*R*]	QMGR Memory Region *R* Control Register (*R* = 0 to 15)	Section 31.4.87
600Ch + 16 × *N*	CTRLD[*N*]	QMGR Queue *N* Control Register D (*N* = 0 to 63)	Section 31.4.88
6800h + 16 × *N*	QSTATA[*N*]	QMGR Queue *N* Status Register A (*N* = 0 to 63)	Section 31.4.89
6804h + 16 × *N*	QSTATB[*N*]	QMGR Queue *N* Status Register B (*N* = 0 to 63)	Section 31.4.90
6808h + 16 × *N*	QSTATC[*N*]	QMGR Queue *N* Status Register C (*N* = 0 to 63)	Section 31.4.91

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.1 Revision Identification Register (REVID)

The revision identification register (REVID) contains the revision for the USB 2.0 OTG controller module. The REVID is shown in Figure 31-27 and described in Table 31-31.

Figure 31-27. Revision Identification Register (REVID)

31	0
REV	
R-4EA1 0800h	

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

Table 31-31. Revision Identification Register (REVID) Field Descriptions

Bit	Field	Value	Description
31-0	REV	4EA1 0800h	Revision ID of the USB module.

31.4.2 Control Register (CTRLR)

The control register (CTRLR) allows the CPU to control various aspects of the module. The CTRLR is shown in Figure 31-28 and described in Table 31-32.

Figure 31-28. Control Register (CTRLR)

31	16
Reserved	
R-0	

15	5	4	3	2	1	0
Reserved		RNDIS	UINT	Reserved	CLKFACK	RESET
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 31-32. Control Register (CTRLR) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4	RNDIS		Global RNDIS mode enable for all endpoints.
		0	Global RNDIS mode is disabled.
		1	Global RNDIS mode is enabled.
3	UINT		USB non-PDR interrupt handler enable.
		0	PDR interrupt handler is enabled.
		1	PDR interrupt handler is disabled.
2	Reserved	0	Reserved
1	CLKFACK		Clock stop fast ACK enable.
		0	Clock stop fast ACK is disabled.
		1	Clock stop fast ACK is enabled.
0	RESET		Soft reset.
		0	No effect.
		1	Writing a 1 starts a module reset. The USB controller will clear this bit when it completes reset.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.3 Status Register (STATR)

The status register (STATR) allows the CPU to check various aspects of the module. The STATR is shown in Figure 31-29 and described in Table 31-33.

Figure 31-29. Status Register (STATR)

31	16
Reserved	
R-0	

15	1	0
Reserved		DRVVBUS
R-0		R-0

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

Table 31-33. Status Register (STATR) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	DRVVBUS		Current DRVVBUS value.
		0	DRVVBUS value is logic 0.
		1	DRVVBUS value is logic 1.

31.4.4 Emulation Register (EMUR)

The emulation register (EMUR) allows the CPU to configure the CBA 3.0 emulation interface. The EMUR is shown in Figure 31-30 and described in Table 31-34.

Figure 31-30. Emulation Register (EMUR)

31	16
Reserved	
R-0	

15	3	2	1	0
Reserved		RT_SEL	SOFT	FREERUN
R-0		R/W-0	R/W-1	R/W-1

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

Table 31-34. Emulation Register (EMUR) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2	RT_SEL		Real-time enable
		0	Enable
		1	No effect
1	SOFT		Soft stop
		0	No effect
		1	Soft stop enable
0	FREERUN		Free run
		0	No effect
		1	Free run enable

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.5 Mode Register (MODE)

The mode register (MODE) allows the CPU to individually enable RNDIS/Generic/CDC modes for each endpoint. Using the global RNDIS bit in the control register (CTRLR) overrides this register and enables RNDIS mode for all endpoints. The MODE is shown in Figure 31-31 and described in Table 31-35.

Figure 31-31. Mode Register (MODE)

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
Reserved	RX4_MODE	Reserved	RX3_MODE	Reserved	RX2_MODE	Reserved	RX1_MODE
R-0	R/W-0	R-0	R/W-0	R-0	R/W-0	R-0	R/W-0

15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
Reserved	TX4_MODE	Reserved	TX3_MODE	Reserved	TX2_MODE	Reserved	TX1_MODE
R-0	R/W-0	R-0	R/W-0	R-0	R/W-0	R-0	R/W-0

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

Table 31-35. Mode Register (MODE) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Reserved
29-28	RX4_MODE	0-3h	Receive endpoint 4 mode control
		0	Transparent mode on Receive endpoint 4
		1h	RNDIS mode on Receive endpoint 4
		2h	CDC mode on Receive endpoint 4
		3h	Generic RNDIS mode on Receive endpoint 4
27-26	Reserved	0	Reserved
25-24	RX3_MODE	0-3h	Receive endpoint 3 mode control
		0	Transparent mode on Receive endpoint 3
		1h	RNDIS mode on Receive endpoint 3
		2h	CDC mode on Receive endpoint 3
		3h	Generic RNDIS mode on Receive endpoint 3
23-22	Reserved	0	Reserved
21-20	RX2_MODE	0-3h	Receive endpoint 2 mode control
		0	Transparent mode on Receive endpoint 2
		1h	RNDIS mode on Receive endpoint 2
		2h	CDC mode on Receive endpoint 2
		3h	Generic RNDIS mode on Receive endpoint 2
19-18	Reserved	0	Reserved
17-16	RX1_MODE	0-3h	Receive endpoint 1 mode control
		0	Transparent mode on Receive endpoint 1
		1h	RNDIS mode on Receive endpoint 1
		2h	CDC mode on Receive endpoint 1
		3h	Generic RNDIS mode on Receive endpoint 1
15-14	Reserved	0	Reserved
13-12	TX4_MODE	0-3h	Transmit endpoint 4 mode control
		0	Transparent mode on Transmit endpoint 4
		1h	RNDIS mode on Transmit endpoint 4
		2h	CDC mode on Transmit endpoint 4
		3h	Generic RNDIS mode on Transmit endpoint 4
11-10	Reserved	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

Table 31-35. Mode Register (MODE) Field Descriptions (continued)

Bit	Field	Value	Description
9-8	TX3_MODE	0-3h	Transmit endpoint 3 mode control
		0	Transparent mode on Transmit endpoint 3
		1h	RNDIS mode on Transmit endpoint 3
		2h	CDC mode on Transmit endpoint 3
		3h	Generic RNDIS mode on Transmit endpoint 3
7-6	Reserved	0	Reserved
5-4	TX2_MODE	0-3h	Transmit endpoint 2 mode control
		0	Transparent mode on Transmit endpoint 2
		1h	RNDIS mode on Transmit endpoint 2
		2h	CDC mode on Transmit endpoint 2
		3h	Generic RNDIS mode on Transmit endpoint 2
3-2	Reserved	0	Reserved
1-0	TX1_MODE	0-3h	Transmit endpoint 1 mode control
		0	Transparent mode on Transmit endpoint 1
		1h	RNDIS mode on Transmit endpoint 1
		2h	CDC mode on Transmit endpoint 1
		3h	Generic RNDIS mode on Transmit endpoint 1

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.6 Auto Request Register (AUTOREQ)

The auto request register (AUTOREQ) allows the CPU to enable an automatic IN token request generation for host mode RX operation per each RX endpoint. This feature has the DMA set the REQPKT bit in the control status register for host receive endpoint (HOST_RXCSR) when it clears the RXPKTRDY bit after reading out a packet. The REQPKT bit is used by the core to generate an IN token to receive data. By using this feature, the host can automatically generate an IN token after the DMA finishes receiving data and empties an endpoint buffer, thus receiving the next data packet as soon as possible from the connected device. Without this feature, the CPU will have to manually set the REQPKT bit for every USB packet.

There are two modes that auto request can function in: always or all except an EOP. The always mode sets the REQPKT bit after every USB packet the DMA receives thus generating a new IN token after each USB packet. The EOP mode sets the REQPKT bit after every USB packet that is not an EOP (end of packet) in the CPPI descriptor. For RNDIS, CDC, and Generic RNDIS modes, the auto request stops when the EOP is received (either via a short packet for RNDIS, CDC, and Generic RNDIS or the count is reached for Generic RNDIS), making it useful for starting a large RNDIS packet and having it auto generate IN tokens until the end of the RNDIS packet. For transparent mode, every USB packet is an EOP CPPI packet so the auto request never functions and acts like auto request is disabled.

The AUTOREQ is shown in Figure 31-32 and described in Table 31-36.

Figure 31-32. Auto Request Register (AUTOREQ)

31	16
Reserved	
R-0	

15	8	7	6	5	4	3	2	1	0
Reserved		RX4_AUTREQ		RX3_AUTREQ		RX2_AUTREQ		RX1_AUTREQ	
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 31-36. Auto Request Register (AUTOREQ) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-6	RX4_AUTREQ	0-3h	Receive endpoint 4 auto request enable
		0	No auto request
		1h	Auto request on all but EOP
		2h	Reserved
		3h	Auto request always
5-4	RX3_AUTREQ	0-3h	Receive endpoint 3 auto request enable
		0	No auto request
		1h	Auto request on all but EOP
		2h	Reserved
		3h	Auto request always
3-2	RX2_AUTREQ	0-3h	Receive endpoint 2 auto request enable
		0	No auto request
		1h	Auto request on all but EOP
		2h	Reserved
		3h	Auto request always
1-0	RX1_AUTREQ	0-3h	Receive endpoint 1 auto request enable
		0	No auto request
		1h	Auto request on all but EOP
		2h	Reserved
		3h	Auto request always

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.7 SRP Fix Time Register (SRPFIXTIME)

The SRP fix time register (SRPFIXTIME) allows the CPU to configure the maximum amount of time the SRP fix logic blocks the Avalid from the PHY to the OTG core. The SRPFIXTIME is shown in Figure 31-33 and described in Table 31-37.

Figure 31-33. SRP Fix Time Register (SRPFIXTIME)

31	0
SRPFIXTIME	
R/W-280 DE80h	

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

Table 31-37. SRP Fix Time Register (SRPFIXTIME) Field Descriptions

Bit	Field	Value	Description
31-0	SRPFIXTIME	0-FFFF FFFFh	SRP fix maximum time in 60 MHz cycles. Default is 700 ms (280 DE80h).

31.4.8 Teardown Register (TEARDOWN)

The teardown register (TEARDOWN) controls the tearing down of receive and transmit FIFOs in the USB controller. When a 1 is written to a valid bit in TEARDOWN, the CPPI FIFO pointers for that endpoint are cleared. TEARDOWN must be used in conjunction with the CPPI DMA teardown mechanism. The Host should also write the FLUSHFIFO bits in the TXCSR and RXCSR registers to ensure a complete teardown of the endpoint.

The TEARDOWN is shown in Figure 31-34 and described in Table 31-38.

Figure 31-34. Teardown Register (TEARDOWN)

31	21	20	17	16
Reserved		TX_TDOWN		Rsvd
R-0		R/W-0		R-0

15	5	4	1	0
Reserved		RX_TDOWN		Rsvd
R-0		R/W-0		R-0

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

Table 31-38. Teardown Register (TEARDOWN) Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	0	Reserved
20-17	TX_TDOWN		Transmit endpoint teardown. Set the bit that corresponds to the Endpoint (for EP1, set bit 17; for EP2, set bit 18; for EP3, set bit 19; for EP4, set bit 20).
		0	Disable
		1	Enable
16-5	Reserved	0	Reserved
4-1	RX_TDOWN		Receive endpoint teardown
		0	Disable
		1	Enable
0	Reserved	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.9 USB Interrupt Source Register (INTSRCR)

The USB interrupt source register (INTSRCR) contains the status of the interrupt sources generated by the USB core (not by the DMA). The INTSRCR is shown in Figure 31-35 and described in Table 31-39.

NOTE: Other than the USB bit field, to make use of INTSRCR, the PDR interrupt handler must be enabled (the UINT bit in the control register (CTRLR) is cleared to 0). If the UINT bit in CTRLR is set to 1, you need to use the interrupt status/flag from the core register space.

Figure 31-35. USB Interrupt Source Register (INTSRCR)

31	25	24	16
Reserved		USB	
R-0		R-0	

15	13	12	9	8	5	4	1	0
Reserved		RXEP[*n*]		Reserved		TXEP[*n*]		EP0
R-0		R-0		R-0		R-0		R-0

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

Table 31-39. USB Interrupt Source Register (INTSRCR) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reserved
24-16	USB	0-1FFh	USB interrupt sources. Generated by the USB core (not by the DMA). Note: INTRUSB core interrupts are mapped onto bits 23-16 and bit 24 is the USBDRVVBUS interrupt status.
15-13	Reserved	0	Reserved
12-9	RXEP[*n*]		Receive endpoint *n* interrupt source.
		0	RXEP*n* interrupt is not generated by the USB core.
		1	RXEP*n* interrupt is generated by the USB core (not by the DMA).
8-5	Reserved	0	Reserved
4-1	TXEP[*n*]		Transmit endpoint *n* interrupt source.
		0	TXEP*n* interrupt is not generated by the USB core.
		1	TXEP*n* interrupt is generated by the USB core (not by the DMA).
0	EP0		Endpoint 0 interrupt source.
		0	Endpoint 0 interrupt is not generated by the USB core.
		1	Endpoint 0 interrupt is generated by the USB core (not by the DMA).

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.10 USB Interrupt Source Set Register (INTSETR)

The USB interrupt source set register (INTSETR) allows the USB interrupt sources to be manually triggered. A read of this register returns the USB interrupt source register value. The INTSETR is shown in Figure 31-36 and described in Table 31-40.

NOTE: Other than the USB bit field, to make use of INTSETR, the PDR interrupt handler must be enabled (the UINT bit in the control register (CTRLR) is cleared to 0). If the UINT bit in CTRLR is set to 1, you need to use the interrupt status/flag from the core register space.

Figure 31-36. USB Interrupt Source Set Register (INTSETR)

31	25	24	16
Reserved		USB	
R-0		R/W-0	

15	13	12	9	8	5	4	1	0
Reserved		RXEP[*n*]		Reserved		TXEP[*n*]		EP0
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 31-40. USB Interrupt Source Set Register (INTSETR) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reserved
24-16	USB	0-1FFh	Write a 1 to set equivalent USB interrupt source. Allows the USB interrupt sources to be manually triggered.
15-13	Reserved	0	Reserved
12-9	RXEP[*n*]		Set receive endpoint *n* interrupt source. Allows the receive endpoint *n* interrupt sources to be manually triggered.
		0	RXEP*n* interrupt is not set.
		1	RXEP*n* interrupt is set.
8-5	Reserved	0	Reserved
4-1	TXEP[*n*]		Set transmit endpoint *n* interrupt source. Allows the transmit endpoint *n* interrupt sources to be manually triggered.
		0	TXEP*n* interrupt is not set.
		1	TXEP*n* interrupt is set.
0	EP0		Set endpoint 0 interrupt source. Allows the endpoint 0 interrupt source to be manually triggered.
		0	Endpoint 0 interrupt is not set.
		1	Endpoint 0 interrupt is set.

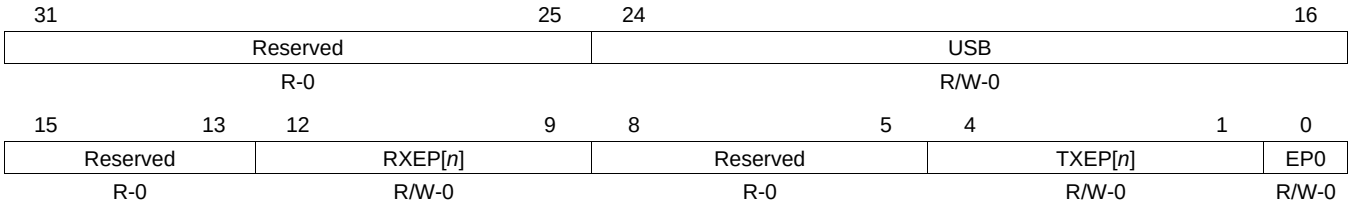
Copyright © 2013–2016, Texas Instruments Incorporated

31.4.11 USB Interrupt Source Clear Register (INTCLRR)

The USB interrupt source clear register (INTCLRR) allows the CPU to acknowledge an interrupt source and turn it off. A read of this register returns the USB interrupt source register value. The INTCLRR is shown in Figure 31-37 and described in Table 31-41.

NOTE: Other than the USB bit field, to make use of INTCLRR, the PDR interrupt handler must be enabled (the UINT bit in the control register (CTRLR) is cleared to 0). If the UINT bit in CTRLR is set to 1, you need to use the interrupt status/flag from the core register space.

Figure 31-37. USB Interrupt Source Clear Register (INTCLRR)



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

Table 31-41. USB Interrupt Source Clear Register (INTCLRR) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reserved
24-16	USB	0-1FFh	Write a 1 to clear equivalent USB interrupt source. Allows the CPU to acknowledge a USB interrupt source and turn it off.
15-13	Reserved	0	Reserved
12-9	RXEP[*n*]		Clear receive endpoint *n* interrupt source. Allows the CPU to acknowledge a receive endpoint *n* interrupt source and turn it off.
		0	RXEP*n* interrupt is not cleared.
		1	RXEP*n* interrupt is cleared.
8-5	Reserved	0	Reserved
4-1	TXEP[*n*]		Clear transmit endpoint *n* interrupt source. Allows the CPU to acknowledge a transmit endpoint *n* interrupt source and turn it off.
		0	TXEP*n* interrupt is not cleared.
		1	TXEP*n* interrupt is cleared.
0	EP0		Clear endpoint 0 interrupt source. Allows the CPU to acknowledge the endpoint 0 interrupt source and turn it off.
		0	Endpoint 0 interrupt is not cleared.
		1	Endpoint 0 interrupt is cleared.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.12 USB Interrupt Mask Register (INTMSKR)

The USB interrupt mask register (INTMSKR) contains the masks of the interrupt sources generated by the USB core (not by the DMA). These masks are used to enable or disable interrupt sources generated on the masked source interrupts (the raw source interrupts are never masked). The bit positions are maintained in the same position as the interrupt sources in the USB interrupt source register (INTSRCR).

The INTMSKR is shown in Figure 31-38 and described in Table 31-42.

NOTE: Other than the USB bit field, to make use of INTMSKR, the PDR interrupt handler must be enabled (the UINT bit in the control register (CTRLR) is cleared to 0). If the UINT bit in CTRLR is set to 1, you need to use the interrupt status/flag from the core register space.

Figure 31-38. USB Interrupt Mask Register (INTMSKR)

31	25	24	16
Reserved		USB	
R-0		R-0	

15	13	12	9	8	5	4	1	0
Reserved		RXEP[*n*]		Reserved		TXEP[*n*]		EP0
R-0		R-0		R-0		R-0		R-0

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

Table 31-42. USB Interrupt Mask Register (INTMSKR) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reserved
24-16	USB	0-1FFh	USB interrupt source masks. Generated by the USB core (not by the DMA).
15-13	Reserved	0	Reserved
12-9	RXEP[*n*]		Receive endpoint *n* interrupt source mask.
		0	RXEP*n* interrupt mask is not generated by the USB core.
		1	RXEP*n* interrupt mask is generated by the USB core (not by the DMA).
8-5	Reserved	0	Reserved
4-1	TXEP[*n*]		Transmit endpoint *n* interrupt source mask.
		0	TXEP*n* interrupt mask is not generated by the USB core.
		1	TXEP*n* interrupt mask is generated by the USB core (not by the DMA).
0	EP0		Endpoint 0 interrupt source mask.
		0	Endpoint 0 interrupt mask is not generated by the USB core.
		1	Endpoint 0 interrupt mask is generated by the USB core (not by the DMA).

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.13 USB Interrupt Mask Set Register (INTMSKSETR)

The USB interrupt mask set register (INTMSKSETR) allows the USB masks to be individually enabled. A read to this register returns the USB interrupt mask register value. The INTMSKSETR is shown in Figure 31-39 and described in Table 31-43.

NOTE: Other than the USB bit field, to make use of INTMSKSETR, the PDR interrupt handler must be enabled (the UINT bit in the control register (CTRLR) is cleared to 0). If the UINT bit in CTRLR is set to 1, you need to use the interrupt status/flag from the core register space.

Figure 31-39. USB Interrupt Mask Set Register (INTMSKSETR)

31	25	24	16
Reserved		USB	
R-0		R/W-0	

15	13	12	9	8	5	4	1	0
Reserved		RXEP[*n*]		Reserved		TXEP[*n*]		EP0
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 31-43. USB Interrupt Mask Set Register (INTMSKSETR) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reserved
24-16	USB	0-1FFh	Write a 1 to set equivalent USB interrupt source mask. Allows the USB interrupt source masks to be manually enabled.
15-13	Reserved	0	Reserved
12-9	RXEP[*n*]		Set receive endpoint *n* interrupt source mask. Allows the receive endpoint *n* interrupt source masks to be manually enabled.
		0	RXEP*n* interrupt mask is not enabled.
		1	RXEP*n* interrupt mask is enabled.
8-5	Reserved	0	Reserved
4-1	TXEP[*n*]		Set transmit endpoint *n* interrupt source mask. Allows the transmit endpoint *n* interrupt source masks to be manually enabled.
		0	TXEP*n* interrupt mask is not enabled.
		1	TXEP*n* interrupt mask is enabled.
0	EP0		Set endpoint 0 interrupt source mask. Allows the endpoint 0 interrupt source mask to be manually enabled.
		0	Endpoint 0 interrupt mask is not enabled.
		1	Endpoint 0 interrupt mask is enabled.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.14 USB Interrupt Mask Clear Register (INTMSKCLRR)

The USB interrupt mask clear register (INTMSKCLRR) allows the USB interrupt masks to be individually disabled. A read to this register returns the USB interrupt mask register value. The INTMSKCLRR is shown in Figure 31-40 and described in Table 31-44.

NOTE: Other than the USB bit field, to make use of INTMSKCLRR, the PDR interrupt handler must be enabled (the UINT bit in the control register (CTRLR) is cleared to 0). If the UINT bit in CTRLR is set to 1, you need to use the interrupt status/flag from the core register space.

Figure 31-40. USB Interrupt Mask Clear Register (INTMSKCLRR)

31	25	24	16
Reserved		USB	
R-0		R/W-0	

15	13	12	9	8	5	4	1	0
Reserved		RXEP[*n*]		Reserved		TXEP[*n*]		EP0
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 31-44. USB Interrupt Mask Clear Register (INTMSKCLRR) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reserved
24-16	USB	0-1FFh	Write a 1 to clear equivalent USB interrupt source mask. Allows the USB interrupt source masks to be manually disabled.
15-13	Reserved	0	Reserved
12-9	RXEP[*n*]		Clear receive endpoint *n* interrupt source mask. Allows the receive endpoint *n* interrupt source masks to be manually disabled.
		0	RXEP*n* interrupt mask is not disabled.
		1	RXEP*n* interrupt mask is disabled.
8-5	Reserved	0	Reserved
4-1	TXEP[*n*]		Clear transmit endpoint *n* interrupt source mask. Allows the transmit endpoint *n* interrupt source masks to be manually disabled.
		0	TXEP*n* interrupt mask is not disabled.
		1	TXEP*n* interrupt mask is disabled.
0	EP0		Clear endpoint 0 interrupt source mask. Allows the endpoint 0 interrupt source mask to be manually disabled.
		0	Endpoint 0 interrupt mask is not disabled.
		1	Endpoint 0 interrupt mask is disabled.

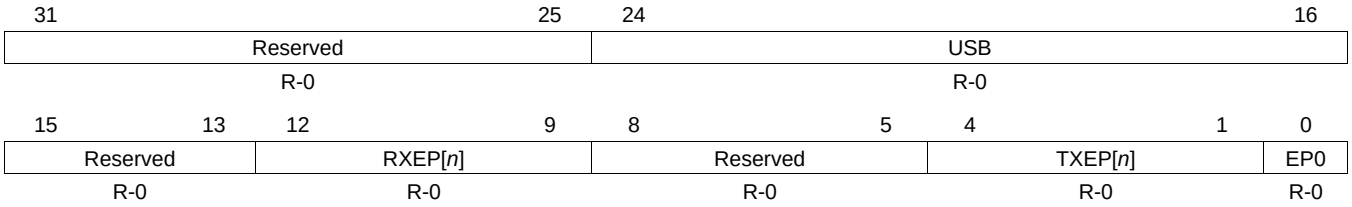
Copyright © 2013–2016, Texas Instruments Incorporated

31.4.15 USB Interrupt Source Masked Register (INTMASKEDR)

The USB interrupt source masked register (INTMASKEDR) contains the status of the interrupt sources generated by the USB core masked by the USB interrupt mask register (INTMSKR) values. The INTMASKEDR is shown in Figure 31-41 and described in Table 31-45.

NOTE: Other than the USB bit field, to make use of INTMASKEDR, the PDR interrupt handler must be enabled (the UINT bit in the control register (CTRLR) is cleared to 0). If the UINT bit in CTRLR is set to 1, you need to use the interrupt status/flag from the core register space.

Figure 31-41. USB Interrupt Source Masked Register (INTMASKEDR)



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

Table 31-45. USB Interrupt Source Masked Register (INTMASKEDR) Field Descriptions

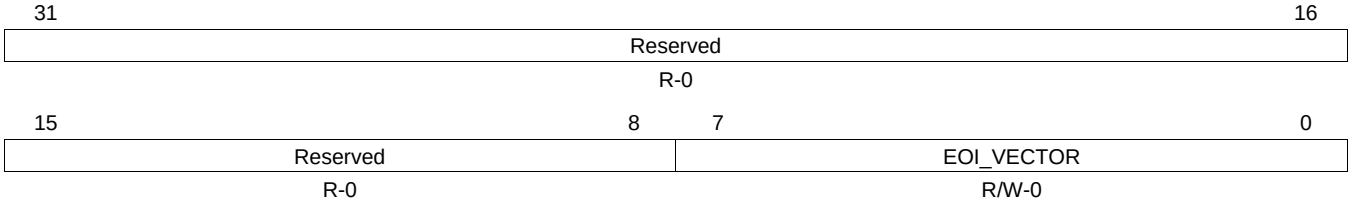
Bit	Field	Value	Description
31-25	Reserved	0	Reserved
24-16	USB	0-1FFh	USB interrupt sources masked.
15-13	Reserved	0	Reserved
12-9	RXEP[*n*]		Receive endpoint *n* interrupt source masked.
		0	RXEP*n* interrupt source is not masked.
		1	RXEP*n* interrupt source is masked.
8-5	Reserved	0	Reserved
4-1	TXEP[*n*]		Transmit endpoint *n* interrupt source masked.
		0	TXEP*n* interrupt source is not masked.
		1	TXEP*n* interrupt source is masked.
0	EP0		Endpoint 0 interrupt source masked.
		0	Endpoint 0 interrupt source is not masked.
		1	Endpoint 0 interrupt source is masked.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.16 USB End of Interrupt Register (EOIR)

The USB end of interrupt register (EOIR) allows the CPU to acknowledge completion of a non-DMA interrupt by writing 0 to the EOI_VECTOR field. The EOIR is shown in Figure 31-42 and described in Table 31-46.

Figure 31-42. USB End of Interrupt Register (EOIR)



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

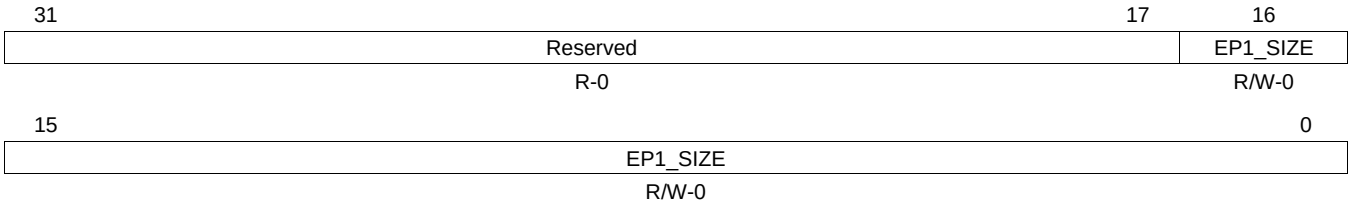
Table 31-46. USB End of Interrupt Register (EOIR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	EOI_VECTOR	0-FFh	End of Interrupt (EOI) Vector.

31.4.17 Generic RNDIS EP1 Size Register (GENRNDISSZ1)

The generic RNDIS EP1 size register (GENRNDISSZ1) is programmed with a RNDIS packet size in bytes. When EP1 is in Generic RNDIS mode, the received USB packets are collected into a single CPPI packet that is completed when the number of bytes equal to the value of this register have been received, or a *short* packet is received. This register must be programmed with a value that is an integer multiple of the endpoint size. The maximum value this register can be programmed with is 10000h, or 65536. The GENRNDISSZ1 is shown in Figure 31-43 and described in Table 31-47.

Figure 31-43. Generic RNDIS EP1 Size Register (GENRNDISSZ1)



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

Table 31-47. Generic RNDIS EP1 Size Register (GENRNDISSZ1) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16-0	EP1_SIZE	0-10000h	Generic RNDIS packet size

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.18 Generic RNDIS EP2 Size Register (GENRNDISSZ2)

The generic RNDIS EP2 size register (GENRNDISSZ2) is programmed with a RNDIS packet size in bytes. When EP2 is in Generic RNDIS mode, the received USB packets are collected into a single CPPI packet that is completed when the number of bytes equal to the value of this register have been received, or a *short* packet is received. This register must be programmed with a value that is an integer multiple of the endpoint size. The maximum value this register can be programmed with is 10000h, or 65536. The GENRNDISSZ2 is shown in Figure 31-44 and described in Table 31-48.

Figure 31-44. Generic RNDIS EP2 Size Register (GENRNDISSZ2)

31	17	16
Reserved		EP2_SIZE
R-0		R/W-0

15	0
EP2_SIZE	
R/W-0	

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

Table 31-48. Generic RNDIS EP2 Size Register (GENRNDISSZ2) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16-0	EP2_SIZE	0-10000h	Generic RNDIS packet size

31.4.19 Generic RNDIS EP3 Size Register (GENRNDISSZ3)

The generic RNDIS EP3 size register (GENRNDISSZ3) is programmed with a RNDIS packet size in bytes. When EP3 is in Generic RNDIS mode, the received USB packets are collected into a single CPPI packet that is completed when the number of bytes equal to the value of this register has been received, or a *short* packet is received. This register must be programmed with a value that is an integer multiple of the endpoint size. The maximum value this register can be programmed with is 10000h, or 65536. The GENRNDISSZ3 is shown in Figure 31-45 and described in Table 31-49.

Figure 31-45. Generic RNDIS EP3 Size Register (GENRNDISSZ3)

31	17	16
Reserved		EP3_SIZE
R-0		R/W-0

15	0
EP3_SIZE	
R/W-0	

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

Table 31-49. Generic RNDIS EP3 Size Register (GENRNDISSZ3) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16-0	EP3_SIZE	0-10000h	Generic RNDIS packet size

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.20 Generic RNDIS EP4 Size Register (GENRNDISSZ4)

The generic RNDIS EP4 size register (GENRNDISSZ4) is programmed with a RNDIS packet size in bytes. When EP4 is in Generic RNDIS mode, the received USB packets are collected into a single CPPI packet that is completed when the number of bytes equal to the value of this register has been received, or a *short* packet is received. This register must be programmed with a value that is an integer multiple of the endpoint size. The maximum value this register can be programmed with is 10000h, or 65536. The GENRNDISSZ4 is shown in Figure 31-46 and described in Table 31-50.

Figure 31-46. Generic RNDIS EP4 Size Register (GENRNDISSZ4)

31	17	16
Reserved		EP4_SIZE
R-0		R/W-0

15	0
EP4_SIZE	
R/W-0	

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

Table 31-50. Generic RNDIS EP4 Size Register (GENRNDISSZ4) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16-0	EP4_SIZE	0-10000h	Generic RNDIS packet size

31.4.21 Function Address Register (FADDR)

The function address register (FADDR) is shown in Figure 31-47 and described in Table 31-51.

Figure 31-47. Function Address Register (FADDR)

7	6	0
Reserved	FUNCADDR	
R-0	R/W-0	

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

Table 31-51. Function Address Register (FADDR) Field Descriptions

Bit	Field	Value	Description
7	Reserved	0	Reserved
6-0	FUNCADDR	0-7Fh	7_bit address of the peripheral part of the transaction. When used in Peripheral mode, this register should be written with the address received through a SET_ADDRESS command, which will then be used for decoding the function address in subsequent token packets. When used in Host mode, this register should be set to the value sent in a SET_ADDRESS command during device enumeration as the address for the peripheral device.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.22 Power Management Register (POWER)

The power management register (POWER) is shown in Figure 31-48 and described in Table 31-52.

Figure 31-48. Power Management Register (POWER)

7	6	5	4	3	2	1	0
ISOUPDATE	SOFTCONN	HSEN	HSMODE	RESET	RESUME	SUSPENDM	ENSUSPM
R/W-0	R/W-0	R/W-1	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 31-52. Power Management Register (POWER) Field Descriptions

Bit	Field	Value	Description
7	ISOUPDATE	0-1	When set, the USB controller will wait for an SOF token from the time TxPktRdy is set before sending the packet. If an IN token is received before an SOF token, then a zero length data packet will be sent. Note: this is only valid in Peripheral Mode. This bit only affects endpoints performing Isochronous transfers.
6	SOFTCONN	0-1	If Soft Connect/Disconnect feature is enabled, then the USB D+/D- lines are enabled when this bit is set and tri-stated when this bit is cleared. Note: this is only valid in Peripheral Mode.
5	HSEN	0-1	When set, the USB controller will negotiate for high-speed mode when the device is reset by the hub. If not set, the device will only operate in full-speed mode.
4	HSMODE	0-1	This bit is set when the USB controller has successfully negotiated for high-speed mode.
3	RESET	0-1	This bit is set when Reset signaling is present on the bus. Note: this bit is Read/Write in Host Mode, but read-only in Peripheral Mode.
2	RESUME	0-1	Set to generate Resume signaling when the controller is in Suspend mode. The bit should be cleared after 10 ms (a maximum of 15 ms) to end Resume signaling. In Host mode, this bit is also automatically set when Resume signaling from the target is detected while the USB controller is suspended.
1	SUSPENDM	0-1	In Host mode, this bit should be set to enter Suspend mode. In Peripheral mode, this bit is set on entry into Suspend mode. It is cleared when the interrupt register is read, or the RESUME bit is set.
0	ENSUSPM	0-1	Set to enable the SUSPENDM output.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.23 Interrupt Register for Endpoint 0 Plus Transmit Endpoints 1 to 4 (INTRTX)

The interrupt register for endpoint 0 plus transmit endpoints 1 to 4 (INTRTX) is shown in Figure 31-49 and described in Table 31-53.

NOTE: Unless the UINT bit in the control register (CTRLR) is set to 1 (non-PDR interrupt mode is enabled), do not read this register directly. Performing a read clears the pending interrupt. Use INTRTX only when in the non-PDR interrupt mode, that is, when handling the interrupt directly from the controller.

Figure 31-49. Interrupt Register for Endpoint 0 Plus Tx Endpoints 1 to 4 (INTRTX)

15	8
Reserved	
R-0	

7	5	4	3	2	1	0
Reserved		EP4TX	EP3TX	EP2TX	EP1TX	EP0
R-0		R-0	R-0	R-0	R-0	R-0

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

Table 31-53. Interrupt Register for Endpoint 0 Plus Transmit Endpoints 1 to 4 (INTRTX) Field Descriptions

Bit	Field	Value	Description
15-5	Reserved	0	Reserved
4	EP4TX	0-1	Transmit Endpoint 4 interrupt active
3	EP3TX	0-1	Transmit Endpoint 3 interrupt active
2	EP2TX	0-1	Transmit Endpoint 2 interrupt active
1	EP1TX	0-1	Transmit Endpoint 1 interrupt active
0	EP0	0-1	Endpoint 0 interrupt active

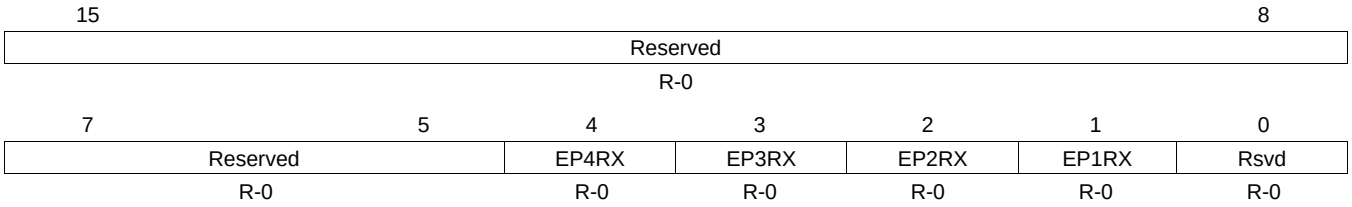
Copyright © 2013–2016, Texas Instruments Incorporated

31.4.24 Interrupt Register for Receive Endpoints 1 to 4 (INTRRX)

The interrupt register for receive endpoints 1 to 4 (INTRRX) is shown in Figure 31-50 and described in Table 31-54.

NOTE: Unless the UINT bit in the control register (CTRLR) is set to 1 (non-PDR interrupt mode is enabled), do not read this register directly. Performing a read clears the pending interrupt. Use INTRRX only when in the non-PDR interrupt mode, that is, when handling the interrupt directly from the controller.

Figure 31-50. Interrupt Register for Receive Endpoints 1 to 4 (INTRRX)



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

Table 31-54. Interrupt Register for Receive Endpoints 1 to 4 (INTRRX) Field Descriptions

Bit	Field	Value	Description
15-5	Reserved	0	Reserved
4	EP4RX	0-1	Receive Endpoint 4 interrupt active
3	EP3RX	0-1	Receive Endpoint 3 interrupt active
2	EP2RX	0-1	Receive Endpoint 2 interrupt active
1	EP1RX	0-1	Receive Endpoint 1 interrupt active
0	Reserved	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.25 Interrupt Enable Register for INTRTX (INTRTXE)

The interrupt enable register for INTRTX (INTRTXE) is shown in Figure 31-51 and described in Table 31-55.

Figure 31-51. Interrupt Enable Register for INTRTX (INTRTXE)

15	8
Reserved	
R-0	

7	5	4	3	2	1	0
Reserved		EP4TX	EP3TX	EP2TX	EP1TX	EP0
R-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 31-55. Interrupt Enable Register for INTRTX (INTRTXE) Field Descriptions

Bit	Field	Value	Description
15-5	Reserved	0	Reserved
4	EP4TX	0-1	Transmit Endpoint 4 interrupt active
3	EP3TX	0-1	Transmit Endpoint 3 interrupt active
2	EP2TX	0-1	Transmit Endpoint 2 interrupt active
1	EP1TX	0-1	Transmit Endpoint 1 interrupt active
0	EP0	0-1	Endpoint 0 interrupt active

31.4.26 Interrupt Enable Register for INTRRX (INTRRXE)

The interrupt enable register for INTRRX (INTRRXE) is shown in Figure 31-52 and described in Table 31-56.

Figure 31-52. Interrupt Enable Register for INTRRX (INTRRXE)

15	8
Reserved	
R-0	

7	5	4	3	2	1	0
Reserved		EP4RX	EP3RX	EP2RX	EP1RX	Reserved
R-0		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 31-56. Interrupt Enable Register for INTRRX (INTRRXE) Field Descriptions

Bit	Field	Value	Description
15-5	Reserved	0	Reserved
4	EP4RX	0-1	Receive Endpoint 4 interrupt active
3	EP3RX	0-1	Receive Endpoint 3 interrupt active
2	EP2RX	0-1	Receive Endpoint 2 interrupt active
1	EP1RX	0-1	Receive Endpoint 1 interrupt active
0	Reserved	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.27 Interrupt Register for Common USB Interrupts (INTRUSB)

The interrupt register for common USB interrupts (INTRUSB) is shown in Figure 31-53 and described in Table 31-57.

NOTE: Unless the UINT bit in the control register (CTRLR) is set to 1 (non-PDR interrupt mode is enabled), do not read this register directly. Performing a read clears the pending interrupt. Use INTRUSB only when in the non-PDR interrupt mode, that is, when handling the interrupt directly from the controller.

Figure 31-53. Interrupt Register for Common USB Interrupts (INTRUSB)

7	6	5	4	3	2	1	0
VBUSERR	SESSREQ	DISCON	CONN	SOF	RESET_BABBLE	RESUME	SUSPEND
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

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

Table 31-57. Interrupt Register for Common USB Interrupts (INTRUSB) Field Descriptions

Bit	Field	Value	Description
7	VBUSERR	0-1	Set when VBus drops below the VBus valid threshold during a session. Only valid when the USB controller is 'A' device. All active interrupts will be cleared when this register is read.
6	SESSREQ	0-1	Set when session request signaling has been detected. Only valid when USB controller is 'A' device.
5	DISCON	0-1	Set in host mode when a device disconnect is detected. Set in peripheral mode when a session ends.
4	CONN	0-1	Set when a device connection is detected. Only valid in host mode.
3	SOF	0-1	Set when a new frame starts.
2	RESET_BABBLE	0-1	Set in peripheral mode when reset signaling is detected on the bus set in host mode when babble is detected.
1	RESUME	0-1	Set when resume signaling is detected on the bus while the USB controller is in suspend mode.
0	SUSPEND	0-1	Set when suspend signaling is detected on the bus only valid in peripheral mode.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.28 Interrupt Enable Register for INTRUSB (INTRUSBE)

The interrupt enable register for INTRUSB (INTRUSBE) is shown in Figure 31-54 and described in Table 31-58.

Figure 31-54. Interrupt Enable Register for INTRUSB (INTRUSBE)

7	6	5	4	3	2	1	0
VBUSERR	SESSREQ	DISCON	CONN	SOF	RESET_BABBLE	RESUME	SUSPEND
R/W-0	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; -*n* = value after reset

Table 31-58. Interrupt Enable Register for INTRUSB (INTRUSBE) Field Descriptions

Bit	Field	Value	Description
7	VBUSERR	0-1	Vbus error interrupt enable
6	SESSREQ	0-1	Session request interrupt enable
5	DISCON	0-1	Disconnect interrupt enable
4	CONN	0-1	Connect interrupt enable
3	SOF	0-1	Start of frame interrupt enable
2	RESET_BABBLE	0-1	Reset interrupt enable
1	RESUME	0-1	Resume interrupt enable
0	SUSPEND	0-1	Suspend interrupt enable

31.4.29 Frame Number Register (FRAME)

The frame number register (FRAME) is shown in Figure 31-55 and described in Table 31-59.

Figure 31-55. Frame Number Register (FRAME)

15	11	10	0
Reserved		FRAMENUMBER	
R-0		R-0	

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

Table 31-59. Frame Number Register (FRAME) Field Descriptions

Bit	Field	Value	Description
15-11	Reserved	0	Reserved
10-0	FRAMENUMBER	0-7FFh	Last received frame number

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.30 Index Register for Selecting the Endpoint Status and Control Registers (INDEX)

The index register for selecting the endpoint status and control registers (INDEX) is shown in Figure 31-56 and described in Table 31-60.

Figure 31-56. Index Register for Selecting the Endpoint Status and Control Registers (INDEX)

7	4	3	0
Reserved		EPSEL	
R-0		R/W-0	

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

Table 31-60. Index Register for Selecting the Endpoint Status and Control Registers (INDEX) Field Descriptions

Bit	Field	Value	Description
7-4	Reserved	0	Reserved
3-0	EPSEL	0-4h	Each transmit endpoint and each receive endpoint have their own set of control/status registers. EPSEL determines which endpoint control/status registers are accessed. Before accessing an endpoint's control/status registers, the endpoint number should be written to the Index register to ensure that the correct control/status registers appear in the memory-map.

31.4.31 Register to Enable the USB 2.0 Test Modes (TESTMODE)

The register to enable the USB 2.0 test modes (TESTMODE) is shown in Figure 31-57 and described in Table 31-61.

Figure 31-57. Register to Enable the USB 2.0 Test Modes (TESTMODE)

7	6	5	4	3	2	1	0
FORCE_HOST	FIFO_ACCESS	FORCE_FS	FORCE_HS	TEST_PACKET	TEST_K	TEST_J	TEST_SE0_NAK
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; W = Write only; -*n* = value after reset

Table 31-61. Register to Enable the USB 2.0 Test Modes (TESTMODE) Field Descriptions

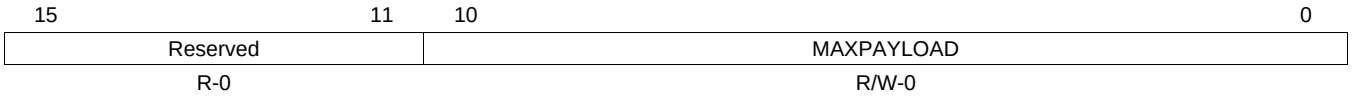
Bit	Field	Value	Description
7	FORCE_HOST	0-1	Set this bit to forcibly put the USB controller into Host mode when SESSION bit is set, regardless of whether it is connected to any peripheral. The controller remains in Host mode until the Session bit is cleared, even if a device is disconnected. And if the FORCE_HOST but remains set, it will re-enter Host mode next time the SESSION bit is set. The operating speed is determined using the FORCE_HS and FORCE_FS bits.
6	FIFO_ACCESS	0-1	Set this bit to transfer the packet in EP0 Tx FIFO to EP0 Receive FIFO. It is cleared automatically.
5	FORCE_FS	0-1	Set this bit to force the USB controller into full-speed mode when it receives a USB reset.
4	FORCE_HS	0-1	Set this bit to force the USB controller into high-speed mode when it receives a USB reset.
3	TEST_PACKET	0-1	Set this bit to enter the Test_Packet test mode. In this mode, the USB controller repetitively transmits a 53-byte test packet on the bus, the form of which is defined in the Universal Serial Bus Specification Revision 2.0. Note: The test packet has a fixed format and must be loaded into the Endpoint 0 FIFO before the test mode is entered.
2	TEST_K	0-1	Set this bit to enter the Test_K test mode. In this mode, the USB controller transmits a continuous K on the bus.
1	TEST_J	0-1	Set this bit to enter the Test_J test mode. In this mode, the USB controller transmits a continuous J on the bus.
0	TEST_SE0_NAK	0-1	Set this bit to enter the Test_SE0_NAK test mode. In this mode, the USB controller remains in high-speed mode, but responds to any valid IN token with a NAK.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.32 Maximum Packet Size for Peripheral/Host Transmit Endpoint (TXMAXP)

The maximum packet size for peripheral/host transmit endpoint (TXMAXP) is shown in Figure 31-58 and described in Table 31-62.

Figure 31-58. Maximum Packet Size for Peripheral/Host Transmit Endpoint (TXMAXP)



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

Table 31-62. Maximum Packet Size for Peripheral/Host Transmit Endpoint (TXMAXP) Field Descriptions

Bit	Field	Value	Description
15-11	Reserved	0	Reserved
10-0	MAXPAYLOAD	0-400h	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 and high-speed operations. The value written to this register should match the wMaxPacketSize field of the Standard Endpoint Descriptor for the associated endpoint. A mismatch could cause unexpected results.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.33 Control Status Register for Endpoint 0 in Peripheral Mode (PERI_CSR0)

The control status register for endpoint 0 in peripheral mode (PERI_CSR0) is shown in Figure 31-59 and described in Table 31-63.

Figure 31-59. Control Status Register for Endpoint 0 in Peripheral Mode (PERI_CSR0)

15						9	8
Reserved							FLUSHFIFO
R-0							W-0
7	6	5	4	3	2	1	0
SERV_SETUPEND	SERV_RXPKTRDY	SENDSTALL	SETUPEND	DATAEND	SENTSTALL	TXPKTRDY	RXPKTRDY
W-0	W-0	W-0	R-0	W-0	R/W-0	R/W-0	R-0

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

Table 31-63. Control Status Register for Endpoint 0 in Peripheral Mode (PERI_CSR0) Field Descriptions

Bit	Field	Value	Description
15-9	Reserved	0	Reserved
8	FLUSHFIFO	0-1	Set this bit to flush the next packet to be transmitted/read from the Endpoint 0 FIFO. The FIFO pointer is reset and the TXPKTRDY/RXPKTRDY bit is cleared. Note: FLUSHFIFO has no effect unless TXPKTRDY/RXPKTRDY is set.
7	SERV_SETUPEND	0-1	Set this bit to clear the SETUPEND bit. It is cleared automatically.
6	SERV_RXPKTRDY	0-1	Set this bit to clear the RXPKTRDY bit. It is cleared automatically.
5	SENDSTALL	0-1	Set this bit to terminate the current transaction. The STALL handshake will be transmitted and then this bit will be cleared automatically.
4	SETUPEND	0-1	This bit will be set when a control transaction ends before the DATAEND bit has been set. An interrupt will be generated, and the FIFO will be flushed at this time. The bit is cleared by the writing a 1 to the SERV_SETUPEND bit.
3	DATAEND	0-1	Set this bit to 1: a. When setting TXPKTRDY for the last data packet. b. When clearing RXPKTRDY after unloading the last data packet. c. When setting TXPKTRDY for a zero length data packet. It is cleared automatically.
2	SENTSTALL	0-1	This bit is set when a STALL handshake is transmitted. This bit should be cleared.
1	TXPKTRDY	0-1	Set this bit after loading a data packet into the FIFO. It is cleared automatically when the data packet has been transmitted. An interrupt is generated (if enabled) when the bit is cleared.
0	RXPKTRDY	0-1	This bit is set when a data packet has been received. An interrupt is generated when this bit is set. This bit is cleared by setting the SERV_RXPKTRDY bit.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.34 Control Status Register for Endpoint 0 in Host Mode (HOST_CSR0)

The control status register for endpoint 0 in host mode (HOST_CSR0) is shown in Figure 31-60 and described in Table 31-64.

Figure 31-60. Control Status Register for Endpoint 0 in Host Mode (HOST_CSR0)

15	12	11	10	9	8
Reserved		DISPING	DATATOGWREN	DATATOG	FLUSHFIFO
R-0		R/W-0	W-0	R/W-0	W-0

7	6	5	4	3	2	1	0
NAK_TIMEOUT	STATUSPKT	REQPKT	ERROR	SETUPPKT	RXSTALL	TXPKTRDY	RXPKTRDY
W-0	R/W-0	R/W-0	W-0	R/W-0	R/W-0	R/W-0	R/W-0

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

Table 31-64. Control Status Register for Endpoint 0 in Host Mode (HOST_CSR0) Field Descriptions

Bit	Field	Value	Description
15-12	Reserved	0	Reserved
11	DISPING	0-1	The CPU writes a 1 to the DSPING bit to instruct the core not to issue PING tokens in the data and status phases of a high-speed control transfer (for use with devices that do not respond to PING).
10	DATATOGWREN	0-1	Write 1 to this bit to enable the DATATOG bit to be written. This bit is automatically cleared once the new value is written to DATATOG.
9	DATATOG	0-1	When read, this bit indicates the current state of the EP0 data toggle. If DATATOGWREN is high, this bit can be written with the required setting of the data toggle. If DATATOGWREN is low, any value written to this bit is ignored.
8	FLUSHFIFO	0-1	Write 1 to this bit to flush the next packet to be transmitted/read from the Endpoint 0 FIFO. The FIFO pointer is reset and the TXPKTRDY/RXPKTRDY bit is cleared. Note: FLUSHFIFO has no effect unless TXPKTRDY/RXPKTRDY is set.
7	NAK_TIMEOUT	0-1	This bit will be set when Endpoint 0 is halted following the receipt of NAK responses for longer than the time set by the NAKLIMIT0 register. This bit should be cleared to allow the endpoint to continue.
6	STATUSPKT	0-1	Set this bit at the same time as the TXPKTRDY or REQPKT bit is set, to perform a status stage transaction. Setting this bit ensures that the data toggle is set so that a DATA1 packet is used for the Status Stage transaction.
5	REQPKT	0-1	Set this bit to request an IN transaction. It is cleared when RXPKTRDY is set.
4	ERROR	0-1	This bit will be set when three attempts have been made to perform a transaction with no response from the peripheral. You should clear this bit. An interrupt is generated when this bit is set.
3	SETUPPKT	0-1	Set this bit, at the same time as the TXPKTRDY bit is set, to send a SETUP token instead of an OUT token for the transaction.
2	RXSTALL	0-1	This bit is set when a STALL handshake is received. You should clear this bit.
1	TXPKTRDY	0-1	Set this bit after loading a data packet into the FIFO. It is cleared automatically when the data packet has been transmitted. An interrupt is generated (if enabled) when the bit is cleared.
0	RXPKTRDY	0-1	This bit is set when a data packet has been received. An interrupt is generated when this bit is set. Clear this bit by setting the SERV_RXPKTRDY bit.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.35 Control Status Register for Peripheral Transmit Endpoint (PERI_TXCSR)

The control status register for peripheral transmit endpoint (PERI_TXCSR) is shown in Figure 31-61 and described in Table 31-65.

Figure 31-61. Control Status Register for Peripheral Transmit Endpoint (PERI_TXCSR)

15	14	13	12	11	10	9 7
AUTOSET	ISO	MODE	DMAEN	FRCDATATOG	DMAMODE	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

6	5	4	3	2	1	0
CLRDATATOG	SENTSTALL	SENDSTALL	FLUSHFIFO	UNDERRUN	FIFONOTEMPTY	TXPKTRDY
W-0	R/W-0	R/W-0	W-0	R/W-0	R/W-0	R/W-0

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

Table 31-65. Control Status Register for Peripheral Transmit Endpoint (PERI_TXCSR) Field Descriptions

Bit	Field	Value	Description
15	AUTOSET	0	DMA Mode: The CPU needs to set the AUTOSET bit prior to enabling the Tx DMA.
		1	CPU Mode: If the CPU sets the AUTOSET bit, the TXPKTRDY bit will be automatically set when data of the maximum packet size (value in TXMAXP) is loaded into the Tx FIFO. If a packet of less than the maximum packet size is loaded, then the TXPKTRDY bit will have to be set manually.
14	ISO	0-1	Set this bit to enable the Tx endpoint for Isochronous transfers, and clear it to enable the Tx endpoint for Bulk or Interrupt transfers.
13	MODE	0-1	Set this bit to enable the endpoint direction as Tx, and clear the bit to enable it as Rx. Note: This bit has any effect only where the same endpoint FIFO is used for both Transmit and Receive transactions.
12	DMAEN	0-1	Set this bit to enable the DMA request for the Tx endpoint.
11	FRCDATATOG	0-1	Set this bit to force the endpoint data toggle to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received. This can be used by Interrupt Tx endpoints that are used to communicate rate feedback for Isochronous endpoints.
10	DMAMODE	0-1	This bit should always be set to 1 when the DMA is enabled.
9-7	Reserved	0	Reserved
6	CLRDATATOG	0-1	Write a 1 to this bit to reset the endpoint data toggle to 0.
5	SENTSTALL	0-1	This bit is set automatically when a STALL handshake is transmitted. The FIFO is flushed and the TXPKTRDY bit is cleared. You should clear this bit.
4	SENDSTALL	0-1	Write a 1 to this bit to issue a STALL handshake to an IN token. Clear this bit to terminate the stall condition. Note: This bit has no effect where the endpoint is being used for Isochronous transfers.
3	FLUSHFIFO	0-1	Write a 1 to this bit to flush the next packet to be transmitted from the endpoint Tx FIFO. The FIFO pointer is reset and the TXPKTRDY bit is cleared. Note: FlushFIFO has no effect unless the TXPKTRDY bit is set. Also note that, if the FIFO is double-buffered, FlushFIFO may need to be set twice to completely clear the FIFO.
2	UNDERRUN	0-1	This bit is set automatically if an IN token is received when TXPKTRDY is not set. You should clear this bit.
1	FIFONOTEMPTY	0-1	This bit is set when there is at least 1 packet in the Tx FIFO. You should clear this bit.
0	TXPKTRDY	0-1	Set this bit after loading a data packet into the FIFO. It is cleared automatically when a data packet has been transmitted. An interrupt is generated (if enabled) when the bit is cleared.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.36 Control Status Register for Host Transmit Endpoint (HOST_TXCSR)

The control status register for host transmit endpoint (HOST_TXCSR) is shown in Figure 31-62 and described in Table 31-66.

Figure 31-62. Control Status Register for Host Transmit Endpoint (HOST_TXCSR)

15	14	13	12	11	10	9	8
AUTOSET	Reserved	MODE	DMAEN	FRCDATATOG	DMAMODE	DATATOGWREN	DATATOG
R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	W-0	R/W-0
7	6	5	4	3	2	1	0
NAK_TIMEOUT	CLRDATATOG	RXSTALL	SETUPPKT	FLUSHFIFO	ERROR	FIFONOTEMPTY	TXPKTRDY
R/W-0	W-0	R/W-0	R/W-0	W-0	R/W-0	R/W-0	R/W-0

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

Table 31-66. Control Status Register for Host Transmit Endpoint (HOST_TXCSR) Field Descriptions

Bit	Field	Value	Description
15	AUTOSET	0	DMA Mode: The CPU needs to set the AUTOSET bit prior to enabling the Tx DMA.
		1	CPU Mode: If the CPU sets the AUTOSET bit, the TXPKTRDY bit will be automatically set when data of the maximum packet size (value in TXMAXP) is loaded into the Tx FIFO. If a packet of less than the maximum packet size is loaded, then the TXPKTRDY bit will have to be set manually.
14	Reserved	0	Reserved
13	MODE	0-1	Set this bit to enable the endpoint direction as Tx, and clear the bit to enable it as Rx. Note: This bit has any effect only where the same endpoint FIFO is used for both Transmit and Receive transactions.
12	DMAEN	0-1	Set this bit to enable the DMA request for the Tx endpoint.
11	FRCDATATOG	0-1	Set this bit to force the endpoint data toggle to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received. This can be used by Interrupt Tx endpoints that are used to communicate rate feedback for Isochronous endpoints.
10	DMAMODE	0-1	This bit should always be set to 1 when the DMA is enabled.
9	DATATOGWREN	0-1	Write 1 to this bit to enable the DATATOG bit to be written. This bit is automatically cleared once the new value is written to DATATOG.
8	DATATOG	0-1	When read, this bit indicates the current state of the Tx EP data toggle. If DATATOGWREN is high, this bit can be written with the required setting of the data toggle. If DATATOGWREN is low, any value written to this bit is ignored.
7	NAK_TIMEOUT	0-1	This bit will be set when the Tx endpoint is halted following the receipt of NAK responses for longer than the time set as the NAKLIMIT by the TXINTERVAL register. It should be cleared to allow the endpoint to continue. Note: This is valid only for Bulk endpoints.
6	CLRDATATOG	0-1	Write a 1 to this bit to reset the endpoint data toggle to 0.
5	RXSTALL	0-1	This bit is set when a STALL handshake is received. The FIFO is flushed and the TXPKTRDY bit is cleared (see below). You should clear this bit.
4	SETUPPKT	0-1	Set this bit at the same time as TXPKTRDY is set, to send a SETUP token instead of an OUT token for the transaction. Note: Setting this bit also clears the DATATOG bit.
3	FLUSHFIFO	0-1	Write a 1 to this bit to flush the next packet to be transmitted from the endpoint Tx FIFO. The FIFO pointer is reset and the TXPKTRDY bit (below) is cleared. Note: FlushFIFO has no effect unless the TXPKTRDY bit is set. Also note that, if the FIFO is double-buffered, FLUSHFIFO may need to be set twice to completely clear the FIFO.
2	ERROR	0-1	The USB controller sets this bit when 3 attempts have been made to send a packet and no handshake packet has been received. You should clear this bit. An interrupt is generated when the bit is set. This is valid only when the endpoint is operating in Bulk or Interrupt mode.
1	FIFONOTEMPTY	0-1	The USB controller sets this bit when there is at least 1 packet in the Tx FIFO.
0	TXPKTRDY	0-1	Set this bit after loading a data packet into the FIFO. It is cleared automatically when a data packet has been transmitted. An interrupt is generated (if enabled) when the bit is cleared.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.37 Maximum Packet Size for Peripheral Host Receive Endpoint (RXMAXP)

The maximum packet size for peripheral host receive endpoint (RXMAXP) is shown in Figure 31-63 and described in Table 31-67.

Figure 31-63. Maximum Packet Size for Peripheral Host Receive Endpoint (RXMAXP)

15	11	10	0
Reserved		MAXPAYLOAD	
R-0		R/W-0	

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

Table 31-67. Maximum Packet Size for Peripheral Host Receive Endpoint (RXMAXP) Field Descriptions

Bit	Field	Value	Description
15-11	Reserved	0	Reserved
10-0	MAXPAYLOAD	0-400h	Defines the maximum amount of data that can be transferred through the selected Receive endpoint in a single frame/microframe (high-speed transfers). 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 and high-speed operations. The value written to this register should match the wMaxPacketSize field of the Standard Endpoint Descriptor for the associated endpoint. A mismatch could cause unexpected results.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.38 Control Status Register for Peripheral Receive Endpoint (PERI_RXCSR)

The control status register for peripheral receive endpoint (PERI_RXCSR) is shown in Figure 31-64 and described in Table 31-68.

Figure 31-64. Control Status Register for Peripheral Receive Endpoint (PERI_RXCSR)

15	14	13	12	11	10		8
AUTOCLEAR	ISO	DMAEN	DISNYET	DMAMODE	Reserved		
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0		
7	6	5	4	3	2	1	0
CLRDATATOG	SENTSTALL	SENDSTALL	FLUSHFIFO	DATAERROR	OVERRUN	FIFOFULL	RXPKTRDY
W-0	R/W-0	R/W-0	W-0	R-0	R/W-0	R-0	R/W-0

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

Table 31-68. Control Status Register for Peripheral Receive Endpoint (PERI_RXCSR) Field Descriptions

Bit	Field	Value	Description
15	AUTOCLEAR	0	DMA Mode: The CPU sets the AUTOCLEAR bit prior to enabling the Rx DMA.
		1	CPU Mode: If the CPU sets the AUTOCLEAR bit, then the RXPKTRDY bit will be automatically cleared when a packet of RXMAXP bytes has been unloaded from the Receive FIFO. When packets of less than the maximum packet size are unloaded, RXPKTRDY will have to be cleared manually.
14	ISO	0-1	Set this bit to enable the Receive endpoint for Isochronous transfers, and clear it to enable the Receive endpoint for Bulk/Interrupt transfers.
13	DMAEN	0-1	Set this bit to enable the DMA request for the Receive endpoints.
12	DISNYET	0	DISNYET: Applies only for Bulk/Interrupt Transactions: The CPU sets this bit to disable the sending of NYET handshakes. When set, all successfully received Rx packets are ACK'd including at the point at which the FIFO becomes full. Note: This bit only has any effect in high-speed mode, in which mode it should be set for all Interrupt endpoints.
		1	PID_ERROR: Applies only for ISO Transactions: The core sets this bit to indicate a PID error in the received packet.
11	DMAMODE	0-1	Always clear this bit to 0.
10-8	Reserved	0	Reserved
7	CLRDATATOG	0-1	Write a 1 to this bit to reset the endpoint data toggle to 0.
6	SENTSTALL	0-1	This bit is set when a STALL handshake is transmitted. The FIFO is flushed and the TXPKTRDY bit is cleared. You should clear this bit.
5	SENDSTALL	0-1	Write a 1 to this bit to issue a STALL handshake. Clear this bit to terminate the stall condition. Note: This bit has no effect where the endpoint is being used for Isochronous transfers.
4	FLUSHFIFO	0-1	Write 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 RXPKTRDY bit is cleared. Note: FLUSHFIFO has no effect unless RXPKTRDY is set. Also note that, if the FIFO is double-buffered, FLUSHFIFO may need to be set twice to completely clear the FIFO.
3	DATAERROR	0-1	This bit is set when RXPKTRDY is set if the data packet has a CRC or bit-stuff error. It is cleared when RXPKTRDY is cleared. Note: This bit is only valid when the endpoint is operating in ISO mode. In Bulk mode, it always returns zero.
2	OVERRUN	0-1	This bit is set if an OUT packet cannot be loaded into the Receive FIFO. You should clear this bit. Note: This bit is only valid when the endpoint is operating in ISO mode. In Bulk mode, it always returns zero.
1	FIFOFULL	0-1	This bit is set when no more packets can be loaded into the Receive FIFO.
0	RXPKTRDY	0-1	This bit is set when a data packet has been received. You should clear this bit when the packet has been unloaded from the Receive FIFO. An interrupt is generated when the bit is set.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.39 Control Status Register for Host Receive Endpoint (HOST_RXCSR)

The control status register for host receive endpoint (HOST_RXCSR) is shown in Figure 31-65 and described in Table 31-69.

Figure 31-65. Control Status Register for Host Receive Endpoint (HOST_RXCSR)

15	14	13	12	11	10	9	8
AUTOCLEAR	AUTOREQ	DMAEN	DISNYET	DMAMODE	DATATOGWREN	DATATOG	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	W-0	R/W-0	R-0
7	6	5	4	3	2	1	0
CLRDATATOG	RXSTALL	REQPKT	FLUSHFIFO	DATAERR_ NAKTIMEOUT	ERROR	FIFOFULL	RXPKTRDY
W-0	R/W-0	R/W-0	W-0	R-0	R/W-0	R-0	R/W-0

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

Table 31-69. Control Status Register for Host Receive Endpoint (HOST_RXCSR) Field Descriptions

Bit	Field	Value	Description
15	AUTOCLEAR	0	DMA Mode: The CPU sets the AUTOCLEAR bit prior to enabling the Rx DMA.
		1	CPU Mode: If the CPU sets the AUTOCLEAR bit, then the RXPKTRDY bit will be automatically cleared when a packet of RXMAXP bytes has been unloaded from the Receive FIFO. When packets of less than the maximum packet size are unloaded, RXPKTRDY will have to be cleared manually.
14	AUTOREQ	1	If the CPU sets the AUTOREQ bit, then the REQPKT bit will be automatically set when the RXPKTRDY bit is cleared. Note: This bit is automatically cleared when a short packet is received.
13	DMAEN	0-1	Set this bit to enable the DMA request for the Receive endpoints.
12	DISNYET	0-1	Set this bit to disable the sending of NYET handshakes. When set, all successfully received Receive packets are ACKED including at the point at which the FIFO becomes full. Note: This bit only has any effect in high-speed mode, in which mode it should be set for all Interrupt endpoints.
11	DMAMODE	0-1	Always clear this bit to 0.
10	DATATOGWREN	0-1	Write 1 to this bit to enable the DATATOG bit to be written. This bit is automatically cleared once the new value is written to DATATOG.
9	DATATOG	0-1	When read, this bit indicates the current state of the Receive EP data toggle. If DATATOGWREN is high, this bit can be written with the required setting of the data toggle. If DATATOGWREN is low, any value written to this bit is ignored.
8	Reserved	0	Reserved
7	CLRDATATOG	0-1	Write a 1 to this bit to reset the endpoint data toggle to 0.
6	RXSTALL	0-1	When a STALL handshake is received, this bit is set and an interrupt is generated. You should clear this bit.
5	REQPKT	0-1	Write a 1 to this bit to request an IN transaction. It is cleared when RXPKTRDY is set.
4	FLUSHFIFO	0-1	Write 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 RXPKTRDY bit is cleared. Note: FLUSHFIFO has no effect unless RXPKTRDY is set. Also note that, if the FIFO is double-buffered, FLUSHFIFO may need to be set twice to completely clear the FIFO.
3	DATAERR_NAKTIMEOUT	0-1	When operating in ISO mode, this bit is set when RXPKTRDY is set if the data packet has a CRC or bit-stuff error and cleared when RXPKTRDY is cleared. In Bulk mode, this bit will be set when the Receive endpoint is halted following the receipt of NAK responses for longer than the time set as the NAK Limit by the RXINTERVAL register. You should clear this bit to allow the endpoint to continue.
2	ERROR	0-1	The USB controller sets this bit when 3 attempts have been made to receive a packet and no data packet has been received. You should clear this bit. An interrupt is generated when the bit is set. Note: This bit is only valid when the transmit endpoint is operating in Bulk or Interrupt mode. In ISO mode, it always returns zero.
1	FIFOFULL	0-1	This bit is set when no more packets can be loaded into the Receive FIFO.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 31-69. Control Status Register for Host Receive Endpoint (HOST_RXCSR) Field Descriptions (continued)

Bit	Field	Value	Description
0	RXPKTRDY	0-1	This bit is set when a data packet has been received. You should clear this bit when the packet has been unloaded from the Receive FIFO. An interrupt is generated when the bit is set.

31.4.40 Count 0 Register (COUNT0)

The count 0 register (COUNT0) is shown in Figure 31-66 and described in Table 31-70.

Figure 31-66. Count 0 Register (COUNT0)

15	7	6	0
Reserved		EP0RXCOUNT	
R-0		R-0	

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

Table 31-70. Count 0 Register (COUNT0) Field Descriptions

Bit	Field	Value	Description
15-7	Reserved	0	Reserved
6-0	EP0RXCOUNT	0-7Fh	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 RXPKTRDY of PERI_CSR0 or HOST_CSR0 is set.

31.4.41 Receive Count Register (RXCOUNT)

The receive count register (RXCOUNT) is shown in Figure 31-67 and described in Table 31-71.

Figure 31-67. Receive Count Register (RXCOUNT)

15	13	12	0
Reserved		EPRXCOUNT	
R-0		R-0	

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

Table 31-71. Receive Count Register (RXCOUNT) Field Descriptions

Bit	Field	Value	Description
15-13	Reserved	0	Reserved
12-0	EPRXCOUNT	0-1FFFh	Holds the number of received data bytes in the packet in the Receive FIFO. The value returned changes as the contents of the FIFO change and is only valid while RXPKTRDY of PERI_RXCSR or HOST_RXCSR is set.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.42 Type Register (Host mode only) (HOST_TYPE0)

The type register (Host mode only) (HOST_TYPE0) is shown in Figure 31-68 and described in Table 31-72.

Figure 31-68. Type Register (Host mode only) (HOST_TYPE0)

7	6	5	0
SPEED		Reserved	
R/W-0		R-0	

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

Table 31-72. Type Register (Host mode only) (HOST_TYPE0) Field Descriptions

Bit	Field	Value	Description
7-6	SPEED	0-3h	Operating Speed of Target Device
		0	Illegal
		1h	High
		2h	Full
		3h	Low
5-0	Reserved	0	Reserved

31.4.43 Transmit Type Register (Host mode only) (HOST_TXTYPE)

The transmit type register (Host mode only) (HOST_TXTYPE) is shown in Figure 31-69 and described in Table 31-73.

Figure 31-69. Transmit Type Register (Host mode only) (HOST_TXTYPE)

7	6	5	4	3	0
SPEED		PROT		TENDPN	
R/W-0		R/W-0		R/W-0	

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

Table 31-73. Transmit Type Register (Host mode only) (HOST_TXTYPE) Field Descriptions

Bit	Field	Value	Description
7-6	SPEED	0-3h	Operating Speed of Target Device
		0	Illegal
		1h	High
		2h	Full
		3h	Low
5-4	PROT	0-3h	Set this to select the required protocol for the transmit endpoint
		0	Control
		1h	Isochronous
		2h	Bulk
		3h	Interrupt
3-0	TENDPN	0-Fh	Set this value to the endpoint number contained in the transmit endpoint descriptor returned to the USB controller during device enumeration.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.44 NAKLimit0 Register (Host mode only) (HOST_NAKLIMIT0)

The NAKlimit0 register (Host mode only) (HOST_NAKLIMIT0) is shown in Figure 31-70 and described in Table 31-74.

Figure 31-70. NAKLimit0 Register (Host mode only) (HOST_NAKLIMIT0)

7	5	4	0
Reserved		EP0NAKLIMIT	
R-0		R/W-0	

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

Table 31-74. NAKLimit0 Register (Host mode only) (HOST_NAKLIMIT0) Field Descriptions

Bit	Field	Value	Description
7-5	Reserved	0	Reserved
4-0	EP0NAKLIMIT	0-1Fh	Sets the number of frames/microframes (high-speed transfers) after which Endpoint 0 should time out on receiving a stream of NAK responses. The number of frames/microframes selected is $2^{(-1)}$ (where m is the value set in the register, valid values 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 will be halted. Note: A value of 0 or 1 disables the NAK timeout function.

31.4.45 Transmit Interval Register (Host mode only) (HOST_TXINTERVAL)

The transmit interval register (Host mode only) (HOST_TXINTERVAL) is shown in Figure 31-71 and described in Table 31-75.

Figure 31-71. Transmit Interval Register (Host mode only) (HOST_TXINTERVAL)

7	0
POLINTVL_NAKLIMIT	
R/W-0	

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

Table 31-75. Transmit Interval Register (Host mode only) (HOST_TXINTERVAL) Field Descriptions

Bit	Field	Value	Description
7-0	POLINTVL_NAKLIMIT	0-FFh	For Interrupt and Isochronous transfers, defines the polling interval for the currently-selected transmit endpoint. For Bulk endpoints, sets the number of frames/microframes after which the endpoint should timeout on receiving a stream of NAK responses. There is a transmit interval register for each configured transmit endpoint (except Endpoint 0). In each case, the value that is set defines a number of frames/microframes (High-Speed transfers), as follows:

Transfer Type	Speed	Valid values (m)	Interpretation
Interrupt	Low Speed or Full Speed	1-255	Polling interval is m frames
	High Speed	1-16	Polling interval is $2^{(-1)}$ microframes
Isochronous	Full Speed or High Speed	1-16	Polling interval is $2^{(-1)}$ frames/microframes
Bulk	Full Speed or High Speed	2-16	NAK Limit is $2^{(-1)}$ frames/microframes

Note: A value of 0 or 1 disables the NAK timeout function.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.46 Receive Type Register (Host mode only) (HOST_RXTYPE)

The receive type register (Host mode only) (HOST_RXTYPE) is shown in Figure 31-72 and described in Table 31-76.

Figure 31-72. Receive Type Register (Host mode only) (HOST_RXTYPE)

7	6	5	4	3	0
SPEED		PROT		RENDPN	
R/W-0		R/W-0		R/W-0	

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

Table 31-76. Receive Type Register (Host mode only) (HOST_RXTYPE) Field Descriptions

Bit	Field	Value	Description
7-6	SPEED	0-3h	Operating Speed of Target Device
		0	Illegal
		1h	High
		2h	Full
		3h	Low
5-4	PROT	0-3h	Set this to select the required protocol for the transmit endpoint
		0	Control
		1h	Isochronous
		2h	Bulk
		3h	Interrupt
3-0	RENDPN	0-Fh	Set this value to the endpoint number contained in the Receive endpoint descriptor returned to the USB controller during device enumeration

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.47 Receive Interval Register (Host mode only) (HOST_RXINTERVAL)

The receive interval register (Host mode only) (HOST_RXINTERVAL) is shown in Figure 31-73 and described in Table 31-77.

Figure 31-73. Receive Interval Register (Host mode only) (HOST_RXINTERVAL)

7	0
POLINTVL_NAKLIMIT	
R/W-0	

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

Table 31-77. Receive Interval Register (Host mode only) (HOST_RXINTERVAL) Field Descriptions

Bit	Field	Value	Description
7-0	POLINTVL_NAKLIMIT	0-FFh	For Interrupt and Isochronous transfers, defines the polling interval for the currently-selected transmit endpoint. For Bulk endpoints, sets the number of frames/microframes after which the endpoint should timeout on receiving a stream of NAK responses. There is a transmit interval register for each configured transmit endpoint (except Endpoint 0). In each case, the value that is set defines a number of frames/microframes (High-Speed transfers), as follows:

Transfer Type	Speed	Valid values (m)	Interpretation
Interrupt	Low Speed or Full Speed	1-255	Polling interval is m frames
	High Speed	1-16	Polling interval is $2^{(m-1)}$ microframes
Isochronous	Full Speed or High Speed	1-16	Polling interval is $2^{(m-1)}$ frames/microframes
Bulk	Full Speed or High Speed	2-16	NAK Limit is $2^{(m-1)}$ frames/microframes

Note: A value of 0 or 1 disables the NAK timeout function.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.48 Configuration Data Register (CONFIGDATA)

The configuration data register (CONFIGDATA) is shown in Figure 31-74 and described in Table 31-78.

Figure 31-74. Configuration Data Register (CONFIGDATA)

7	6	5	4	3	2	1	0
MPRXE	MPTXE	BIGENDIAN	HBRXE	HBTXE	DYNFIFO	SOFTCONE	UTMIDATAWIDTH
R-0	R-0	R-0	R-0	R-0	R-1	R-1	R-0

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

Table 31-78. Configuration Data Register (CONFIGDATA) Field Descriptions

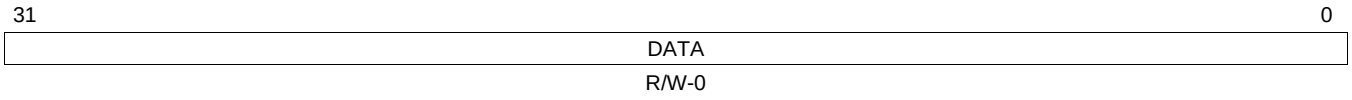
Bit	Field	Value	Description
7	MPRXE		Indicates automatic amalgamation of bulk packets.
		0	Automatic amalgamation of bulk packets is not selected.
		1	Automatic amalgamation of bulk packets is selected.
6	MPTXE		Indicates automatic splitting of bulk packets.
		0	Automatic splitting of bulk packets is not selected.
		1	Automatic splitting of bulk packets is selected.
5	BIGENDIAN		Indicates endian ordering.
		0	Little-endian ordering is selected.
		1	Big-endian ordering is selected.
4	HBRXE		Indicates high-bandwidth Rx ISO endpoint support.
		0	High-bandwidth Rx ISO endpoint support is not selected.
		1	High-bandwidth Rx ISO endpoint support is selected.
3	HBTXE		Indicates high-bandwidth Tx ISO endpoint support.
		0	High-bandwidth Tx ISO endpoint support is not selected.
		1	High-bandwidth Tx ISO endpoint support is selected.
2	DYNFIFO		Indicates dynamic FIFO sizing.
		0	Dynamic FIFO sizing option is not selected.
		1	Dynamic FIFO sizing option is selected.
1	SOFTCONE		Indicates soft connect/disconnect.
		0	Soft connect/disconnect option is not selected
		1	Soft connect/disconnect option is selected
0	UTMIDATAWIDTH		Indicates selected UTMI data width.
		0	8 bits
		1	16 bits

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.49 Transmit and Receive FIFO Register for Endpoint 0 (FIFO0)

The transmit and receive FIFO register for endpoint 0 (FIFO0) is shown in Figure 31-75 and described in Table 31-79.

Figure 31-75. Transmit and Receive FIFO Register for Endpoint 0 (FIFO0)



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

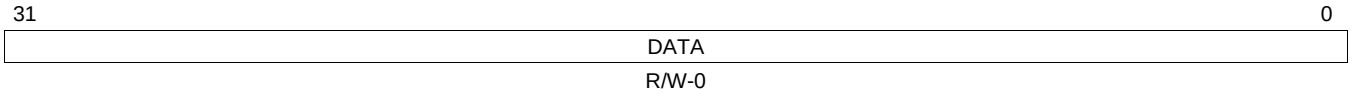
Table 31-79. Transmit and Receive FIFO Register for Endpoint 0 (FIFO0) Field Descriptions

Bit	Field	Value	Description
31-0	DATA	0-FFFF FFFFh	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.

31.4.50 Transmit and Receive FIFO Register for Endpoint 1 (FIFO1)

The transmit and receive FIFO register for endpoint 1 (FIFO1) is shown in Figure 31-76 and described in Table 31-80.

Figure 31-76. Transmit and Receive FIFO Register for Endpoint 1 (FIFO1)



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

Table 31-80. Transmit and Receive FIFO Register for Endpoint 1 (FIFO1) Field Descriptions

Bit	Field	Value	Description
31-0	DATA	0-FFFF FFFF	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.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.51 Transmit and Receive FIFO Register for Endpoint 2 (FIFO2)

The transmit and receive FIFO register for endpoint 2 (FIFO2) is shown in Figure 31-77 and described in Table 31-81.

Figure 31-77. Transmit and Receive FIFO Register for Endpoint 2 (FIFO2)

31	0
DATA	
R/W-0	

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

Table 31-81. Transmit and Receive FIFO Register for Endpoint 2 (FIFO2) Field Descriptions

Bit	Field	Value	Description
31-0	DATA	0-FFFF FFFFh	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.

31.4.52 Transmit and Receive FIFO Register for Endpoint 3 (FIFO3)

The transmit and receive FIFO register for endpoint 3 (FIFO3) is shown in Figure 31-78 and described in Table 31-82.

Figure 31-78. Transmit and Receive FIFO Register for Endpoint 3 (FIFO3)

31	0
DATA	
R/W-0	

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

Table 31-82. Transmit and Receive FIFO Register for Endpoint 3 (FIFO3) Field Descriptions

Bit	Field	Value	Description
31-0	DATA	0-FFFF FFFFh	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.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.53 Transmit and Receive FIFO Register for Endpoint 4 (FIFO4)

The transmit and receive FIFO register for endpoint 4 (FIFO4) is shown in Figure 31-79 and described in Table 31-83.

Figure 31-79. Transmit and Receive FIFO Register for Endpoint 4 (FIFO4)

31	0
DATA	
R/W-0	

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

Table 31-83. Transmit and Receive FIFO Register for Endpoint 4 (FIFO4) Field Descriptions

Bit	Field	Value	Description
31-0	DATA	0-FFFF FFFFh	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.

31.4.54 Device Control Register (DEVCTL)

The device control register (DEVCTL) is shown in Figure 31-80 and described in Table 31-84.

Figure 31-80. Device Control Register (DEVCTL)

7	6	5	4	3	2	1	0
BDEVICE	FSDEV	LSDEV	VBUS		HOSTMODE	HOSTREQ	SESSION
R-0	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 31-84. Device Control Register (DEVCTL) Field Descriptions

Bit	Field	Value	Description
7	BDEVICE		This read-only bit indicates whether the USB controller is operating as the 'A' device or the 'B' device.
		0	A device
		1	B device
			Only valid while a session is in progress.
6	FSDEV	0-1	This read-only bit is set when a full-speed or high-speed device has been detected being connected to the port (high-speed devices are distinguished from full-speed by checking for high-speed chirps when the device is reset). Only valid in Host mode.
5	LSDEV	0-1	This read-only bit is set when a low-speed device has been detected being connected to the port. Only valid in Host mode.
4-3	VBUS	0-3h	These read-only bits encode the current VBus level as follows:
		0	Below Session End
		1h	Above Session End, below AValid
		2h	Above AValid, below VBusValid
		3h	Above VBusValid
2	HOSTMODE	0-1	This read-only bit is set when the USB controller is acting as a Host.
1	HOSTREQ	0-1	When set, the USB controller will initiate the Host Negotiation when Suspend mode is entered. It is cleared when Host Negotiation is completed. ('B' device only)
0	SESSION	0-1	When operating as an 'A' device, you must set or clear this bit start or end a session. When operating as a 'B' device, this bit is set/cleared by the USB controller when a session starts/ends. You must also set this bit to initiate the Session Request Protocol (SRP). When the USB controller is in Suspend mode, you may clear the bit to perform a software disconnect. A special software routine is required to perform SRP.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.55 Transmit Endpoint FIFO Size (TXFIFOSZ)

Section 31.2.6 describes dynamically setting endpoint FIFO sizes. The option of dynamically setting endpoint FIFO sizes only applies to Endpoints 1-4. The Endpoint 0 FIFO has a fixed size (64 bytes) and a fixed location (start address 0). It is the responsibility of the firmware to ensure that all the Tx and Rx endpoints that are active in the current USB configuration have a block of RAM assigned exclusively to that endpoint. The RAM must be at least as large as the maximum packet size set for that endpoint.

The transmit endpoint FIFO size (TXFIFOSZ) is shown in Figure 31-81 and described in Table 31-85.

Figure 31-81. Transmit Endpoint FIFO Size (TXFIFOSZ)

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 31-85. Transmit Endpoint FIFO Size (TXFIFOSZ) Field Descriptions

Bit	Field	Value	Description
7-5	Reserved	0	Reserved
4	DPB		Double packet buffering enable
		0	Single packet buffering is supported
		1	Double packet buffering is enabled
3-0	SZ	0-Fh	Maximum packet size to be allowed (before any splitting within the FIFO of Bulk packets prior to transmission). If m = SZ, the FIFO size is calculated as $2^{(m+3)}$ for single packet buffering and $2^{(m+4)}$ for dual packet buffering.

31.4.56 Receive Endpoint FIFO Size (RXFIFOSZ)

Section 31.2.6 describes dynamically setting endpoint FIFO sizes. The option of dynamically setting endpoint FIFO sizes only applies to Endpoints 1-4. The Endpoint 0 FIFO has a fixed size (64 bytes) and a fixed location (start address 0). It is the responsibility of the firmware to ensure that all the Tx and Rx endpoints that are active in the current USB configuration have a block of RAM assigned exclusively to that endpoint. The RAM must be at least as large as the maximum packet size set for that endpoint.

The receive endpoint FIFO size (RXFIFOSZ) is shown in Figure 31-82 and described in Table 31-86.

Figure 31-82. Receive Endpoint FIFO Size (RXFIFOSZ)

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 31-86. Receive Endpoint FIFO Size (RXFIFOSZ) Field Descriptions

Bit	Field	Value	Description
7-5	Reserved	0	Reserved
4	DPB		Double packet buffering enable
		0	Single packet buffering is supported
		1	Double packet buffering is enabled
3-0	SZ	0-Fh	Maximum packet size to be allowed (before any splitting within the FIFO of Bulk packets prior to transmission). If m = SZ, the FIFO size is calculated as $2^{(m+3)}$ for single packet buffering and $2^{(m+4)}$ for dual packet buffering.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.57 Transmit Endpoint FIFO Address (TXFIFOADDR)

Section 31.2.6 describes dynamically setting endpoint FIFO sizes.

The transmit endpoint FIFO address (TXFIFOADDR) is shown in Figure 31-83 and described in Table 31-87.

Figure 31-83. Transmit Endpoint FIFO Address (TXFIFOADDR)

15	13	12	0
Reserved		ADDR	
R-0		R/W-0	

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

Table 31-87. Transmit Endpoint FIFO Address (TXFIFOADDR) Field Descriptions

Bit	Field	Value	Description
15-13	Reserved	0	Reserved
12-0	ADDR	0-1FFFh	Start Address of endpoint FIFO in units of 8 bytes If m = ADDR, then the start address is 8 × m

31.4.58 Receive Endpoint FIFO Address (RXFIFOADDR)

Section 31.2.6 describes dynamically setting endpoint FIFO sizes.

The receive endpoint FIFO address (RXFIFOADDR) is shown in Figure 31-84 and described in Table 31-88.

Figure 31-84. Receive Endpoint FIFO Address (RXFIFOADDR)

15	13	12	0
Reserved		ADDR	
R-0		R/W-0	

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

Table 31-88. Receive Endpoint FIFO Address (RXFIFOADDR) Field Descriptions

Bit	Field	Value	Description
15-13	Reserved	0	Reserved
12-0	ADDR	0-1FFFh	Start Address of endpoint FIFO in units of 8 bytes If m = ADDR, then the start address is 8 × m

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.59 Hardware Version Register (HWVERS)

The hardware version register (HWVERS) contains the RTL major and minor version numbers for the USB 2.0 OTG controller module. The RTL version number is REVMAJ.REVMIN. The HWVERS is shown in Figure 31-85 and described in Table 31-89.

Figure 31-85. Hardware Version Register (HWVERS)

15	14 10	9 0
RC	REVMAJ	REVMIN
R-0	R-0	R-0

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

Table 31-89. Hardware Version Register (HWVERS) Field Descriptions

Bit	Field	Value	Description
15	RC	0-1	Set to 1 if RTL is used from a Release Candidate, rather than from a full release of the core.
14-10	REVMAJ	0-1Fh	Major version of RTL. Range is 0-31.
9-0	REVMIN	0-3E7h	Minor version of RTL. Range is 0-999.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.60 Transmit Function Address (TXFUNCADDR)

The transmit function address (TXFUNCADDR) is shown in Figure 31-86 and described in Table 31-90.

Figure 31-86. Transmit Function Address (TXFUNCADDR)

7	6	0
Reserved	FUNCADDR	
R-0	R/W-0	

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

Table 31-90. Transmit Function Address (TXFUNCADDR) Field Descriptions

Bit	Field	Value	Description
7	Reserved	0	Reserved
6-0	FUNCADDR	0-7Fh	Address of target function

31.4.61 Transmit Hub Address (TXHUBADDR)

The transmit hub address (TXHUBADDR) is shown in Figure 31-87 and described in Table 31-91.

Figure 31-87. Transmit Hub Address (TXHUBADDR)

7	6	0
MULT_TRANS	HUBADDR	
R/W-0	R/W-0	

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

Table 31-91. Transmit Hub Address (TXHUBADDR) Field Descriptions

Bit	Field	Value	Description
7	MULT_TRANS	0-1	Set to 1 if hub has multiple transaction translators. Cleared to 0 if only single transaction translator is available.
6-0	HUBADDR	0-7Fh	Address of hub

31.4.62 Transmit Hub Port (TXHUBPORT)

The transmit hub port (TXHUBPORT) is shown in Figure 31-88 and described in Table 31-92.

Figure 31-88. Transmit Hub Port (TXHUBPORT)

7	6	0
Reserved	HUBPORT	
R-0	R/W-0	

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

Table 31-92. Transmit Hub Port (TXHUBPORT) Field Descriptions

Bit	Field	Value	Description
7	Reserved	0	Reserved
6-0	HUBPORT	0-7Fh	Port number of the hub

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.63 Receive Function Address (RXFUNCADDR)

The receive function address (RXFUNCADDR) is shown in Figure 31-89 and described in Table 31-93.

Figure 31-89. Receive Function Address (RXFUNCADDR)

7	6	0
Reserved	FUNCADDR	
R-0	R/W-0	

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

Table 31-93. Receive Function Address (RXFUNCADDR) Field Descriptions

Bit	Field	Value	Description
7	Reserved	0	Reserved
6-0	FUNCADDR	0-7Fh	Address of target function

31.4.64 Receive Hub Address (RXHUBADDR)

The receive hub address (RXHUBADDR) is shown in Figure 31-90 and described in Table 31-94.

Figure 31-90. Receive Hub Address (RXHUBADDR)

7	6	0
MULT_TRANS	HUBADDR	
R/W-0	R/W-0	

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

Table 31-94. Receive Hub Address (RXHUBADDR) Field Descriptions

Bit	Field	Value	Description
7	MULT_TRANS	0-1	Set to 1 if hub has multiple transaction translators. Cleared to 0 if only single transaction translator is available.
6-0	HUBADDR	0-7Fh	Address of hub

31.4.65 Receive Hub Port (RXHUBPORT)

The receive hub port (RXHUBPORT) is shown in Figure 31-91 and described in Table 31-95.

Figure 31-91. Receive Hub Port (RXHUBPORT)

7	6	0
Reserved	HUBPORT	
R-0	R/W-0	

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

Table 31-95. Receive Hub Port (RXHUBPORT) Field Descriptions

Bit	Field	Value	Description
7	Reserved	0	Reserved
6-0	HUBPORT	0-7Fh	Port number of hub

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.66 CDMA Revision Identification Register (DMAREVID)

The CDMA revision identification register (DMAREVID) contains the revision for the module. The DMAREVID is shown in Figure 31-92 and described in Table 31-96.

Figure 31-92. CDMA Revision Identification Register (DMAREVID)

31	0
REV	
R-0053 1900h	

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

Table 31-96. CDMA Revision Identification Register (DMAREVID) Field Descriptions

Bit	Field	Value	Description
31-0	REV	0053 1900h	Revision ID of the CPPI DMA (CDMA) module.

31.4.67 CDMA Teardown Free Descriptor Queue Control Register (TDFDQ)

The CDMA teardown free descriptor queue control register (TDFDQ) is used to inform the DMA of the location in memory or descriptor array which is to be used for signaling of a teardown complete for each transmit and receive channel. The CDMA teardown free descriptor queue control register (TDFDQ) is shown in Figure 31-93 and described in Table 31-97.

Figure 31-93. CDMA Teardown Free Descriptor Queue Control Register (TDFDQ)

31	16
Reserved	
R-0	

15	14	13	12	11	0
Reserved		TD_DESC_QMGR		TD_DESC_QNUM	
R-0		R/W-0		R/W-0	

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

Table 31-97. CDMA Teardown Free Descriptor Queue Control Register (TDFDQ) Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Reserved
13-12	TD_DESC_QMGR	0-3h	Controls which of the four queue managers the DMA accesses to allocate a channel teardown descriptor from the teardown descriptor queue.
11-0	TD_DESC_QNUM	0-FFFh	Controls which of the 2K queues in the indicated queue manager should be read to allocate the channel teardown descriptors.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.68 CDMA Emulation Control Register (DMAEMU)

The CDMA emulation controls the behavior of the DMA when the emususp input is asserted. The CDMA emulation control register (DMAEMU) is shown in Figure 31-94 and described in Table 31-98.

Figure 31-94. CDMA Emulation Control Register (DMAEMU)

31	2	1	0
Reserved		SOFT	FREE
R-0		R/W-0	R/W-0

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

Table 31-98. CDMA Emulation Control Register (DMAEMU) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	SOFT		Determines emulation mode functionality. When the FREE bit is cleared to 0, the SOFT bit selects the mode.
		0	Upon emulation suspend, operation is not affected.
		1	In response to an emulation suspend event, the logic halts after the current transaction is completed.
0	FREE		Free run emulation control. Determines emulation mode functionality. When the FREE bit is cleared to 0, the SOFT bit selects the mode.
		0	The SOFT bit selects the mode.
		1	Runs free regardless of the SOFT bit.

31.4.69 CDMA Transmit Channel n Global Configuration Registers (TXGCR[0]-TXGCR[3])

The transmit channel *n* configuration registers (TXGCR[*n*]) initialize the behavior of each of the transmit DMA channels. There are four configuration registers, one for each transmit DMA channels. The transmit channel *n* configuration registers (TXGCR[*n*]) are shown in Figure 31-95 and described in Table 31-99.

Figure 31-95. CDMA Transmit Channel *n* Global Configuration Registers (TXGCR[*n*])

31	30	29	16
TX_ENABLE	TX_TEARDOWN	Reserved	
R/W-0	R/W-0	R-0	

15	14	13	12	11	0
Reserved		TX_DEFAULT_QMGR		TX_DEFAULT_QNUM	
R-0		W-0		W-0	

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

Table 31-99. CDMA Transmit Channel *n* Global Configuration Registers (TXGCR[*n*]) Field Descriptions

Bit	Field	Value	Description
31	TX_ENABLE		Channel control. The TX_ENABLE field is cleared after a channel teardown is complete.
		0	Disables channel
		1	Enables channel
30	TX_TEARDOWN	0-1	Setting this bit requests the channel to be torn down. The TX_TEARDOWN field remains set after a channel teardown is complete.
29-14	Reserved	0	Reserved
13-12	TX_DEFAULT_QMGR	0-3h	Controls the default queue manager number that is used to queue teardown descriptors back to the host.
11-0	TX_DEFAULT_QNUM	0-FFFh	Controls the default queue number within the selected queue manager onto which teardown descriptors are queued back to the host. This is the Tx Completion Queue.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.70 CDMA Receive Channel n Global Configuration Registers (RXGCR[0]-RXGCR[3])

The receive channel *n* global configuration registers (RXGCR[*n*]) initialize the global (non-descriptor-type specific) behavior of each of the receive DMA channels. There are four configuration registers, one for each receive DMA channels. If the enable bit is being set, the receive channel *n* global configuration register should only be written after all of the other receive configuration registers have been initialized. The receive channel *n* global configuration registers (RXGCR[*n*]) are shown in Figure 31-96 are described in Table 31-100.

Figure 31-96. CDMA Receive Channel *n* Global Configuration Registers (RXGCR[*n*])

31	30	29–25	24	23–16
RX_ENABLE	RX_TEARDOWN	Reserved	RX_ERROR_HANDLING	RX_SOP_OFFSET
R/W-0	R/W-0	R-0	W-0	W-0

15–14	13–12	11–0
RX_DEFAULT_DESC_TYPE	RX_DEFAULT_RQ_QMGR	RX_DEFAULT_RQ_QNUM
R-0	W-0	W-0

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

Table 31-100. CDMA Receive Channel *n* Global Configuration Registers (RXGCR[*n*]) Field Descriptions

Bit	Field	Value	Description
31	RX_ENABLE		Channel control. Field is cleared after a channel teardown is complete.
		0	Disables channel
		1	Enables channel
30	RX_TEARDOWN	0-1	Indicates whether a receive operation is complete. Field should be cleared when a channel is initialized. Field is set after a channel teardown is complete.
29-25	Reserved	0	Reserved
24	RX_ERROR_HANDLING		Controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occur):
		0	Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to.
		1	Starvation errors result in subsequent retry of the descriptor allocation operation. In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface. This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.
23-16	RX_SOP_OFFSET	0–FFh	Specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload. This value must be less than the minimum size of a buffer in the system.
15-14	RX_DEFAULT_DESC_TYPE	0-3h	Indicates the default descriptor type to use. The actual descriptor type that is used for reception can be overridden by information provided in the CPPI FIFO data block.
		0	Reserved
		1h	Host
		2h-3h	Reserved
13-12	RX_DEFAULT_RQ_QMGR	0-3h	Indicates the default receive queue manager that this channel should use. The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.
11-0	RX_DEFAULT_RQ_QNUM	0-FFFh	Indicates the default receive queue that this channel should use. The actual receive queue that is used for reception can be overridden by information provided in the CPPI FIFO data block. This is the Rx Completion Queue.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.71 CDMA Receive Channel n Host Packet Configuration Registers A (RXHPCRA[0]-RXHPCRA[3])

The receive channel *n* host packet configuration registers A (RXHPCRA[*n*]) initialize the behavior of each of the receive DMA channels for reception of host type packets. There are four configuration A registers, one for each receive DMA channels. The receive channel *n* host packet configuration registers A (RXHPCRA[*n*]) are shown in Figure 31-97 and described in Table 31-101.

Figure 31-97. Receive Channel *n* Host Packet Configuration Registers A (RXHPCRA[*n*])

31	30	29	28	27	16
Reserved		RX_HOST_FDQ1_QMGR		RX_HOST_FDQ1_QNUM	
R-0		W-0		W-0	

15	14	13	12	11	0
Reserved		RX_HOST_FDQ0_QMGR		RX_HOST_FDQ0_QNUM	
R-0		W-0		W-0	

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

Table 31-101. Receive Channel *n* Host Packet Configuration Registers A (RXHPCRA[*n*]) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Reserved
29-28	RX_HOST_FDQ1_QMGR	0-3h	Specifies which buffer manager should be used for the second receive buffer in a host type packet.
27-16	RX_HOST_FDQ1_QNUM	0-FFFh	Specifies which free descriptor/buffer pool should be used for the second receive buffer in a host type packet. This is the Rx Submit Queue for the second Incoming Packet.
15-14	Reserved	0	Reserved
13-12	RX_HOST_FDQ0_QMGR	0-3h	Specifies which buffer manager should be used for the first receive buffer in a host type packet.
11-0	RX_HOST_FDQ0_QNUM	0-FFFh	Specifies which free descriptor/buffer pool should be used for the first receive buffer in a host type packet. This is the Rx Submit Queue for the first Incoming Packet.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.72 CDMA Receive Channel n Host Packet Configuration Registers B (RXHPCRB[0]-RXHPCRB[3])

The receive channel *n* host packet configuration registers B (RXHPCRB[*n*]) initialize the behavior of each of the receive DMA channels for reception of host type packets. There are four configuration B registers, one for each receive DMA channels. The receive channel *n* host packet configuration registers B (RXHPCRB[*n*]) are shown in Figure 31-98 and described in Table 31-102.

Figure 31-98. Receive Channel *n* Host Packet Configuration Registers B (RXHPCRB[*n*])

31 30	29 28	27 16
Reserved	RX_HOST_FDQ3_QMGR	RX_HOST_FDQ3_QNUM
R-0	W-0	W-0

15 14	13 12	11 0
Reserved	RX_HOST_FDQ2_QMGR	RX_HOST_FDQ2_QNUM
R-0	W-0	W-0

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

Table 31-102. Receive Channel *n* Host Packet Configuration Registers B (RXHPCRB[*n*]) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Reserved
29-28	RX_HOST_FDQ3_QMGR	0-3h	Specifies which buffer manager should be used for the fourth or later receive buffer in a host type packet.
27-16	RX_HOST_FDQ3_QNUM	0-FFFh	Specifies which free descriptor/buffer pool should be used for the fourth or later receive buffer in a host type packet. This is the Rx Submit Queue for the fourth and remaining Incoming Packet.
15-14	Reserved	0	Reserved
13-12	RX_HOST_FDQ2_QMGR	0-3h	Specifies which buffer manager should be used for the third receive buffer in a host type packet.
11-0	RX_HOST_FDQ2_QNUM	0-FFFh	Specifies which free descriptor/buffer pool should be used for the third receive buffer in a host type packet. This is the Rx Submit Queue for the third Incoming Packet.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.73 CDMA Scheduler Control Register (DMA_SCHED_CTRL)

The CDMA scheduler control register (DMA_SCHED_CTRL) enables the scheduler and indicates the last entry in the scheduler table. The CDMA scheduler control register (DMA_SCHED_CTRL) is shown in Figure 31-99 and described in Table 31-103.

Figure 31-99. CDMA Scheduler Control Register (DMA_SCHED_CTRL)

31	30	16
ENABLE	Reserved	
R/W-0	R-0	

15	8	7	0
Reserved		LAST_ENTRY	
R-0		R/W-0	

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

Table 31-103. CDMA Scheduler Control Register (DMA_SCHED_CTRL) Field Descriptions

Bit	Field	Value	Description
31	ENABLE		This is the enable bit for the scheduler and is encoded as follows:
		0	Scheduler is disabled and will no longer fetch entries from the scheduler table or pass credits to the DMA controller
		1	Scheduler is enabled. This bit should only be set after the table has been initialized.
30-8	Reserved	0	Reserved
7-0	LAST_ENTRY	0-FFh	Indicates the last valid entry in the scheduler table. There are 64 words in the table and there are 4 entries in each word. The table can be programmed with any integer number of entries from 1 to 256. The corresponding encoding for this field is as follows:
		0	1 entry
		1h	2 entries
		2h-FFh	3 entries to 256 entries

31.4.74 CDMA Scheduler Table Word n Registers (WORD[0]-WORD[63])

The CDMA scheduler table word *n* registers (WORD[*n*]) has 4 entries (ENTRY[0] to ENTRY[3]) that provide information about the scheduler. The CDMA scheduler table word *n* registers (WORD[*n*]) are shown in Figure 31-100 and described in Table 31-104.

Figure 31-100. CDMA Scheduler Table Word *n* Registers (WORD[*n*])

31	30	28	27	24	23	22	20	19	16
ENTRY3_RXTX	Reserved		ENTRY3_CHANNEL		ENTRY2_RXTX	Reserved		ENTRY2_CHANNEL	
W-0	R-0		W-0		W-0	R-0		W-0	

15	14	12	11	8	7	6	4	3	0
ENTRY1_RXTX	Reserved		ENTRY1_CHANNEL		ENTRY0_RXTX	Reserved		ENTRY0_CHANNEL	
W-0	R-0		W-0		W-0	R-0		W-0	

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

Table 31-104. CDMA Scheduler Table Word *n* Registers (WORD[*n*]) Field Descriptions

Bit	Field	Value	Description
31	ENTRY3_RXTX		This entry is for a transmit or a receive channel.
		0	Transmit channel
		1	Receive channel
30-28	Reserved	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

Table 31-104. CDMA Scheduler Table Word *n* Registers (WORD[*n*]) Field Descriptions (continued)

Bit	Field	Value	Description
27-24	ENTRY3_CHANNEL	0-Fh	Indicates the channel number that is to be given an opportunity to transfer data. If this is a transmit entry, the DMA will be presented with a scheduling credit for that exact transmit channel. If this is a receive entry, the DMA will be presented with a scheduling credit for the receive FIFO that is associated with this channel. For receive FIFOs which carry traffic for more than one receive DMA channel, the exact channel number that is given in the receive credit will actually be the channel number which is currently on the head element of that Rx FIFO, which is not necessarily the channel number given in the scheduler table entry.
23	ENTRY2_RXTX		This entry is for a transmit or a receive channel.
		0	Transmit channel
		1	Receive channel
22-20	Reserved	0	Reserved
19-16	ENTRY2_CHANNEL	0-Fh	Indicates the channel number that is to be given an opportunity to transfer data. If this is a transmit entry, the DMA will be presented with a scheduling credit for that exact transmit channel. If this is a receive entry, the DMA will be presented with a scheduling credit for the receive FIFO that is associated with this channel. For receive FIFOs which carry traffic for more than one receive DMA channel, the exact channel number that is given in the receive credit will actually be the channel number which is currently on the head element of that Rx FIFO, which is not necessarily the channel number given in the scheduler table entry.
15	ENTRY1_RXTX		This entry is for a transmit or a receive channel.
		0	Transmit channel
		1	Receive channel
14-12	Reserved	0	Reserved
11-8	ENTRY1_CHANNEL	0-Fh	Indicates the channel number that is to be given an opportunity to transfer data. If this is a transmit entry, the DMA will be presented with a scheduling credit for that exact transmit channel. If this is a receive entry, the DMA will be presented with a scheduling credit for the receive FIFO that is associated with this channel. For receive FIFOs which carry traffic for more than one receive DMA channel, the exact channel number that is given in the receive credit will actually be the channel number which is currently on the head element of that Rx FIFO, which is not necessarily the channel number given in the scheduler table entry.
7	ENTRY0_RXTX		This entry is for a transmit or a receive channel.
		0	Transmit channel
		1	Receive channel
6-4	Reserved	0	Reserved
3-0	ENTRY0_CHANNEL	0-Fh	Indicates the channel number that is to be given an opportunity to transfer data. If this is a transmit entry, the DMA will be presented with a scheduling credit for that exact transmit channel. If this is a receive entry, the DMA will be presented with a scheduling credit for the receive FIFO that is associated with this channel. For receive FIFOs which carry traffic for more than one receive DMA channel, the exact channel number that is given in the receive credit will actually be the channel number which is currently on the head element of that Rx FIFO, which is not necessarily the channel number given in the scheduler table entry.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.75 Queue Manager Revision Identification Register (QMGRREVID)

The queue manager revision identification register (QMGRREVID) contains the major and minor revisions for the module. The QMGRREVID is shown in Figure 31-101 and described in Table 31-105.

Figure 31-101. Queue Manager Revision Identification Register (QMGRREVID)

31	0
REV	
R-0052 1200h	

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

Table 31-105. Queue Manager Revision Identification Register (QMGRREVID) Field Descriptions

Bit	Field	Value	Description
31-0	REV	0052 1200h	Revision ID of the queue manager.

31.4.76 Queue Manager Queue Diversion Register (DIVERSION)

The queue manager queue diversion register (DIVERSION) is used to transfer the contents of one queue onto another queue. It does not support byte accesses. The queue manager queue diversion register (DIVERSION) is shown in Figure 31-102and described in Table 31-106.

Figure 31-102. Queue Manager Queue Diversion Register (DIVERSION)

31	30	29	16
HEAD_TAIL	Rsvd	DEST_QNUM	
W-0	R-0	W-0	

15	14	13	0
Reserved		SOURCE_QNUM	
R-0		W-0	

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

Table 31-106. Queue Manager Queue Diversion Register (DIVERSION) Field Descriptions

Bit	Field	Value	Description
31	HEAD_TAIL		Indicates whether queue contents should be merged on to the head or tail of the destination queue.
		0	Head
		1	Tail
30	Reserved	0	Reserved
29-16	DEST_QNUM	0-3FFFh	Destination Queue Number
15-14	Reserved	0	Reserved
13-0	SOURCE_QNUM	0-3FFFh	Source Queue Number

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.77 Queue Manager Free Descriptor/Buffer Starvation Count Register 0 (FDBSC0)

The free descriptor/buffer queue starvation count register (FDBSC0) provides statistics about how many starvation events are occurring on the receive free descriptor/buffer queues. It does not support byte accesses. The free descriptor/buffer queue starvation count register (FDBSC0) is shown in Figure 31-103 and described in Table 31-107.

Figure 31-103. Queue Manager Free Descriptor/Buffer Starvation Count Register 0 (FDBSC0)

31	24	23	16
FDBQ3_STARVE_CNT		FDBQ2_STARVE_CNT	
RC-0		RC-0	

15	8	7	0
FDBQ1_STARVE_CNT		FDBQ0_STARVE_CNT	
RC-0		RC-0	

LEGEND: RC = Cleared on read; -*n* = value after reset

Table 31-107. Queue Manager Free Descriptor/Buffer Starvation Count Register 0 (FDBSC0) Field Descriptions

Bit	Field	Value	Description
31-24	FDBQ3_STARVE_CNT	0-FFh	This field increments each time the Free Descriptor/Buffer Queue 3 is read while it is empty. This field is cleared when read.
23-16	FDBQ2_STARVE_CNT	0-FFh	This field increments each time the Free Descriptor/Buffer Queue 2 is read while it is empty. This field is cleared when read.
15-8	FDBQ1_STARVE_CNT	0-FFh	This field increments each time the Free Descriptor/Buffer Queue 1 is read while it is empty. This field is cleared when read.
7-0	FDBQ0_STARVE_CNT	0-FFh	This field increments each time the Free Descriptor/Buffer Queue 0 is read while it is empty. This field is cleared when read.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.78 Queue Manager Free Descriptor/Buffer Starvation Count Register 1 (FDBSC1)

The free descriptor/buffer queue starvation count register 1 (FDBSC1) provides statistics about how many starvation events are occurring on the receive free descriptor/buffer queues. It does not support byte accesses. The free descriptor/buffer queue starvation count register 1 (FDBSC1) is shown in Figure 31-104 and described in Table 31-108.

Figure 31-104. Queue Manager Free Descriptor/Buffer Starvation Count Register 1 (FDBSC1)

31	24	23	16
FDBQ7_STARVE_CNT		FDBQ6_STARVE_CNT	
RC-0		RC-0	

15	8	7	0
FDBQ5_STARVE_CNT		FDBQ4_STARVE_CNT	
RC-0		RC-0	

LEGEND: RC = Cleared on read; -*n* = value after reset

Table 31-108. Queue Manager Free Descriptor/Buffer Starvation Count Register 1 (FDBSC1) Field Descriptions

Bit	Field	Value	Description
31-24	FDBQ7_STARVE_CNT	0-FFh	This field increments each time the Free Descriptor/Buffer Queue 7 is read while it is empty. This field is cleared when read.
23-16	FDBQ6_STARVE_CNT	0-FFh	This field increments each time the Free Descriptor/Buffer Queue 6 is read while it is empty. This field is cleared when read.
15-8	FDBQ5_STARVE_CNT	0-FFh	This field increments each time the Free Descriptor/Buffer Queue 5 is read while it is empty. This field is cleared when read.
7-0	FDBQ4_STARVE_CNT	0-FFh	This field increments each time the Free Descriptor/Buffer Queue 4 is read while it is empty. This field is cleared when read.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.79 Queue Manager Free Descriptor/Buffer Starvation Count Register 2 (FDBSC2)

The free descriptor/buffer queue starvation count register 2 (FDBSC2) provides statistics about how many starvation events are occurring on the receive free descriptor/buffer queues. It does not support byte accesses. The free descriptor/buffer queue starvation count register 2 (FDBSC2) is shown in Figure 31-105 and described in Table 31-109.

Figure 31-105. Queue Manager Free Descriptor/Buffer Starvation Count Register 2 (FDBSC2)

31	24	23	16
FDBQ11_STARVE_CNT		FDB10_STARVE_CNT	
RC-0		RC-0	

15	8	7	0
FDBQ9_STARVE_CNT		FDBQ8_STARVE_CNT	
RC-0		RC-0	

LEGEND: RC = Cleared on read; -*n* = value after reset

Table 31-109. Queue Manager Free Descriptor/Buffer Starvation Count Register 2 (FDBSC2) Field Descriptions

Bit	Field	Value	Description
31-24	FDBQ11_STARVE_CNT	0-FFh	This field increments each time the Free Descriptor/Buffer Queue 11 is read while it is empty. This field is cleared when read.
23-16	FDBQ10_STARVE_CNT	0-FFh	This field increments each time the Free Descriptor/Buffer Queue 10 is read while it is empty. This field is cleared when read.
15-8	FDBQ9_STARVE_CNT	0-FFh	This field increments each time the Free Descriptor/Buffer Queue 9 is read while it is empty. This field is cleared when read.
7-0	FDBQ8_STARVE_CNT	0-FFh	This field increments each time the Free Descriptor/Buffer Queue 8 is read while it is empty. This field is cleared when read.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.80 Queue Manager Free Descriptor/Buffer Starvation Count Register 3 (FDBSC3)

The free descriptor/buffer queue starvation count register 3 (FDBSC3) provides statistics about how many starvation events are occurring on the receive free descriptor/buffer queues. It does not support byte accesses. The free descriptor/buffer queue starvation count register 3 (FDBSC3) is shown in Figure 31-106 and described in Table 31-110.

Figure 31-106. Queue Manager Free Descriptor/Buffer Starvation Count Register 3 (FDBSC3)

31 – 24	23 – 16
FDBQ15_STARVE_CNT	FDB14_STARVE_CNT
RC-0	RC-0

15 – 8	7 – 0
FDBQ13_STARVE_CNT	FDBQ12_STARVE_CNT
RC-0	RC-0

LEGEND: RC = Cleared on read; -*n* = value after reset

Table 31-110. Queue Manager Free Descriptor/Buffer Starvation Count Register 3 (FDBSC3) Field Descriptions

Bit	Field	Value	Description
31-24	FDBQ15_STARVE_CNT	0-FFh	This field increments each time the Free Descriptor/Buffer Queue 15 is read while it is empty. This field is cleared when read.
23-16	FDBQ14_STARVE_CNT	0-FFh	This field increments each time the Free Descriptor/Buffer Queue 14 is read while it is empty. This field is cleared when read.
15-8	FDBQ13_STARVE_CNT	0-FFh	This field increments each time the Free Descriptor/Buffer Queue 13 is read while it is empty. This field is cleared when read.
7-0	FDBQ12_STARVE_CNT	0-FFh	This field increments each time the Free Descriptor/Buffer Queue 12 is read while it is empty. This field is cleared when read.

31.4.81 Queue Manager Linking RAM Region 0 Base Address Register (LRAM0BASE)

The queue manager linking RAM region 0 base address register (LRAM0BASE) sets the base address for the first portion of the Linking RAM. This address must be 32-bit aligned. It is used by the Queue Manager to calculate the 32-bit linking address for a given descriptor index. It does not support byte accesses. The queue manager linking RAM region 0 base address register (LRAM0BASE) is shown in Figure 31-107 and described in Table 31-111.

Figure 31-107. Queue Manager Linking RAM Region 0 Base Address Register (LRAM0BASE)

31 – 0
REGION0_BASE
R/W-0

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

Table 31-111. Queue Manager Linking RAM Region 0 Base Address Register (LRAM0BASE) Field Descriptions

Bit	Field	Value	Description
31-0	REGION0_BASE	0-FFFF FFFFh	This field stores the base address for the first region of the linking RAM. This may be anywhere in 32-bit address space but would be typically located in on-chip memory.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.82 Queue Manager Linking RAM Region 0 Size Register (LRAM0SIZE)

The queue manager linking RAM region 0 size register (LRAM0SIZE) sets the size of the array of linking pointers that are located in Region 0 of Linking RAM. The size specified the number of descriptors for which linking information is stored in this region. It does not support byte accesses. The queue manager linking RAM region 0 size register (LRAM0SIZE) is shown in Figure 31-108 and described in Table 31-112.

Figure 31-108. Queue Manager Linking RAM Region 0 Size Register (LRAM0SIZE)

31	16
Reserved	
R-0	

15	14	13	0
Reserved		REGION0_SIZE	
R-0		R/W-0	

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

Table 31-112. Queue Manager Linking RAM Region 0 Size Register (LRAM0SIZE) Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Reserved
13-0	REGION0_SIZE	0-3FFFh	This field indicates the number of entries that are contained in the linking RAM region 0. A descriptor with index less than region0_size value has its linking location in region 0. A descriptor with index greater than region0_size has its linking location in region 1. The queue manager will add the index (left shifted by 2 bits) to the appropriate regionX_base_addr to get the absolute 32-bit address to the linking location for a descriptor.

31.4.83 Queue Manager Linking RAM Region 1 Base Address Register (LRAM1BASE)

The queue manager linking RAM region 1 base address register (LRAM1BASE) is used to set the base address for the first portion of the Linking RAM. This address must be 32-bit aligned. It is used by the Queue Manager to calculate the 32-bit linking address for a given descriptor index. It does not support byte accesses. The queue manager linking RAM region 1 base address register (LRAM1BASE) is shown in Figure 31-109 and described in Table 31-113.

Figure 31-109. Queue Manager Linking RAM Region 1 Base Address Register (LRAM1BASE)

31	0
REGION1_BASE	
R/W-0	

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

Table 31-113. Queue Manager Linking RAM Region 1 Base Address Register (LRAM1BASE) Field Descriptions

Bit	Field	Value	Description
31-0	REGION1_BASE	0-FFFF FFFFh	This field stores the base address for the second region of the linking RAM. This may be anywhere in 32-bit address space but would be typically located in off-chip memory.

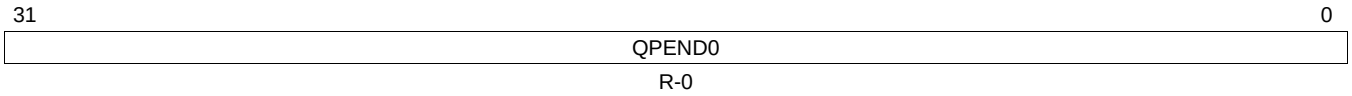
Copyright © 2013–2016, Texas Instruments Incorporated

31.4.84 Queue Manager Queue Pending Register 0 (PEND0)

The queue pending register 0 (PEND0) can be read to find the pending status for queues 31 to 0. It does not support byte accesses. The queue pending register 0 (PEND0) is shown in Figure 31-110 and described in Table 31-114.

NOTE: The pending bit gets set when a Descriptor address is loaded in a Queue. The loading action causes the corresponding bit for that Queue to get set. Similarly, the pending bit gets cleared when the Descriptor address is off-loaded from a Queue by reading it. One way to check if the receive or transmit transfer has completed is by checking the bit that corresponds to the desired Completion Queue for that particular transfer. When the Queue Manager is finished with the transfer, it will load the Descriptor address to the Completion Queue.

Figure 31-110. Queue Manager Queue Pending Register 0 (PEND0)



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

Table 31-114. Queue Manager Queue Pending Register 0 (PEND0) Field Descriptions

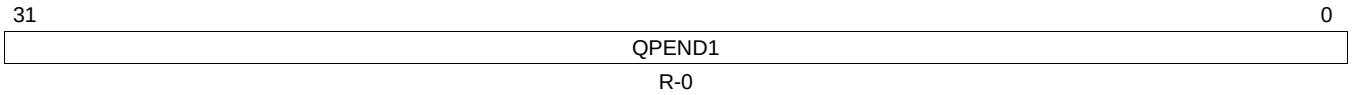
Bit	Field	Value	Description
31-0	QPEND0	0-FFFF FFFFh	This field indicates the queue pending status for queues 31-0.

31.4.85 Queue Manager Queue Pending Register 1 (PEND1)

The queue pending register 1 (PEND1) can be read to find the pending status for queues 63 to 32. It does not support byte accesses. The queue pending register 1 (PEND1) is shown in Figure 31-111 and described in Table 31-115.

NOTE: The pending bit gets set when a Descriptor address is loaded in a Queue. The loading action causes the corresponding bit for that Queue to get set. Similarly, the pending bit gets cleared when the Descriptor address is off-loaded from a Queue by reading it. One way to check if the receive or transmit transfer has completed is by checking the bit that corresponds to the desired Completion Queue for that particular transfer. When the Queue Manager is finished with the transfer, it will load the Descriptor address to the Completion Queue.

Figure 31-111. Queue Manager Queue Pending Register 1 (PEND1)



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

Table 31-115. Queue Manager Queue Pending Register 1 (PEND1) Field Descriptions

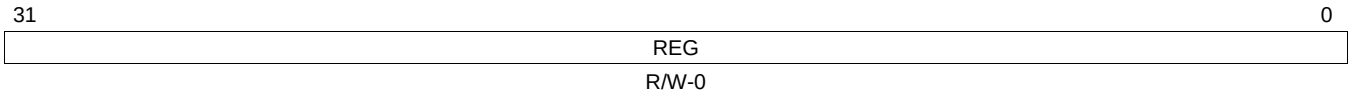
Bit	Field	Value	Description
31-0	QPEND1	0-FFFF FFFFh	This field indicates the queue pending status for queues 63-32.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.86 Queue Manager Memory Region *R* Base Address Registers (QMEMRBASE[0]-QMEMRBASE[15])

The memory region *R* base address register (QMEMRBASE[*R*]) is written by the host to set the base address of memory region *R*, where *R* is 0-15. This memory region will store a number of descriptors of a particular size as determined by the memory region *R* control register. It does not support byte accesses. The memory region *R* base address register (QMEMRBASE[*R*]) is shown in Figure 31-112 and described in Table 31-116.

Figure 31-112. Queue Manager Memory Region *R* Base Address Registers (QMEMRBASE[*R*])



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

Table 31-116. Queue Manager Memory Region *R* Base Address Registers (QMEMRBASE[*R*]) Field Descriptions

Bit	Field	Value	Description
31-0	REG	0-FFFF FFFFh	This field contains the base address of the memory region R.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.87 Queue Manager Memory Region *R* Control Registers (QMEMRCTRL[0]-QMEMRCTRL[15])

The memory region *R* control register (QMEMRCTRL[*R*]) is written by the host to configure various parameters of memory region *R*, where *R* is 0-15. It does not support byte accesses. The memory region *R* control register (QMEMRCTRL[*R*])) is shown in Figure 31-113 and described in Table 31-117.

Figure 31-113. Queue Manager Memory Region *R* Control Registers (QMEMRCTRL[*R*])

31	30	29	16
Reserved		START_INDEX	
R-0		R/W-0	

15	12	11	8	7	3	2	0
Reserved		DESC_SIZE		Reserved		REG_SIZE	
R-0		R/W-0		R-0		R/W-0	

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

Table 31-117. Queue Manager Memory Region *R* Control Registers (QMEMRCTRL[*R*]) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Reserved
29-16	START_INDEX	0-3FFFh	This field indicates where in linking RAM the descriptor linking information corresponding to memory region R starts.
15-12	Reserved	0	Reserved
11-8	DESC_SIZE	0-Fh	This field indicates the size of each descriptor in this memory region.
		0	32
		1h	64
		2h	128
		3h	256
		4h	512
		5h	1K
		6h	2K
		7h	4K
		8h	8K
		9h-Fh	Reserved
7-3	Reserved	0	Reserved
2-0	REG_SIZE	0-7h	This field indicates the size of the memory region (in terms of number of descriptors).
		0	32
		1h	64
		2h	128
		3h	256
		4h	512
		5h	1K
		6h	2K
		7h	4K

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.88 Queue Manager Queue N Control Register D (CTRLD[0]-CTRLD[63])

The queue manager queue *N* control register D (CTRLD[*N*]) is written to add a packet to the queue and read to pop a packets off a queue. The packet is only pushed or popped to/from the queue when the queue manager queue *N* control register D is written. It does not support byte accesses. The queue manager queue *N* control register D (CTRLD[*N*]) is shown in Figure 31-114 and described in Table 31-118.

Figure 31-114. Queue Manager Queue *N* Control Register D (CTRLD[*N*])

31	16
DESC_PTR	
R/W-0	

15	5	4	0
DESC_PTR		DESC_SIZE	
R/W-0		R/W-0	

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

Table 31-118. Queue Manager Queue *N* Control Register D (CTRLD[*N*]) Field Descriptions

Bit	Field	Value	Description
31-5	DESC_PTR		Descriptor Pointer
		0	Queue is empty.
		1	Indicates a 32-bit aligned address that points to a descriptor.
4-0	DESC_SIZE	0-1Fh	The descriptor size is encoded in 4-byte increments. This field returns a 0 when an empty queue is read.
		0	24 bytes
		1h	28 bytes
		2h	32 bytes
		3h-1Fh	36 bytes to 148 bytes

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.89 Queue Manager Queue *N* Status Register A (QSTATA[0]-QSTATA[63])

The queue manager queue *N* status register A (QSTATA[*N*]) is an optional register that is only implemented for a queue if the queue supports entry/byte count feature. The entry count feature provides a count of the number of entries that are currently valid in the queue. It does not support byte accesses. The queue manager queue *N* status register A (QSTATA[*N*]) is shown in Figure 31-115 and described in Table 31-119.

Figure 31-115. Queue Manager Queue *N* Status Register A (QSTATA[*N*])

31	16
Reserved	
R-0	

15	14	13	0
Reserved		QUEUE_ENTRY_COUNT	
R-0		R-0	

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

Table 31-119. Queue Manager Queue *N* Status Register A (QSTATA[*N*]) Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Reserved
13-0	QUEUE_ENTRY_COUNT	0-3FFFh	This field indicates how many packets are currently queued on the queue.

31.4.90 Queue Manager Queue *N* Status Register B (QSTATB[0]-QSTATB[63])

The queue manager queue *N* status register B (QSTATB[*N*]) is an optional register that is only implemented for a queue if the queue supports a total byte count feature. The total byte count feature provides a count of the total number of bytes in all of the packets that are currently valid in the queue. It does not support byte accesses. The queue manager queue *N* status register B (QSTATB[*N*]) is shown in Figure 31-116 and described in Table 31-120.

Figure 31-116. Queue Manager Queue *N* Status Register B (QSTATB[*N*])

31	28	27	16
Reserved		QUEUE_BYTE_COUNT	
R-0		R-0	

15	0
QUEUE_BYTE_COUNT	
R-0	

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

Table 31-120. Queue Manager Queue *N* Status Register B (QSTATB[*N*]) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reserved
27-0	QUEUE_BYTE_COUNT	0-FFF FFFFh	Indicates how many bytes total are contained in all of the packets which are currently queued on this queue.

Copyright © 2013–2016, Texas Instruments Incorporated

31.4.91 Queue Manager Queue *N* Status Register C (QSTATC[0]-QSTATC[63])

The queue manager queue *N* status register C (QSTATC[*N*]) specifies the packet size for the head element of a queue. It does not support byte accesses. The queue manager queue *N* status register C (QSTATC[*N*]) is shown in Figure 31-117 and described in Table 31-121.

Figure 31-117. Queue Manager Queue *N* Status Register C (QSTATC[*N*])

31	16
Reserved	
R-0	

15	14	13	0
Reserved		PACKET_SIZE	
R-0		R-0	

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

Table 31-121. Queue Manager Queue *N* Status Register C (QSTATC[*N*]) Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Reserved
13-0	PACKET_SIZE	0-3FFFh	This field indicates how many packets are currently queued on the queue.

Copyright © 2013–2016, Texas Instruments Incorporated

Video Port Interface (VPIF)

This chapter describes the video port interface (VPIF).

Topic **Page**

32.1 Introduction **1525**
32.2 Architecture **1529**
32.3 Registers **1550**

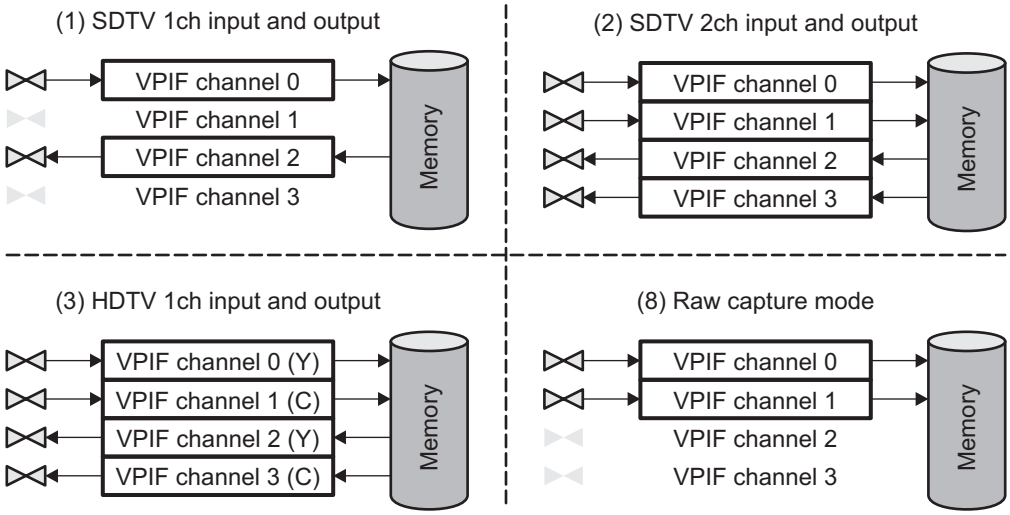
Copyright © 2013–2016, Texas Instruments Incorporated

32.1 Introduction

32.1.1 Overview

The video port interface (VPIF) has two video input channels and two video output channels. Channels 0 and 1 share the same receive architecture, and channels 2 and 3 share the same transmit architecture. The input and output channel combinations of the VPIF are shown in Figure 32-1.

Figure 32-1. Input and Output Channels of VPIF



Copyright © 2013–2016, Texas Instruments Incorporated

32.1.2 Features

The VPIF is designed to support the following features (note that some device designs may support reduced features because of system-level performance limitations):

- ITU-BT.656 format
- ITU-BT.1120 and SMTPE 296 formats
- Raw data capture
- VBI data storage
- Clipping of output data (to eliminate FFh and 00h values)

32.1.3 Features Not Supported

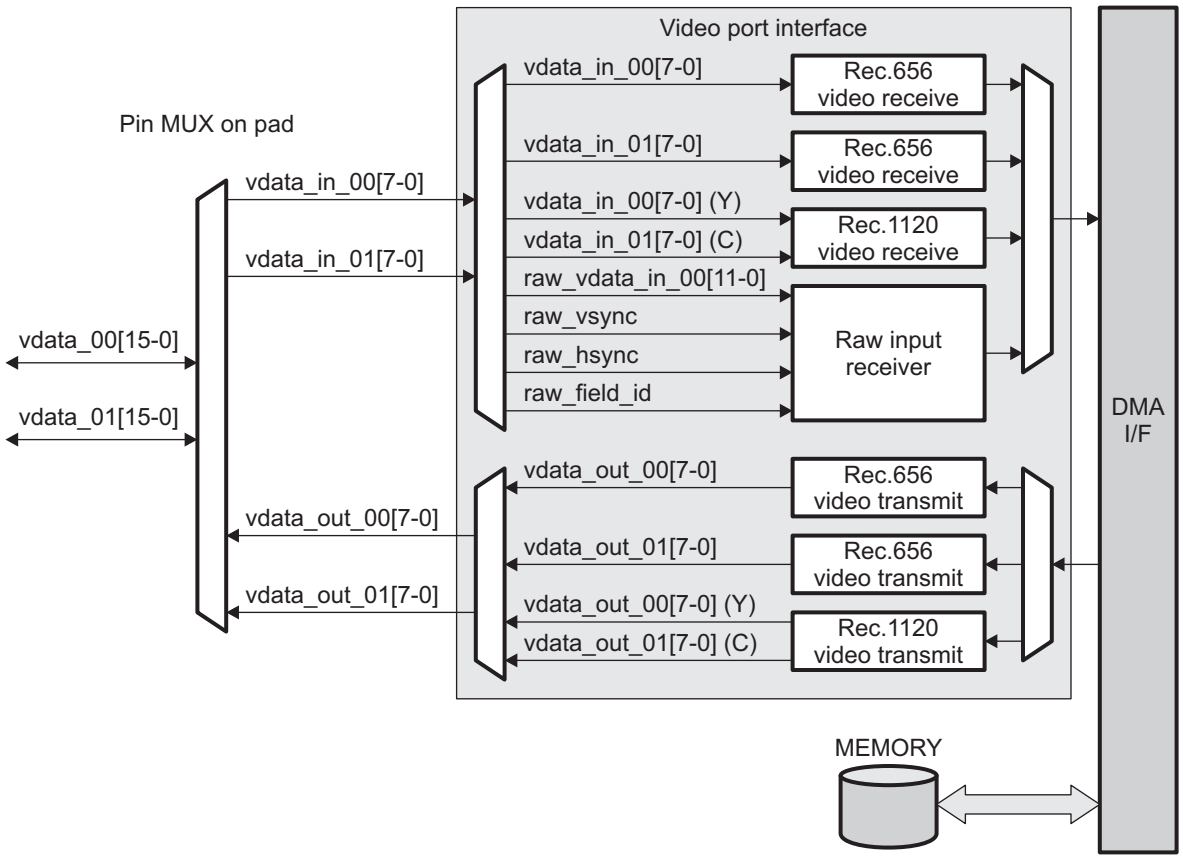
The following functions are not supported:

- ITU-BT.601 format
- Separated synchronization format (requiring vertical sync, horizontal sync, and field ID signals independent of the pixel data) is not supported (except for the raw data capture mode)

32.1.4 Functional Block Diagram

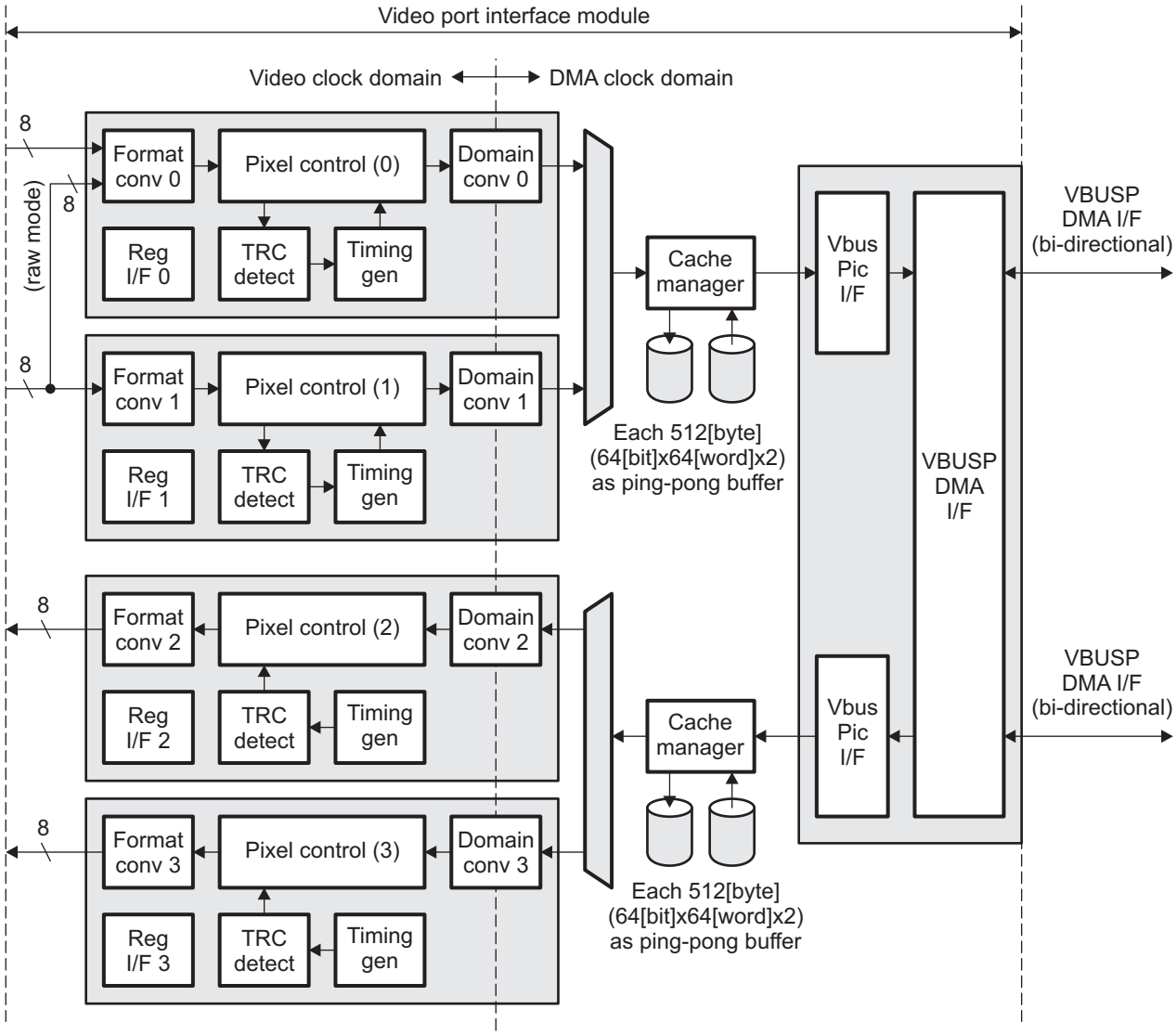
A block diagram of the VPIF is shown in Figure 32-2. A block diagram of the internal architecture of the VPIF is shown in Figure 32-3.

Figure 32-2. Video Port Interface (VPIF) Block Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

Figure 32-3. VPIF Architecture Block Diagram



Copyright © 2013–2016, Texas Instruments Incorporated

Supported Use Cases

The following describes supported VPIF use cases. Actual VPIF performance may vary between devices designs because of system-level limitations.

The VPIF module has two input channels and two output channels. All channels can be activated simultaneously (see Table 32-1). The module uses four 512 byte buffers to cache data.

- Channels 0 and 1 are prepared only for input.
- Channels 2 and 3 are prepared only for output.

As shown in Table 32-1, both NTSC and PAL formats are supported. The supported VBI format is be based on ITU-BT.1364. Table 32-2 describes the usage combinations that are supported in the VPIF.

Table 32-1. Supported Formats on VPIF

	TV Definition Format	
TV System Format	**HDTV (rec. 1120) (no support of ancillary data)**	**SDTV (rec. 656) (ancillary data is based on BT.1364)**
NTSC	1125 line/60 field (vertical)	525 line/60 field (vertical)
	2200 pixel (horizontal)	858 pixel (horizontal)
PAL	1250 line/50 field (vertical)	625 line/50 field (vertical)
	2304 pixel (horizontal)	864 pixel (horizontal)
Square pixel common image format	1080-30p	-

Table 32-2. Input and Output Usage Combinations on VPIF

	Output Format		
Input Format	**HDTV Output**	**SDTV Output**	**No Output**
HDTV input (1 channel only)	√	√	√
Raw capture mode	√	√	√
SDTV input	√ (both 1-channel and 2-channel input)	√ (both 1-channel and 2-channel input/output)	√ (both 1-channel and 2-channel input)
No input	√	√ (both 1-channel and 2-channel output)	

Copyright © 2013–2016, Texas Instruments Incorporated

32.2 Architecture

This section describes the architecture of the video port interface module (VPIF).

32.2.1 Clock Control

The VPIF has 4 clock input pins and 2 clock output pins. Each channel has 1 clock input pin with clock edge control using the CLKEDGE bit in the channel *n* control register (C*n*CTRL). VPIF can provide a clock for an external device on channel 2 or 3 using the CLKEN bit in the appropriate C2CTRL or C3CTRL register. The clock generated by VPIF will have the same frequency as the input clock.

32.2.2 Signal Descriptions

Table 32-3 describes the pin assignment on the VPIF for receiving video and capturing raw data. Shaded signals in Table 32-3 are synchronization signals that are necessary for capturing raw data.

Table 32-4 describes the pin assignments for transmitting video data.

Table 32-3. Receive Pin Multiplexing Control

Pin Name	Rec. 656	Rec. 1120	Raw Data Capture	Pin Name	Rec. 656	Rec. 1120	Raw Data Capture
DIN[0]	data0[0]	luma[0]	raw_data[0]	DIN[8]	data1[0]	chroma[0]	raw_data[8]
DIN[1]	data0[1]	luma[1]	raw_data[1]	DIN[9]	data1[1]	chroma[1]	raw_data[9]
DIN[2]	data0[2]	luma[2]	raw_data[2]	DIN[10]	data1[2]	chroma[2]	raw_data[10]
DIN[3]	data0[3]	luma[3]	raw_data[3]	DIN[11]	data1[3]	chroma[3]	raw_data[11]
DIN[4]	data0[4]	luma[4]	raw_data[4]	DIN[12]	data1[4]	chroma[4]	not used
DIN[5]	data0[5]	luma[5]	raw_data[5]	DIN[13]	data1[5]	chroma[5]	raw_field_id
DIN[6]	data0[6]	luma[6]	raw_data[6]	DIN[14]	data1[6]	chroma[6]	raw_h_valid
DIN[7]	data0[7]	luma[7]	raw_data[7]	DIN[15]	data1[7]	chroma[7]	raw_v_valid
CLKIN0	data0_clk	luma_clk	raw_data_clk	CLKIN1	data1_clk	chroma_clk	raw_data_clk

Table 32-4. Transmit Pin Multiplexing Control

Pin Name	Trans. 656	Trans. 1120	Pin Name	Trans. 656	Trans. 1120
DOUT[0]	data2[0]	luma[0]	DOUT[8]	data3[0]	chroma[0]
DOUT[1]	data2[1]	luma[1]	DOUT[9]	data3[1]	chroma[1]
DOUT[2]	data2[2]	luma[2]	DOUT[10]	data3[2]	chroma[2]
DOUT[3]	data2[3]	luma[3]	DOUT[11]	data3[3]	chroma[3]
DOUT[4]	data2[4]	luma[4]	DOUT[12]	data3[4]	chroma[4]
DOUT[5]	data2[5]	luma[5]	DOUT[13]	data3[5]	chroma[5]
DOUT[6]	data2[6]	luma[6]	DOUT[14]	data3[6]	chroma[6]
DOUT[7]	data2[7]	luma[7]	DOUT[15]	data3[7]	chroma[7]
CLKIN2	data2_ref_clk	luma_ref_clk	CLKIN3	data3_ref_clk	chroma_ref_clk
CLKOUT2	data2_clk	luma_clk	CLKOUT3	data3_clk	chroma_clk

Copyright © 2013–2016, Texas Instruments Incorporated

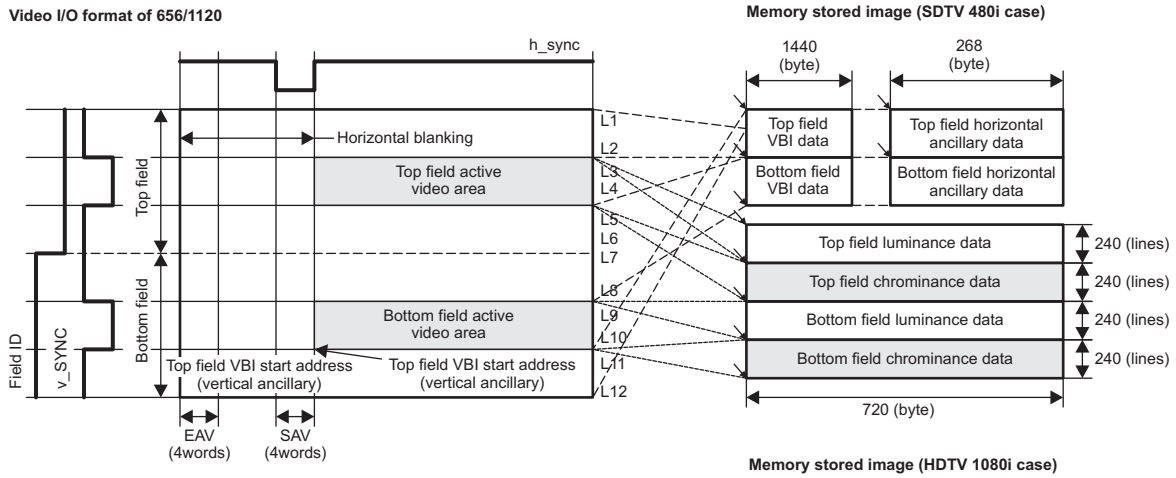
Interlaced and Progressive Video

VPIF supports both interlaced and progressive video formats.

Interlaced Video

The components of interlaced video are shown in Figure 32-4.

Figure 32-4. Interlaced Video



	Role of each line number	HDTV case	SDTV case	F	V
L1	First line of top field	1 (1)	4 (1)	0	1
L2	Last line of digital field blanking for top-field	40 (44)	19 (22)	0	1
L3	First line of top field for active video area	41 (45)	20 (23)	0	0
L4	Last line of top-field for active video area	557 (620) (video size = 517/576)	263 (310)	0	0
L5	First line of digital field blanking for bottom of field	556 (621)	264 (311)	0	1
L6	Last line of top-field	563 (625)	265 (312)	0	1
L7	First line of bottom field	564 (626)	266 (313)	1	1
L8	Last line of digital field blanking for bottom field	602 (669)	282 (335)	1	1
L9	First line of bottom field for active video area	603 (670)	283 (336)	1	0
L10	Last line of bottom field for active video area	1120 (1245)	525 (623)	1	0
L11	First line of digital field blanking for top-field	1121 (1246)	1 (624)	1	1
L12	Last line of bottom field	1125 (1250)	3 (625)	1	1

The values in the table are from the BT.656 and BT.1120 standards. L11 is regarded as the start line of the frame in the VPIF.

All parameters (L1 to L12) for interlaced video are configured through VPIF register settings. Values listed in Figure 32-4 are for NTSC and (PAL).

Interlaced video is divided into eight buffer spaces:

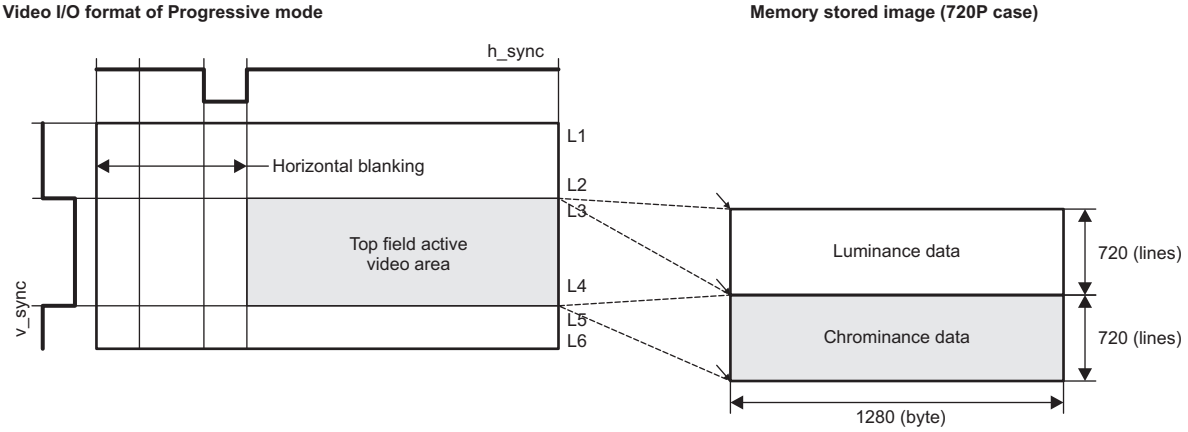
1. Top field VBI data in vertical ancillary area
2. Top field VBI data in horizontal ancillary area
3. Bottom field VBI data in vertical ancillary area
4. Bottom field VBI data in horizontal ancillary area
5. Top field luminance data
6. Top field chrominance data
7. Bottom field luminance data
8. Bottom field chrominance data

Copyright © 2013–2016, Texas Instruments Incorporated

Progressive Video

The components of progressive video are shown in Figure 32-5.

Figure 32-5. Progressive Video



	Role of each line number	HDTV case	F	V
L1	First line of upper blanking for the frame	1	0	1
L2	Last line of upper blanking for the frame	25	0	1
L3	First line of the frame for active video area	26	0	0
L4	Last line of the frame for active video area	745	0	0
L5	First line of lower blanking for the frame	746	0	1
L6	Last line of lower blanking for the frame	750	0	1

The buffer register configurations are latched at L1 in progressive format. Data on L5 to L6 is stored into a different memory buffer from data on L1 to L2 because the base address is changed at timing of L1.

All parameters (L1 to L6) for progressive video are configured through VPIF register settings.

Progressive video is divided into four buffer spaces:

1. VBI data in vertical ancillary area
2. VBI data in horizontal ancillary area
3. Luminance data
4. Chrominance data

32.2.3 Memory Interface

DMA Request Size

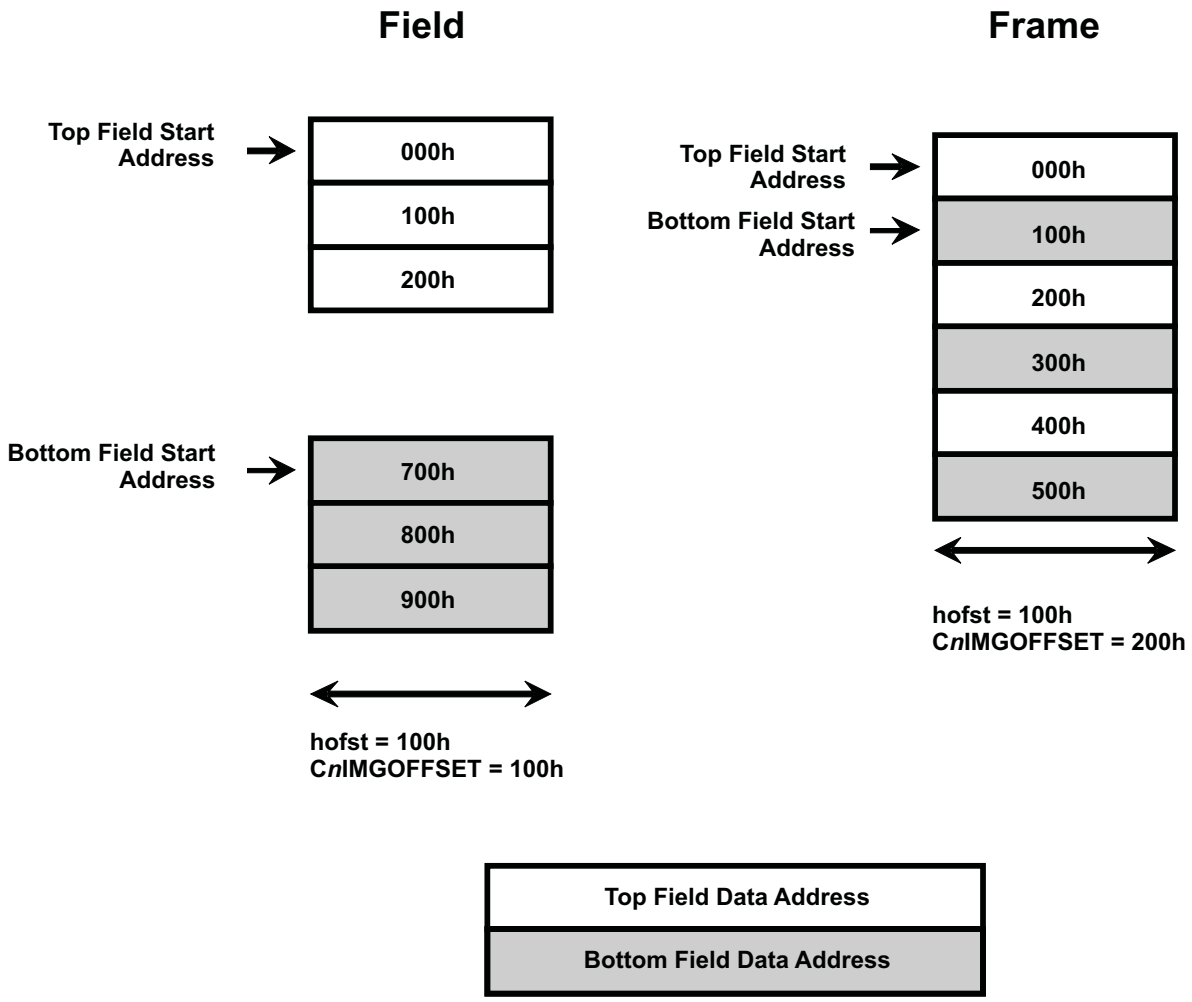
The DMA burst size between the VPIF module and storage memory is configured through the DMA Request Size Control Register (REQSIZE).

Storage Format

The VPIF supports two data buffer memory addressing modes for luminance and chrominance data: Field and Frame, as shown in Figure 32-6. The FIELDFRAME bit (in C0CTRL and C2CTRL) selects which storage method is used. Progressive video must use the Frame storage mode, but interlaced video can use either Field or Frame storage modes.

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 32-6. Memory Storage Modes for Interlaced Video



The storage modes are optimized for the following use conditions for interlaced video:

- Field format
  - Top- and bottom-field data-sets are stored in isolated buffers
  - Top- and bottom-field buffer start addresses are independent
  - Address offset per line of data in each top- and bottom-field buffer is equal to hofst (C*n*IMGOFFSET = hofst)
- Frame format
  - Top- and bottom-field data-sets are interlaced in memory
  - Top-field buffer start address is coupled with bottom-field buffer start address (Top Start = Bottom Start - hofst)
  - Bottom-field buffer start address is coupled with top-field buffer start address (Bottom Start = Top Start + hofst)
  - Address offset per line of data in each top- or bottom-field buffer is equal to hofst × 2 (C*n*IMGOFFSET = hofst × 2)

Note that because progressive video only uses top-field buffers, the address offset per line of data is equal to hofst (C*n*IMGOFFSET = hofst).

Note that when capturing 10-bit or 12-bit video in the raw capture mode, the offset per line of data is doubled because of bit padding.

Copyright © 2013–2016, Texas Instruments Incorporated

32.2.4 Video Transmit

VPIF channels 2 and 3 are used for video transmit.

Y/C Mux

For BT.656 video, luminance (Y) and chrominance (C) values are multiplexed into a single byte-stream on one channel (either channel 2 or channel 3).

For BT.1120 video, channels 2 and 3 function as a pair without Y/C multiplexing in the following configuration:

- Channel 2 transmits Y data (C buffer settings for channel 2 are ignored)
- Channel 3 transmits C data (Y buffer settings for channel 3 are ignored)

The Y/C multiplex function for BT.656 video is enabled by setting the YCMUX bit in the channel control registers C*n*CTRL. The YCMUX bit must be the same for both channels 2 and 3.

Transmit Clocking

Each transmit channel uses a reference input clock (CLKIN*n*) and a video output clock (CLKOUT*n*). When transmit is enabled, VPIF will generate CLKOUT*n* with the same frequency as CLKIN*n*, and video data will be transmitted synchronously with CLKOUT*n*.

The CLKOUT*n* output signal is enabled by setting the CLKEN bit in the channel control register C*n*CTRL; the signal can also be inverted by setting the CLKEDGE bit in the C*n*CTRL register.

When transmitting BT.1120 video data, the two transmit channels operate as a pair. To ensure that video data is transmitted at the same time across both channels, the CLKIN*n* signals must share the same reference clock source.

Video Pixel Enable

The PIXEL bit of the C2CTRL and C3CTRL control registers enables the transmission of video data stored in memory. When the PIXEL bit is set, video data is transmitted normally. When the PIXEL bit is cleared, the VPIF will transmit blank video data (Y = 10h, C = 80h). The PIXEL bit should only be set when the associated VPIF channel is disabled, otherwise the VPIF may output corrupt data and issue an ERROR interrupt for buffer underflow.

Copyright © 2013–2016, Texas Instruments Incorporated

32.2.5 Video Receive

VPIF channels 0 and 1 are used for video receive.

Y/C Mux

For BT.656 video, luminance (Y) and chrominance (C) values are multiplexed into a single byte-stream on one channel (either channel 0 or channel 1).

For BT.1120 video, channels 0 and 1 function as a pair without Y/C multiplexing in the following configuration:

- Channel 0 receives Y data (C buffer settings for channel 0 are ignored)
- Channel 1 receives C data (Y buffer settings for channel 1 are ignored)

The Y/C multiplex function for BT.656 video is enabled by setting the YCMUX bit in the channel control registers C*n*CTRL. The YCMUX bit must be the same for both channels 0 and 1.

Receive Clocking

Each receive channel uses an input clock (CLKIN*n*) . When receive is enabled, video data is latched synchronously with the rising edge of CLKIN*n*. The VPIF can be reconfigured to latch on the falling edge of CLKIN*n* by setting the CLKEDGE bit in the C*n*CTRL register.

When receiving BT.1120 video data, the two receive channels operate as a pair. To ensure that video data is received at the same time across both channels, the CLKIN*n* signals must share the same reference clock.

Capture Mode

The CAPMODE bit of the C0CTRL and C1CTRL control registers determines what data format will be captured by VPIF. The BT/YC video mode will look for video sync signals that are embedded within the video byte stream (standard for BT video). The CCD/CMOS (Raw Data Capture) mode will look for video sync signals on the dedicated VPIF sync pins (common for CCD and CMOS sensors).

When the CCD/CMOS mode is selected, channels 0 and 1 will work as a pair. Thus, both channels must be enabled before VPIF will capture receive data.

Capture mode settings must be the same for both channels 0 and 1.

Copyright © 2013–2016, Texas Instruments Incorporated

32.2.6 Raw Data Capture

The VPIF supports raw data capturing. With this function, you can connect a camera AFE output signal directly into an input port of the CPU. Usually, the output format of the camera AFE device is in raw format that consists of an RGB component (sometimes RGrGbB format). The following functions are supported:

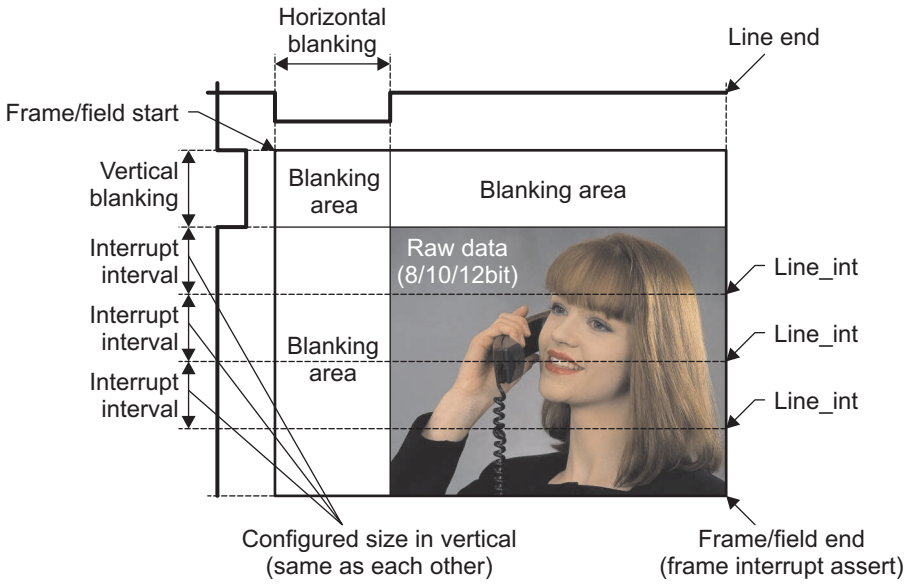
- Storing pixel data in memory (no storage of blanking data).
- Data bit widths from 8 bits/pixel, 10 bits/pixel, and 12 bits/pixel mode.
- Selectable polarity for H/V pixel valid signal and field ID signal.
- Separated field storage (top field and bottom field are stored independently) and interleaved field storage (normal frame format) support in memory storage.
- Two kinds of interrupt support. One is asserted once per each configured line size (line_interrupt through the FRAME1 signal) and the other is asserted at the end of the capture area (frame_interrupt through the FRAME0 signal). See Figure 32-7. Note that line_interrupt is only supported in raw mode. In other modes (BT.656, BT.1120, and SMPTE 296M), line_interrupt is not supported.

The following functions are not supported:

- No support of color space conversion from RGB to YCbCr.
- No CFA interpolation for each raw data pattern (such as Bayer or Foveon).
- No push-storage function on non-byte aligned data format (10 bits/pixel and 12 bits/pixel). Data should be stored in memory in byte-aligned format.

The active period of each synchronization signal is regarded as the blanking area and any other area is regarded as the active video area that is stored in memory. See Figure 32-7.

Figure 32-7. Functional Image of Raw Data Capturing Mode



All register configurations related to raw capture mode are reflected with the falling edge of the internal V-sync, which source is raw_v_valid, in normal polarity (low = blanking, high = data).

32.2.6.1 Raw Capture Mode Signals

Both interlaced and progressive interface modes are supported. The following signals are assigned to the interface signal of the raw capture mode:

- raw_h_valid: horizontal pixel valid signal (regarded as horizontal synchronization signal)
- raw_v_valid: vertical line valid signal (regarded as vertical synchronization signal)
- raw_field_id: field ID signal
- vin_data_raw[11:0]: raw data input (8 bits/pixel, 10 bits/pixel, and 12 bits/pixel)

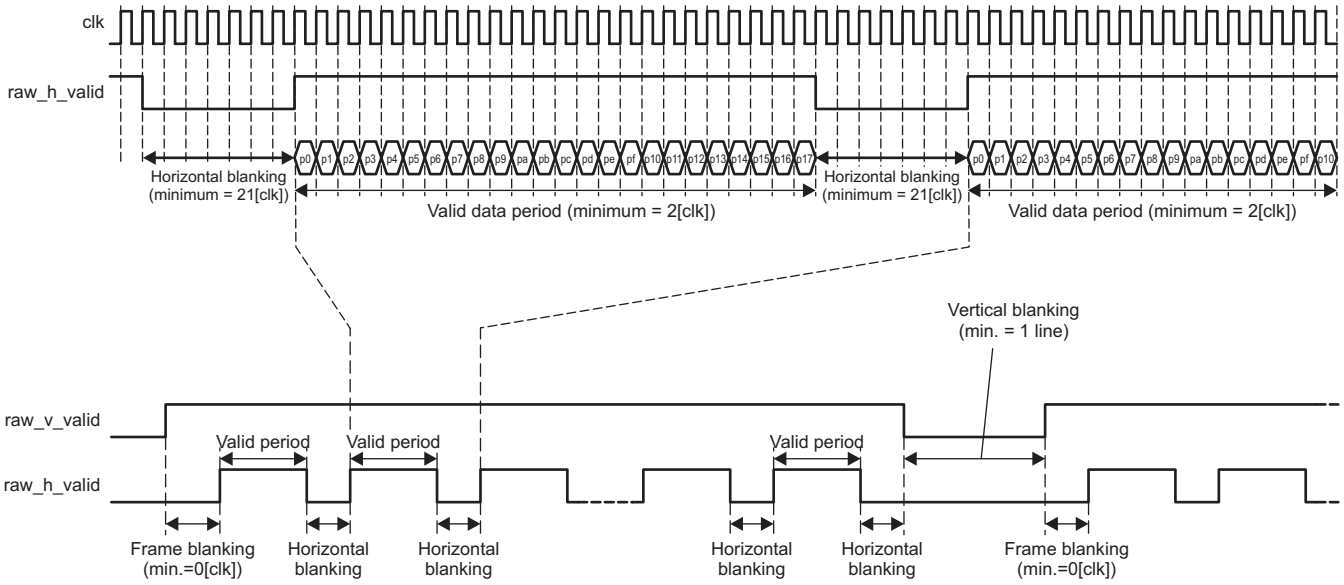
Copyright © 2013–2016, Texas Instruments Incorporated

32.2.6.1.1 Progressive CCD Raw Capture Mode

The CCD Raw Capture mode is illustrated in Figure 32-8. In this mode, data within the active periods of both raw_v_valid and raw_h_valid is captured by VPIF. The falling edges of the two valid signals serve as the vertical and horizontal synchronization signals (the valid signal edge polarity can be configured in the C0CTRL register). You have to set the image address offset.

In this mode, without the activated period of raw_v_valid, no raw_h_valid signal is activated. Only in the period when both raw_v_valid and raw_h_valid signals are activated, the incoming data is regarded as valid data.

Figure 32-8. Raw Capture Progressive Mode



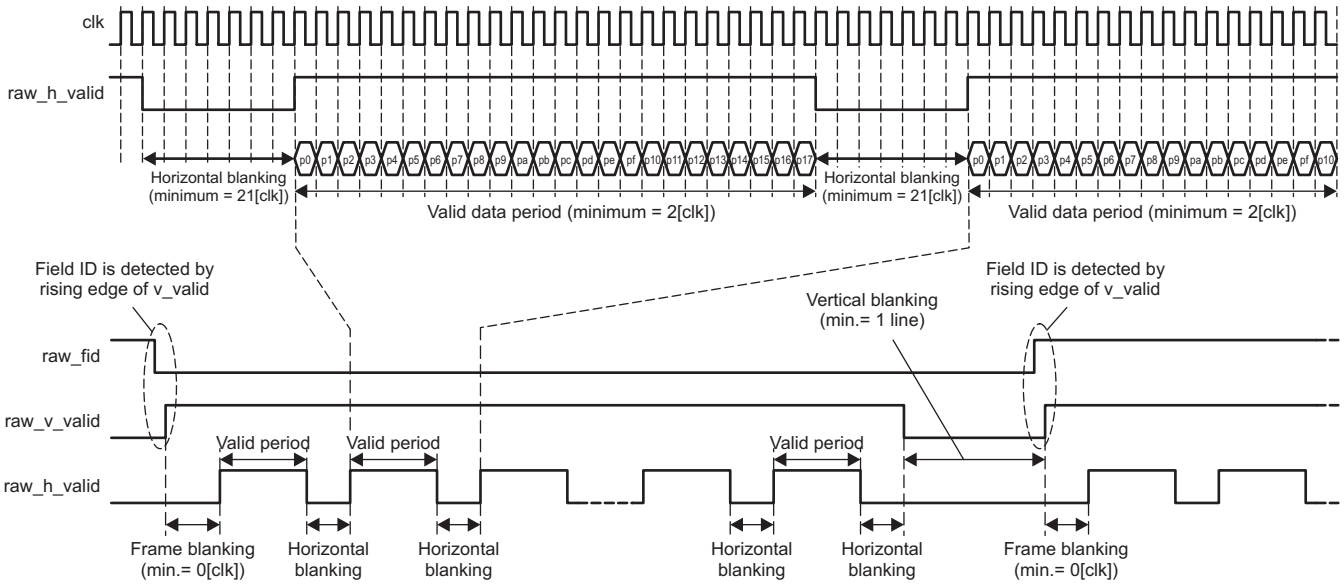
Copyright © 2013–2016, Texas Instruments Incorporated

32.2.6.1.2 Interlaced CCD Raw Capture Mode

See Figure 32-9. The falling edge of the vertical valid signal (raw_v_valid) and the horizontal valid signal (raw_h_valid) is regarded as the normal vertical and horizontal synchronization signals, respectively. The field ID (raw_fid) is detected at the rising edge of the raw_v_valid signal. The description of the detail value in Figure 32-8 is based on the description on the Micron Image sensor device specification sheet.

In this mode, data with an active period of both of raw_v_valid and raw_h_valid is detected by the VPIF and valid data is stored in extra-VPIF memory. The falling edge of the two signals is regarded as the vertical and horizontal synchronization signals (the valid signal polarity and the field ID polarity can be configured by the channel 0 control register (C0CTRL)). You have to set the image address offset.

Figure 32-9. Raw Capture Interlaced Mode



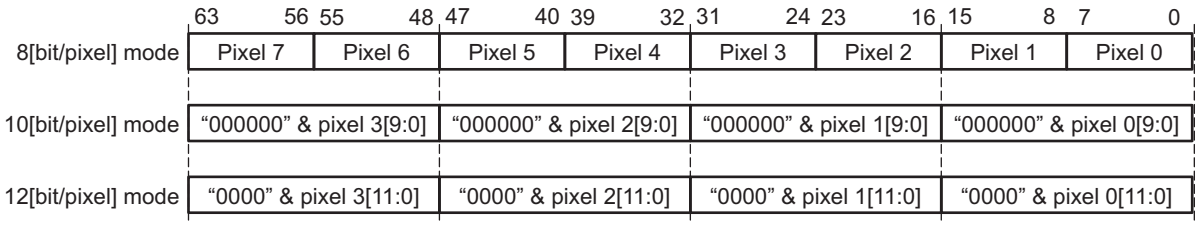
32.2.6.2 Configuration Settings for Raw Capture Mode

In raw capture mode, VPIF automatically detects the incoming image size by using the raw_h_valid and raw_v_valid signals. Thus, the C*n*HCFG, C*n*VCFG0, C*n*VCFG1, and C*n*VSIZE registers are not required.

32.2.6.3 Memory Format on Raw Capture Mode

All active video data is stored in memory outside of VPIF in byte-aligned format. Bit stuffing is performed (as shown in Figure 32-10) when 10 bits/pixel or 12 bits/pixel mode is selected.

Figure 32-10. Stuffing Manner in Storage Memory



Copyright © 2013–2016, Texas Instruments Incorporated

32.2.7 VBI Ancillary Data

The VPIF can transmit and receive ancillary data in the video blanking interval (VBI) of BT.656 and BT.1120 video streams. The ancillary data regions are referred to as:

- Horizontal Ancillary (HANC) - Data between EAV and SAV (horizontal blanking interval), and
- Vertical Ancillary (VANC) - Data between SAV and EAV (horizontal active video area).

Transmit and receive of ancillary data is enabled by setting the HANC and VANC bits in the C0CTRL and C2CTRL control registers.

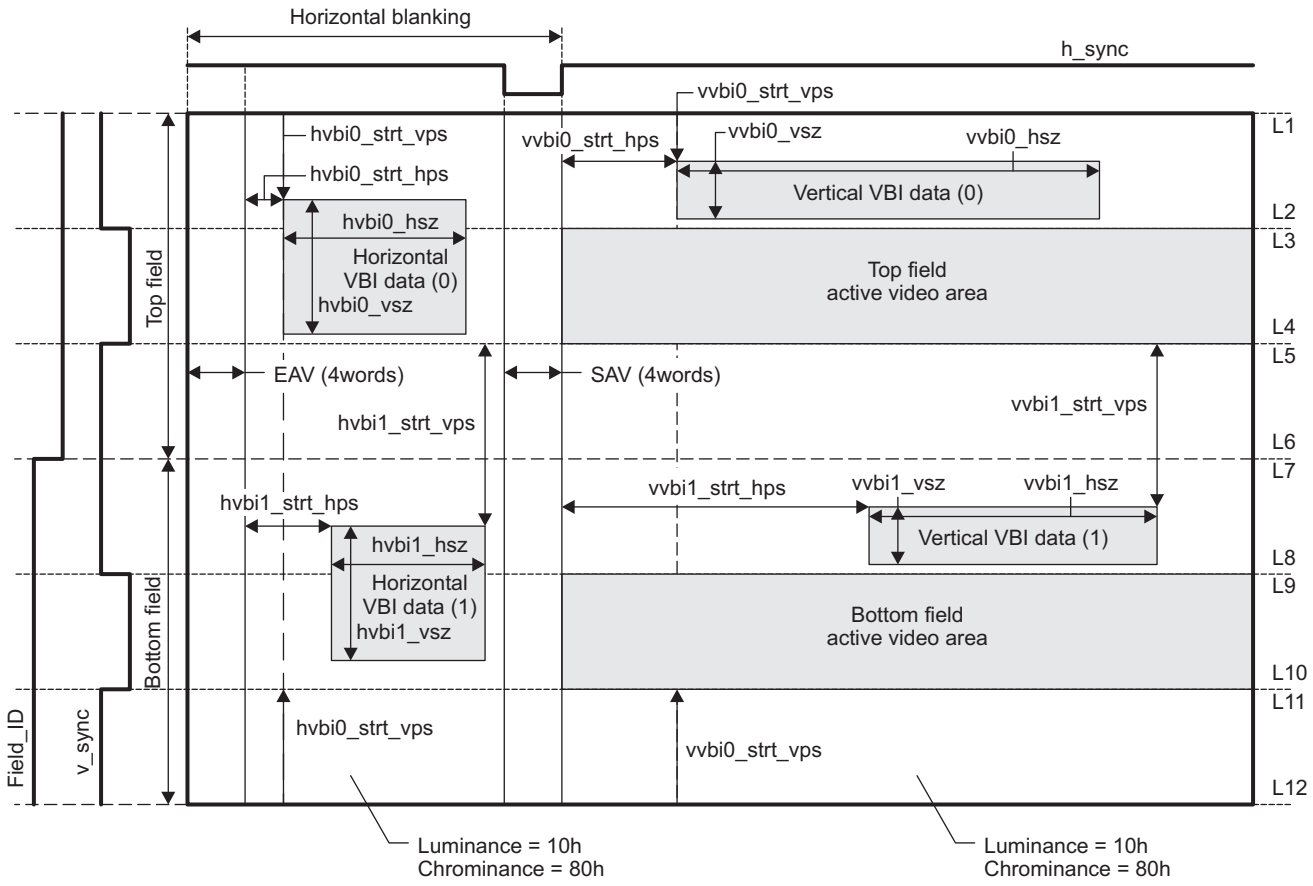
32.2.7.1 VBI Ancillary Data Transmit

The ancillary data to be transmitted is prepared by the user outside of VPIF. Ancillary register settings determine the origin position of the ancillary data within the VBI and how much ancillary data is to be transmitted. Note that the user is expected to place valid ancillary data in a memory buffer that is representative of the entire VBI region of interest. However, only the valid ancillary data region needs to be initialized -- the VPIF will automatically transmit blanking data (Y = 10h, C = 80h) for non-valid ancillary data regions.

As shown in Figure 32-11, the vertical-origin of each data region is offset from the falling edge of the vertical synchronization signal, and the horizontal-origin of each data region is offset from the SAV and EAV signals.

The value of the horizontal start position should be a multiple of 8 (HPOS[2:0] = 0) on both the horizontal and vertical ancillary data regions; the horizontal size can be defined as any value that does not exceed the data region.

Figure 32-11. VBI Result Data Transmit Image for Interlaced Image



Copyright © 2013–2016, Texas Instruments Incorporated

32.2.7.2 VBI Ancillary Data Receive

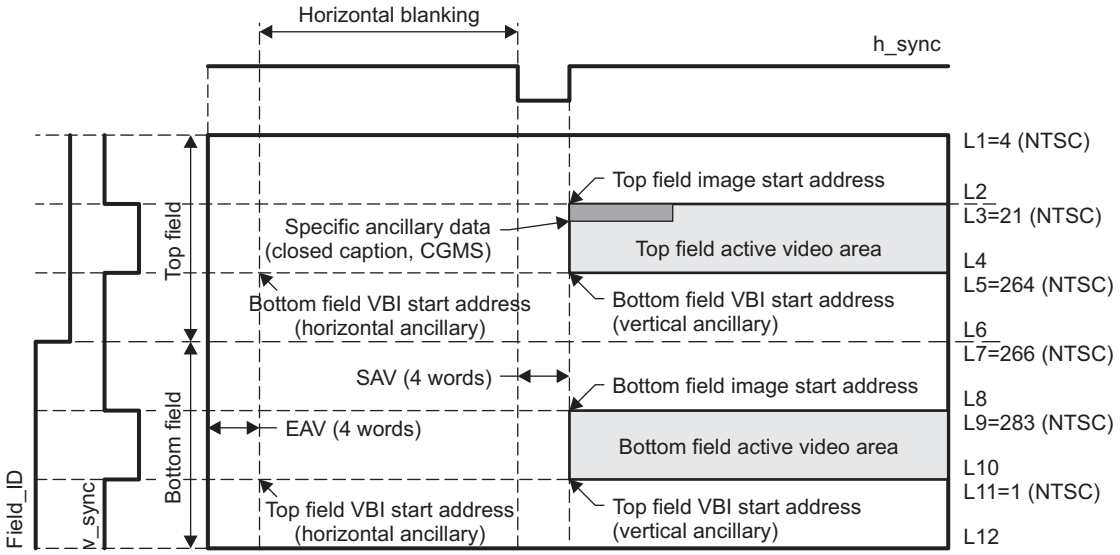
When the VBI receive function is enabled, the VPIF receives all data in the associated blanking region. The VPIF cannot selectively receive sub-regions in the VBI space. The CPU has to receive valid data from this data and you have to prepare for the correct size of data buffer. For example in the NTSC case, the horizontal ancillary data needs 268 bytes × 525 lines and the vertical ancillary data needs 1440 bytes × 38 lines buffer.

The address line offset for the vertical ancillary data uses the channel *n* image data address offset register (C*n*IMGOFFSET).

Processing Method for Specific Ancillary Data

The VPIF has the ability to capture/assert video ancillary data (Image of Specific Ancillary Data on NTSC) that is not video image data but is VBI data. In most cases, video ancillary data is inserted in the blanking interval for either the horizontal or vertical direction. But in some cases, such as CGMS or closed-caption that is in Japanese and US applications, the line number where these kinds of ancillary data is inserted is in the active video area. In the case that ancillary data is inserted in the active video area, the VPIF regards the incoming (or stored data) as video data.

Image of Specific Ancillary Data on NTSC



Clipping Function for Output

The clipping function is prepared for avoidance of miss-synchronization on an external device (video encoder), which receives output of the VPIF module. In some cases, the source data for the active video area has a combination of FF-00-00, which is the same as TRC (synchronization code). Some video encoders regard this combination as part of TRC and then the synchronization is failed.

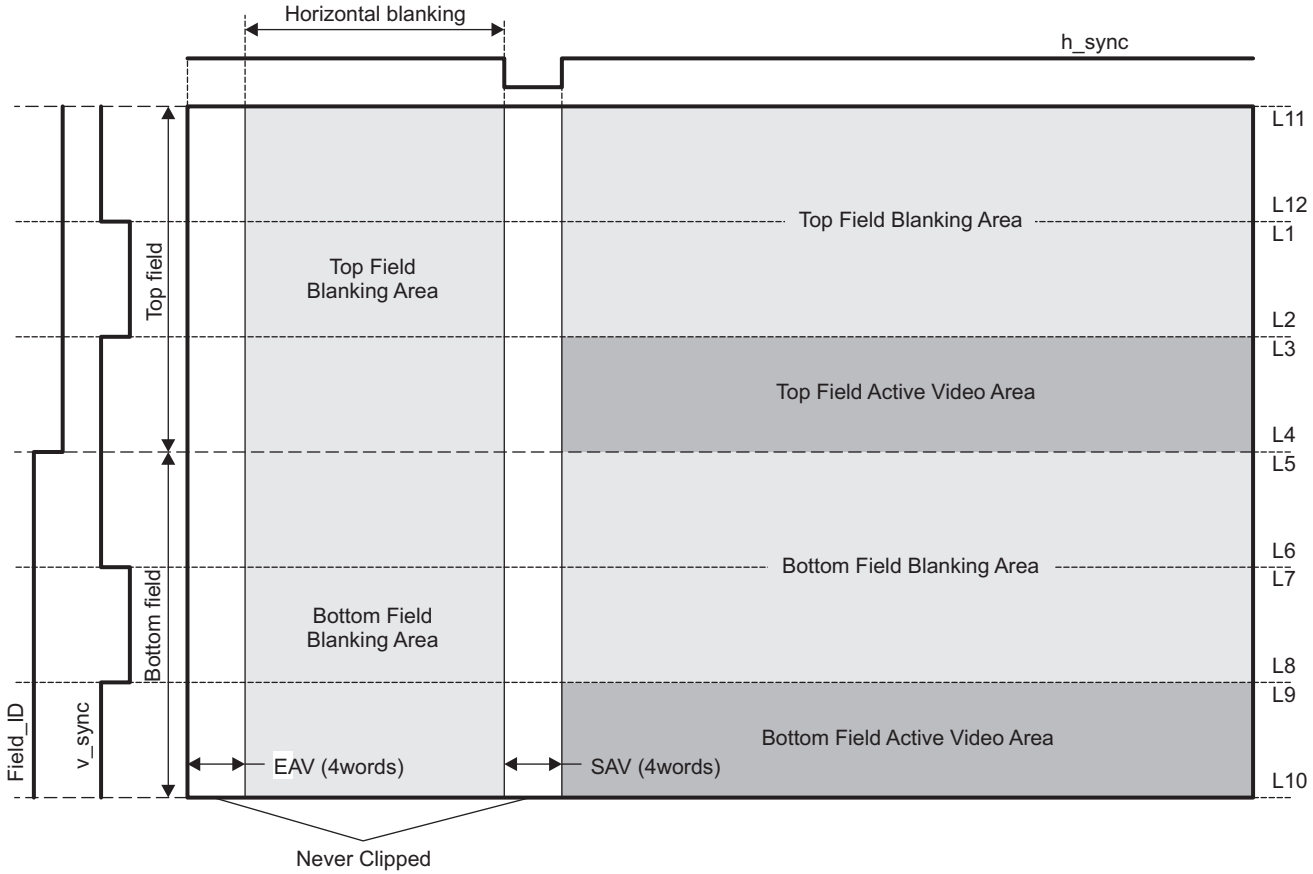
In order to avoid this issue, the VPIF module is capable of clipping the output data value except the TRC within the range of 01~FEh. If this function is activated, the value 00h is clipped to 01h and the value FFh is clipped to FEh. Any other value is passed through.

As shown in Data Areas and Clipping Function, VPIF has two data areas other than TRC: one is blanking data (ancillary data) and the second is video image data. The clipping function can be independently enabled for each data area. Note that field-level control is not supported.

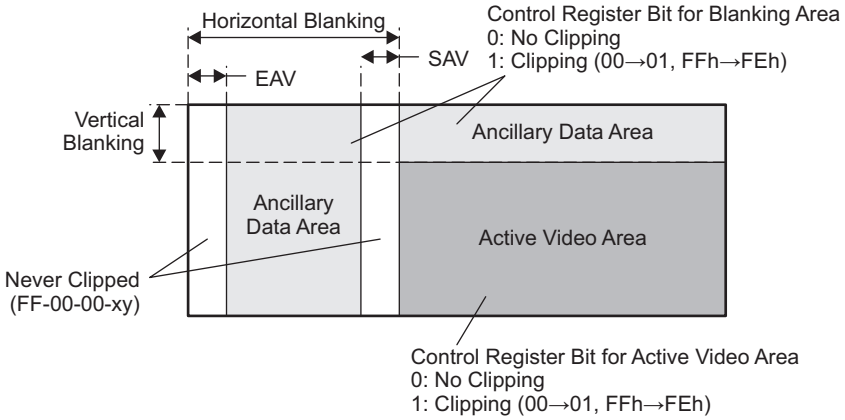
The relationship between register bit assignment and the clipping function is shown in Register Bit Assignment on Clipping Function. Control register bits in Register Bit Assignment on Clipping Function are prepared for both channel 2 and channel 3. Clipping on the channels is independently activated.

Copyright © 2013–2016, Texas Instruments Incorporated

Data Areas and Clipping Function



Register Bit Assignment on Clipping Function



32.2.8 Reset Considerations

The VPIF does not have a software reset. When a hardware reset is asserted, all VPIF registers are set to their default values.

Copyright © 2013–2016, Texas Instruments Incorporated

32.2.9 Initialization

The general procedure for VPIF initialization is:

1. Enable the VPIF in the LPSC.
2. Program the corresponding bits in the DMA size control register (REQSIZE) and the channel *n* sub-picture configuration register (C*n*SUBPIC).
3. Program the Emulation related registers, such as the emulation suspend control register (EMUCTRL) in the VPIF and the emulation suspend source register (SUSPSRC) in the System Module (see Emulation Considerations for more detail). The default values of SUSPSRC are configured to the ARM as the main processor.

32.2.10 Interrupt Support

Interrupt Events and Requests

The VPIF sends the following interrupt events to the CPU:

- FRAME0
- FRAME1
- FRAME2
- FRAME3
- ERROR

The ERROR interrupt is generated for the following reasons:

- For channels 0 or 1:
  - Internal buffer overflow
  - Length of EAV2SAV or SAV2EAV is not the same as the configured value
  - Bit error detected on 4th field of TRC
- For channels 2 or 3:
  - Internal buffer overflow

Note that the VPIF peripheral has no mechanism to reissue interrupts that may have been dropped by the CPU so care should be taken when VPIF interrupts are combined at the system level. In a combined interrupt configuration, the CPU may be busy servicing a VPIF interrupt when another VPIF interrupt arrives. If the interrupt controller has no means to count VPIF interrupts, the CPU will drop subsequent VPIF interrupts until servicing of the first VPIF interrupt is complete.

Field/Frame Interrupts to CPU

This section describes the conditions of the field/frame based interrupt assertion from the VPIF. The VPIF interrupt is designed to make the processor identify the timing for updating the address register of the VPIF module. The timing of the interrupt from the VPIF module is different from the timing of the V-sync. The interrupt is generated when the last data transfer between the VPIF module and the VBUS is finished.

Interrupt Condition

The interrupt signal from the VPIF described in this section means the beginning of a field (or frame). A way to indicate the beginning of a field (or a frame) varies from the function mode; normal YC receive/transmit mode and CCD/CMOS capture mode differ because of the vertical synchronization signal.

The VPIF generates the following events as conditions for interrupt assertion:

- In normal YC receive/transmit mode, EAV on line L1 (and EAV on line L7 in field interrupt for bottom field). Both L1 and L7 are described in Figure 32-4 and Figure 32-5.
- In CCD/CMOS capture mode, starting edge for first line of vertical valid pixel area.

Copyright © 2013–2016, Texas Instruments Incorporated

First Interrupt from VPIF

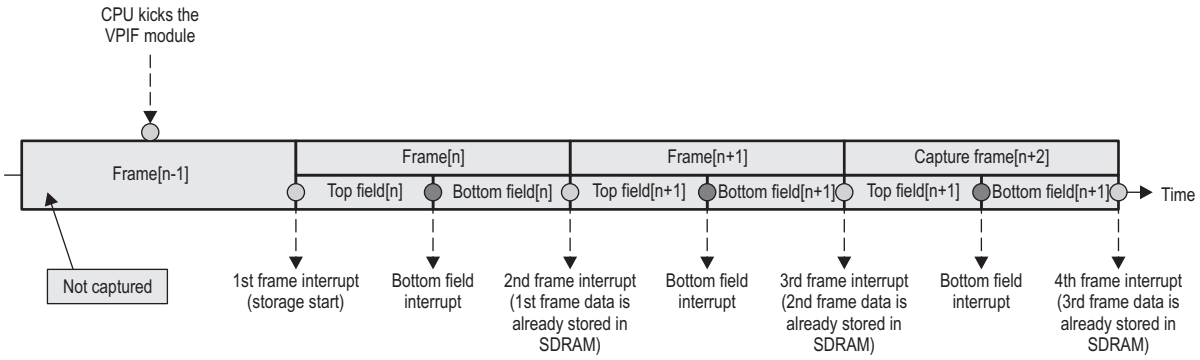
NOTE: The first vertical synchronization signal is defined as the transition from the L10 line to the L11 line in the interlace mode or from the L4 line to the L5 line in the progressive mode.

For raw capture mode, the VPIF immediately starts to capture the data if you enable middle of frame.

In channels 0 and 1, the VPIF starts to capture incoming picture data from the first vertical synchronization signal after the CPU activates the VPIF with register configuration. The interrupt signal from the VPIF is asserted when the vertical synchronization signal is received. So, no incoming data is written in memory when the first interrupt is asserted from the VPIF (Relationship Between the First Interrupt and Incoming Data).

If the CPU uses this interrupt signal not only for the time interval between each video frame but also for the timing to read the stored data from the defined area in memory. Note that the CPU has to ignore the first interrupt signal from the VPIF.

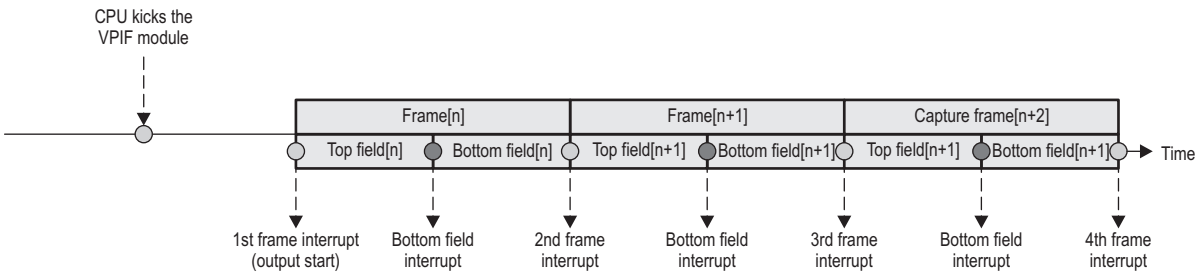
Relationship Between the First Interrupt and Incoming Data



In channels 2 and 3, the VPIF starts to assert displaying picture data after the CPU activates the VPIF with register configuration. The output displaying data is asserted from the VPIF just after the VPIF reads the output data from memory (Relationship Between the First Interrupt and Outgoing Data).

So, any redundant data such as the frame[n - 1] area in Relationship Between the First Interrupt and Incoming Data is not necessary to be prepared. The relationship between the interrupt pulse activation and the register configuration is the same as for capturing data.

Relationship Between the First Interrupt and Outgoing Data



If the Relationship Between the First Interrupt and Outgoing Data described relationship with the CPU activates the VPIF and the incoming data and frame[n] start the L11 line for the interlace mode or the L5 line for the progressive mode, the VPIF starts the data store from frame[n+1] not frame[n]. The first frame interrupt happens after frame[n] (at the 2nd frame interrupt in Relationship Between the First Interrupt and Outgoing Data), because at the start of the incoming data, there is no first vertical synchronization signal.

Copyright © 2013–2016, Texas Instruments Incorporated

Line Interrupts to CPU

In raw data capture mode (CCD/CMOS capture), VPIF will issue line-interval triggered interrupts using the field/frame interrupt signals. The line-interval is programmable in the C0CTRL register, and it uses the FRAME1 interrupt signal.

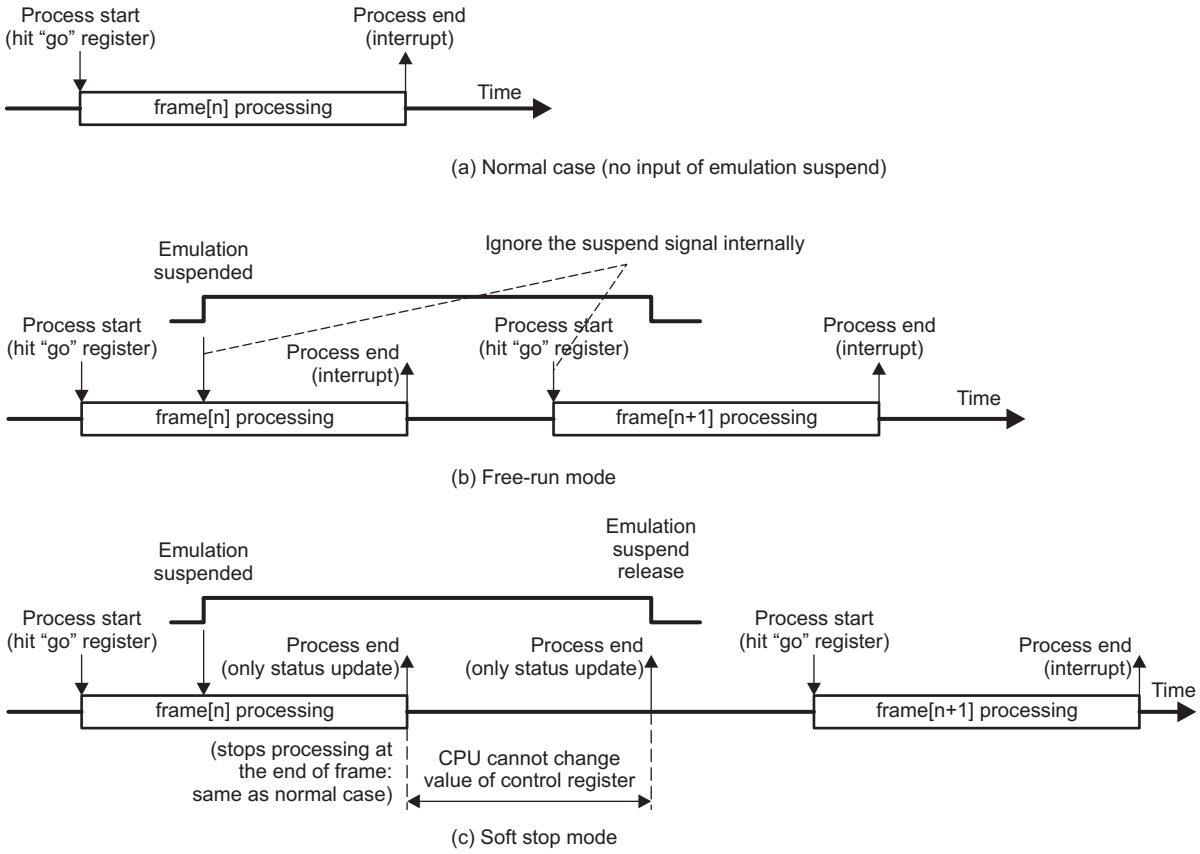
Emulation Considerations

32.2.10.1 Emulation Suspend Mode Support

The VPIF supports the emulation suspend signal from the CPU. The emulation suspend signal (a high signal indicates that the CPU is suspended) is asserted by the CPU when the CPU is halted with a breakpoint or any other reason during debug.

Functional performance, when the emulation suspend signal is received, is defined by the register configuration and is different for each use (receiver and transmitter). Because the VPIF has to process real-time incoming (or outgoing) data that has a unit size of 1 frame, the VPIF can be stalled at the nearest end of the present processing frame when the emulation suspend signal is received. The fundamental performance of emulation suspend is shown in Figure 32-12.

Figure 32-12. Module Performance with Emulation Suspend Signal



Copyright © 2013–2016, Texas Instruments Incorporated

32.2.10.1.1 Receiver (Channels 0 and 1)

The VPIF receives input data from an external video device. The input format is BT.656 or BT.1120. If the emulation suspend comes from the CPU during image processing, the VPIF will first try to run at the end of the present frame and then the VPIF will be stalled (see Figure 32-12 (c)).

The VPIF should only have a software stop mode; the hardware stop mode is not supported. If the CPU changes the register configuration during an emulation suspended period, the new configuration should be validated at the first V-sync after the suspended period.

32.2.10.1.2 Transmitter (Channels 2 and 3)

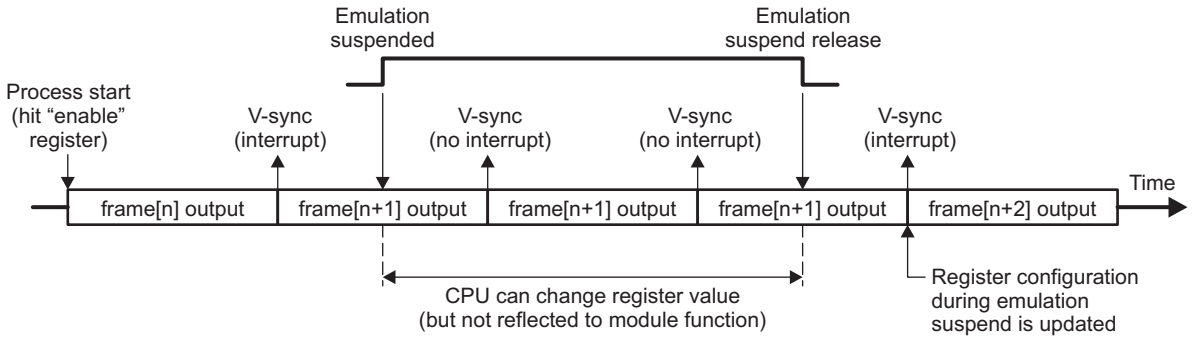
The VPIF transmits output data to an external video device. Source data for this output data is stored in SDRAM. The VPIF needs information about the start address of the source data stored in SDRAM. The output format is BT.656 or BT.1120.

In this mode, any suspend function should not be activated because you would like to see a displayed picture, taking usage of this signal into consideration. From a system's stand point, the VPIF should act in the following sequence if the emulation suspend signal is detected:

1. The VPIF continues processing at the end of the present frame.
2. If the present frame is finished, interrupt assertion is stalled.
3. At phase (b), during the active period of the emulation suspend signal, the VPIF continues reading the same frame data from SDRAM (no register change is reflected to module performance during this period).
4. After the CPU returns back to the normal state, the VPIF performs as usual. New register configurations done during an emulation suspend period are reflected in the functional performance at the first V-sync after deassertion of the suspend signal.

If the CPU changes the register configuration during an emulation suspended period, the new configuration should be validated at the first V-sync after the suspended period. Functional performance of this mode is shown in Figure 32-13.

Figure 32-13. Emulation Suspend Function on Channels 2 and 3 (Transmit)



Copyright © 2013–2016, Texas Instruments Incorporated

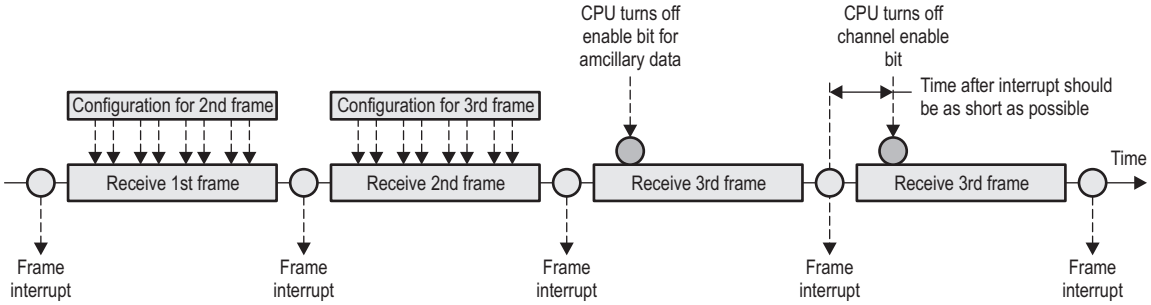
Rules for Module Control

When disabling the VPIF follow the sequence below:

1. Disable ancillary data (HANC and VANC) in the C*n*CTRL control registers.
2. Wait for frame interrupt from the active channel.
3. Clear the channel enable bit of the channel in C*n*CTRL.

The VPIF stops immediately after writing 0 to the C*n*EN bit in the channel *n* control register (C*n*CTRL). Writing to C*n*CTRL takes effect exactly at the same time you write to it, it does not wait to be latched by the VSYNC.

Figure 32-14. Method for Turning off Module Channel



Standard Video Modes

BT.656 Mode

The input clock source of 27 MHZ is used for the source clock of the video output, see Figure 32-15. In BT.656 mode, either channel of the input port on the VPIF is used as the actual input of the data. The VPIF can provide a 27-MHZ output to an external PLL device that provides the audio clock synchronized to the video output clock.

Parameter Configuration for BT.656 Mode

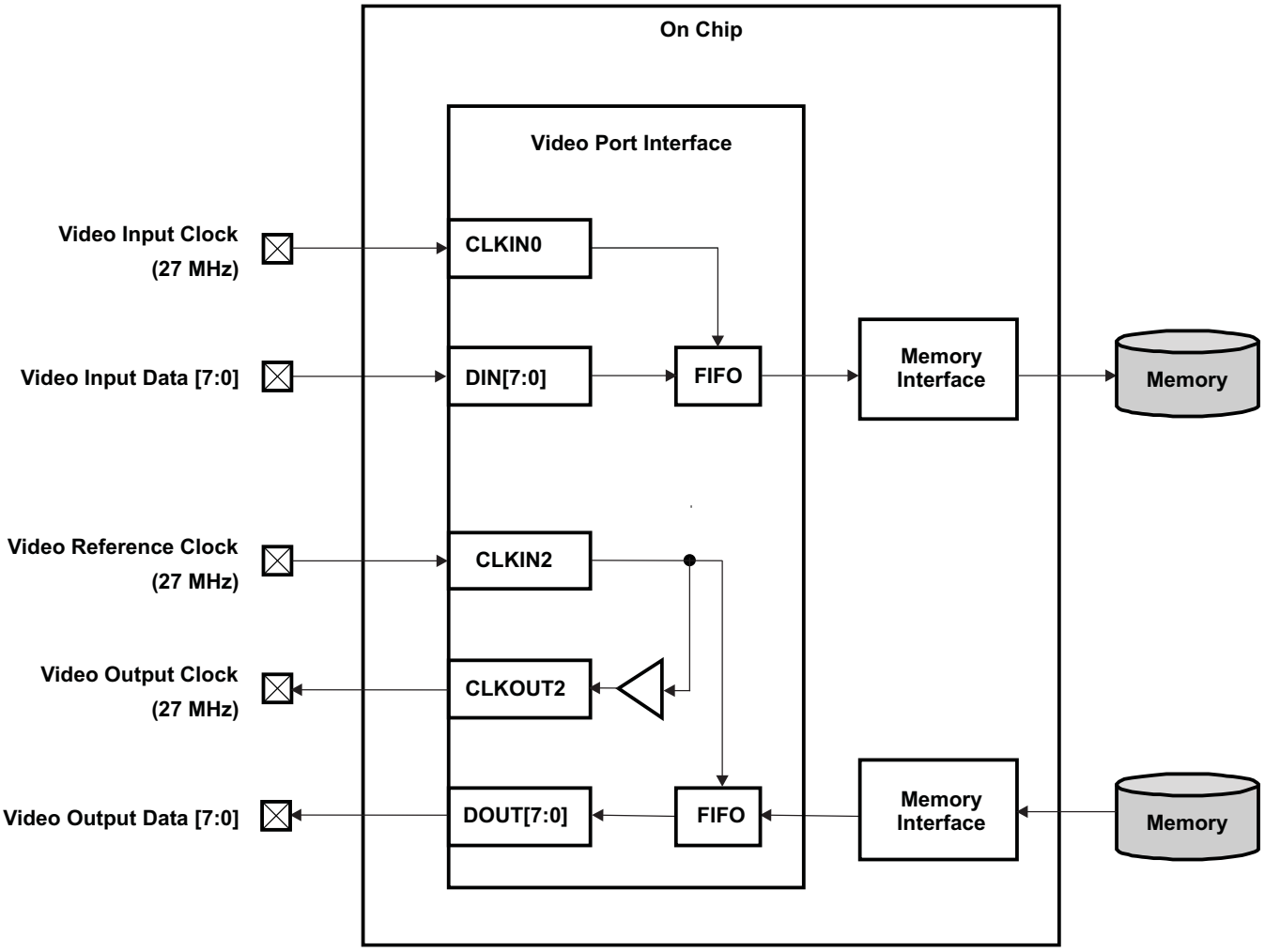
The configuration for each register in BT.656 mode is shown in Register Configuration on BT.656 Input/Output (Unit Size = Byte in unsigned).

Register Configuration on BT.656 Input/Output (Unit Size = Byte in unsigned)

Parameter	Register	Bit Name	NTSC	PAL
EAV2SAV	C*n*HCFG	EAV2SAV	268	280
SAV2EAV	C*n*HCFG	SAV2EAV	1440	1440
Vertical frame size	C*n*VSIZE	VSIZE	525	625
L1	C*n*VCFG0	L1	4	1
L3	C*n*VCFG0	L3	20	23
L5	C*n*VCFG1	L5	264	311
L7	C*n*VCFG1	L7	266	313
L9	C*n*VCFG2	L9	283	336
L11	C*n*VCFG2	L11	1	624

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 32-15. Clock Control on Video Input and Output with SDTV Encoding



Copyright © 2013–2016, Texas Instruments Incorporated

BT.1120 Mode

The BT.1120 format is necessary to support HDTV input and output for the VPIF. The BT.1120 mode requires an input clock source of 74.25 MHZ. Two VPIF channels are necessary to receive (encode) an input image and to display (local decode) the output image. The functional image and clock control is shown in Figure 32-16. In this case, VPIF channels 0 and 1 are used for input and VPIF channels 2 and 3 are used for output.

Parameter Configuration for BT.1120 Mode (1125/60/2:1 and 1250/50/2:1 system)

The configuration for each register in BT.1120 mode (1125/60/2:1 and 1250/50/2:1 system) is shown in Register Configuration on BT.1120 (1125/60/2:1 and 1250/50/2:1 System) Input/Output (Unit Size = Byte in unsigned).

Register Configuration on BT.1120 (1125/60/2:1 and 1250/50/2:1 System) Input/Output (Unit Size = Byte in unsigned)

Parameter	Register	Bit Name	NTSC	PAL
EAV2SAV	C*n*HCFG	EAV2SAV	272	376
SAV2EAV	C*n*HCFG	SAV2EAV	1920	1920
Vertical frame size	C*n*VSIZE	VSIZE	1125	1250
L1	C*n*VCFG0	L1	1	1
L3	C*n*VCFG0	L3	41	45
L5	C*n*VCFG1	L5	558	621
L7	C*n*VCFG1	L7	564	626
L9	C*n*VCFG2	L9	603	670
L11	C*n*VCFG2	L11	1121	1246

Parameter Configuration for BT.1120 Mode (1080-30p system)

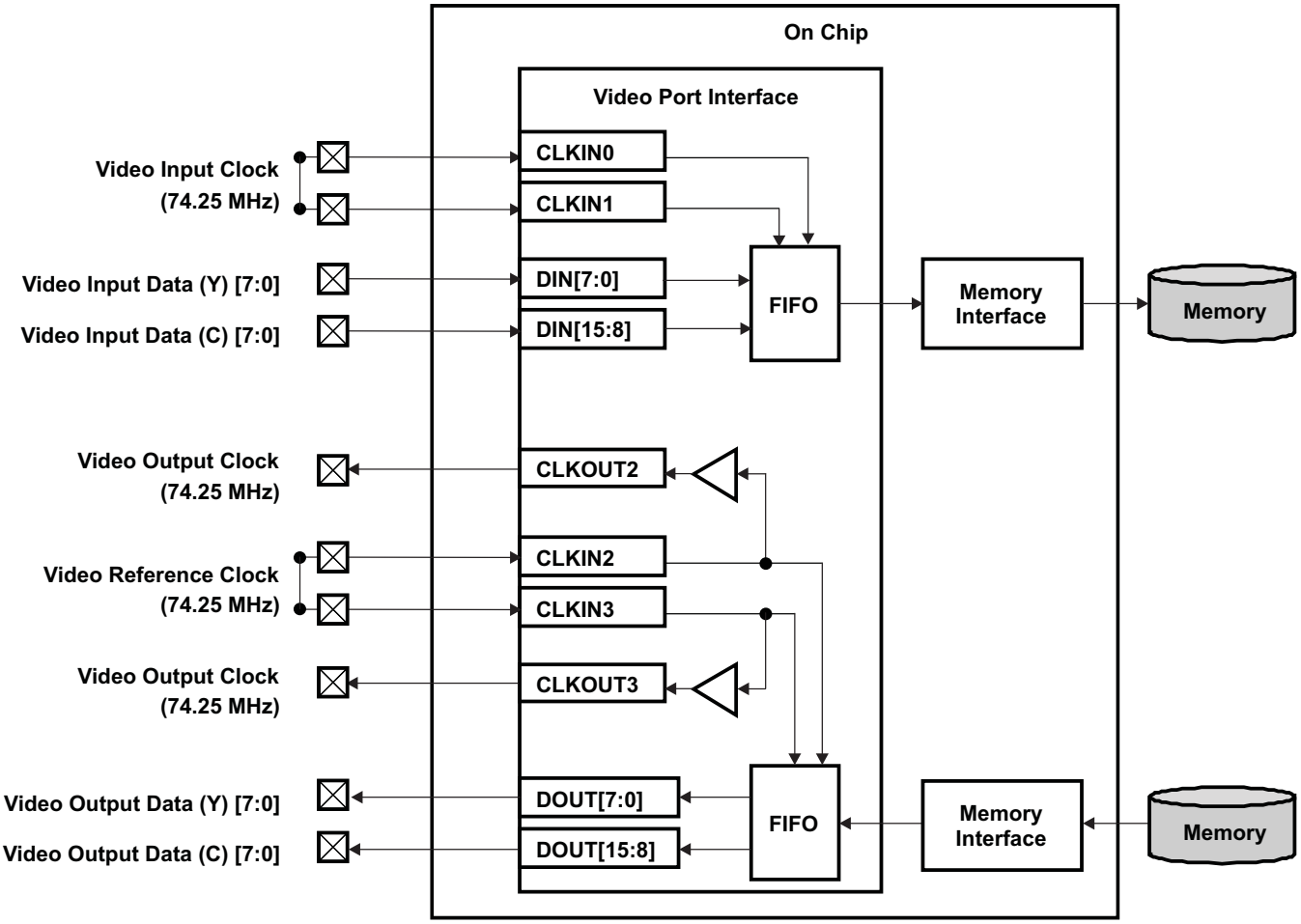
The configuration for each register in BT.1120 mode (1080-30p system) is shown in Register Configuration on BT.1120 (1080-30p System) Input/Output (Unit Size = Byte in unsigned).

Register Configuration on BT.1120 (1080-30p System) Input/Output (Unit Size = Byte in unsigned)

Parameter	Register	Bit Name	Value
EAV2SAV	C*n*HCFG	EAV2SAV	272
SAV2EAV	C*n*HCFG	SAV2EAV	1920
Vertical frame size	C*n*VSIZE	VSIZE	1125
L1	C*n*VCFG0	L1	1
L3	C*n*VCFG0	L3	42
L5	C*n*VCFG1	L5	1122

Copyright © 2013–2016, Texas Instruments Incorporated

Figure 32-16. Clock Control on Video Input and Output with HDTV Encoding



Copyright © 2013–2016, Texas Instruments Incorporated

SMPTE 296M Mode

The SMPTE 296M mode requires the same clock control for the VPIF module as in BT.656 and BT.1120 modes. The video input clock source is 74.25 MHZ. Two VPIFs are necessary to receive an input image and to display the output image. The functional image and clock control is shown in Figure 32-17. In this case, VPIF channels 0 and 1 are used for input and VPIF channels 2 and 3 are used for output.

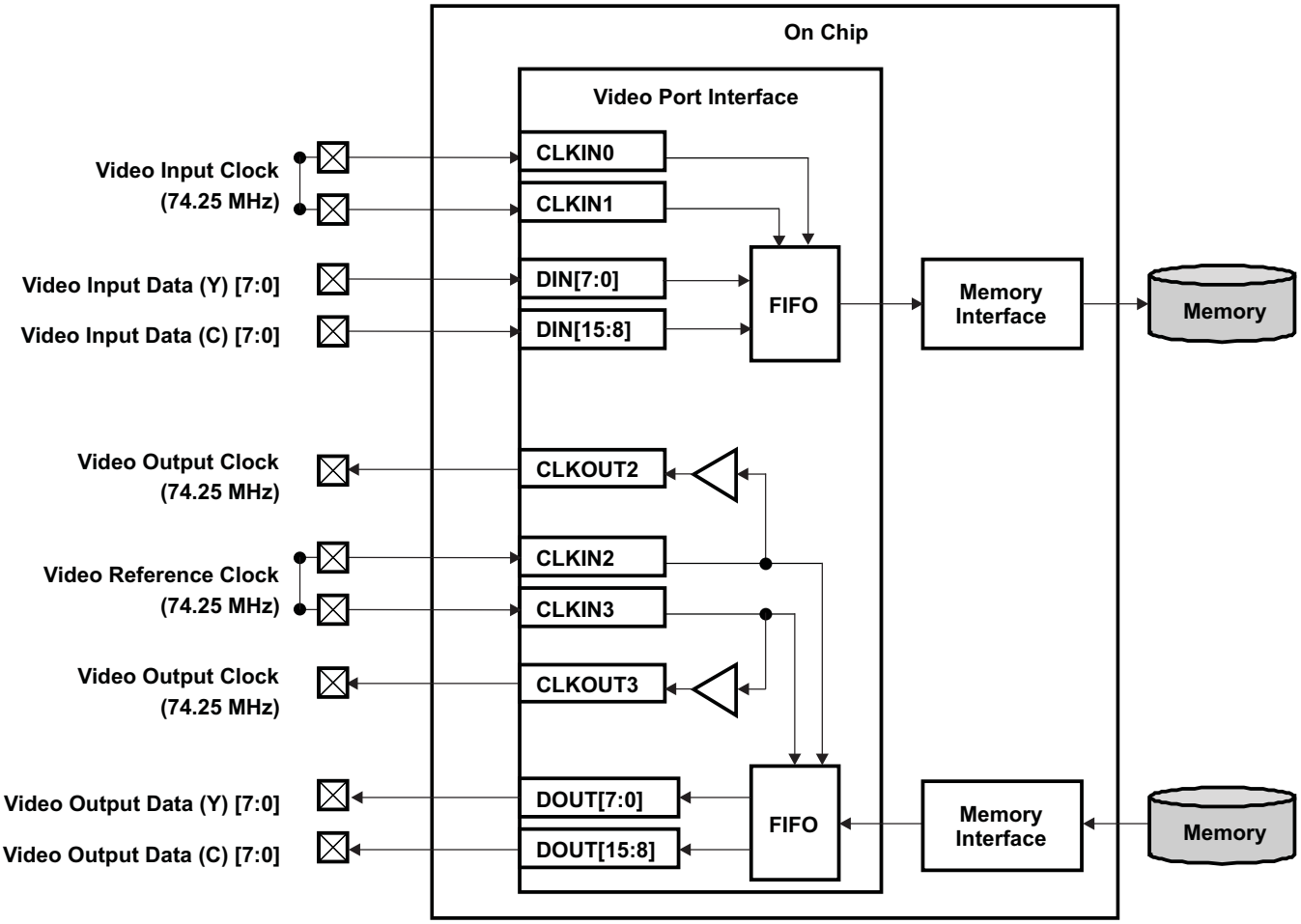
Parameter Configuration for SMPTE 296M Mode

The configuration for each register in SMPTE 296M mode is shown in Register Configuration on SMPTE 296M Input/Output (Unit Size = Byte in unsigned).

Register Configuration on SMPTE 296M Input/Output (Unit Size = Byte in unsigned)

Parameter	Register	Bit Name	Value
EAV2SAV	C*n*HCFG	EAV2SAV	362
SAV2EAV	C*n*HCFG	SAV2EAV	1280
Vertical frame size	C*n*VSIZE	VSIZE	750
L1	C*n*VCFG0	L1	1
L3	C*n*VCFG0	L3	26
L5	C*n*VCFG1	L5	746

Figure 32-17. Clock Control on Video Input and Output with HDTV Encoding



Copyright © 2013–2016, Texas Instruments Incorporated

Raw Capture Mode

The raw capture mode requires the same input clock for VPIF channel 0 and channel 1 (clock skew has to be aligned). A single clock signal should be connected to both VP_CLKIN0 and VP_CLKIN1.

Parameter Configuration for Raw Capture Mode

When the VPIF sets the raw capture mode, the VPIF recognizes the image size automatically using the H/V valid signals. You do not need to configure the image size in the following registers:

- channel *n* horizontal data size configuration register (C*n*HCFG)
- channel *n* vertical data size configuration 0 register (C*n*VCFG0)
- channel *n* vertical data size configuration 1 register (C*n*VCFG1)
- channel *n* vertical image size register (C*n*VSIZE)

However, you still need to configure the storage memory address control related registers.

32.3 Registers

All register values should be configured before you set the C*n*EN bit in the channel *n* control register (C*n*CTRL) to 1. Also note that all register values except the C*n*EN bit in C*n*CTRL are detected by the first falling edge of the vertical synchronization signal on each channel as shown in Relationship Between Register and Data Access.

Table 32-5 lists the memory-mapped registers for the video port interface (VPIF). See the device-specific data manual for the memory address of these registers.

Relationship Between Register and Data Access

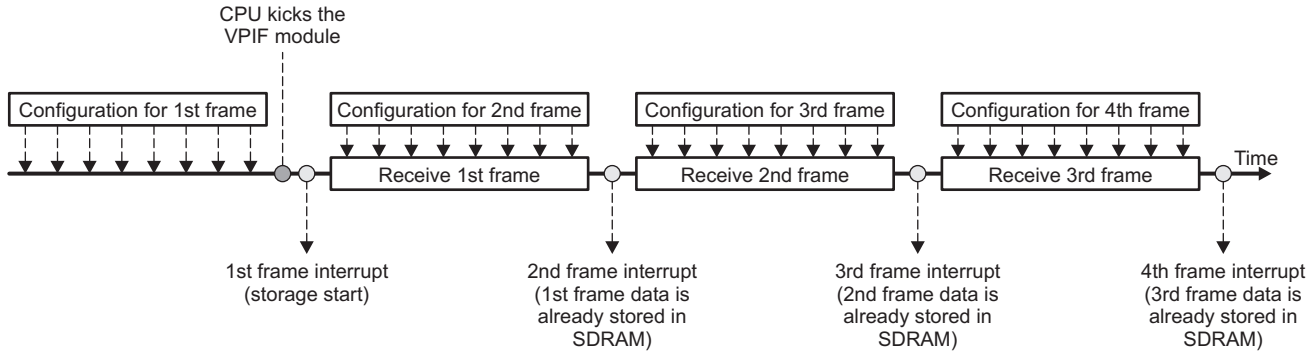


Table 32-5. Video Port Interface (VPIF) Registers

Offset	Acronym	Register Description	Section
0h	REVID	VPIF Revision ID register	Section 32.3.1
4h	C0CTRL	Channel 0 control register	Section 32.3.2
8h	C1CTRL	Channel 1 control register	Section 32.3.3
Ch	C2CTRL	Channel 2 control register	Section 32.3.4
10h	C3CTRL	Channel 3 control register	Section 32.3.5
20h	INTEN	Interrupt enable register	Section 32.3.6
24h	INTSET	Interrupt enable set register	Section 32.3.7
28h	INTCLR	Interrupt enable clear register	Section 32.3.8
2Ch	INTSTAT	Interrupt status register	Section 32.3.9
30h	INTSTATCLR	Interrupt status clear register	Section 32.3.10
34h	EMUCTRL	Emulation suspend control register	Section 32.3.11
38h	REQSIZE	DMA size control register	Section 32.3.12

Copyright © 2013–2016, Texas Instruments Incorporated

Table 32-5. Video Port Interface (VPIF) Registers (continued)

Offset	Acronym	Register Description	Section
		Channel 0	
40h	C0TLUMA	Channel 0 top field luminance address register	Section 32.3.13
44h	C0BLUMA	Channel 0 bottom field luminance address register	Section 32.3.14
48h	C0TCHROMA	Channel 0 top field chrominance address register	Section 32.3.15
4Ch	C0BCHROMA	Channel 0 bottom field chrominance address register	Section 32.3.16
50h	C0THANC	Channel 0 top field horizontal ancillary address register	Section 32.3.17
54h	C0BHANC	Channel 0 bottom field horizontal ancillary address register	Section 32.3.18
58h	C0TVANC	Channel 0 top field vertical ancillary address register	Section 32.3.19
5Ch	C0BVANC	Channel 0 bottom field vertical ancillary address register	Section 32.3.20
60h	Reserved	Reserved	
64h	C0IMGOFFSET	Channel 0 image data address offset register	Section 32.3.21
68h	C0HANCOFFSET	Channel 0 horizontal ancillary address offset register	Section 32.3.22
6Ch	C0HCFG	Channel 0 horizontal data size configuration register	Section 32.3.23
70h	C0VCFG0	Channel 0 vertical data size configuration 0 register	Section 32.3.24
74h	C0VCFG1	Channel 0 vertical data size configuration 1 register	Section 32.3.25
78h	C0VCFG2	Channel 0 vertical data size configuration 2 register	Section 32.3.26
7Ch	C0VSIZE	Channel 0 vertical image size register	Section 32.3.27
		Channel 1	
80h	C1TLUMA	Channel 1 top field luminance address register	Section 32.3.13
84h	C1BLUMA	Channel 1 bottom field luminance address register	Section 32.3.14
88h	C1TCHROMA	Channel 1 top field chrominance address register	Section 32.3.15
8Ch	C1BCHROMA	Channel 1 bottom field chrominance address register	Section 32.3.16
90h	C1THANC	Channel 1 top field horizontal ancillary address register	Section 32.3.17
94h	C1BANC	Channel 1 bottom field horizontal ancillary address register	Section 32.3.18
98h	C1TVANC	Channel 1 top field vertical ancillary address register	Section 32.3.19
9Ch	C1BVANC	Channel 1 bottom field vertical ancillary address register	Section 32.3.20
A0h	Reserved	Reserved	
A4h	C1IMGOFFSET	Channel 1 image data address offset register	Section 32.3.21
A8h	C1HANCOFFSET	Channel 1 horizontal ancillary address offset register	Section 32.3.22
ACh	H1HCFG	Channel 1 horizontal data size configuration register	Section 32.3.23
B0h	C1VCFG1	Channel 1 vertical data size configuration 0 register	Section 32.3.24
B4h	C1VCFG1	Channel 1 vertical data size configuration 1 register	Section 32.3.25
B8h	C1VCFG2	Channel 1 vertical data size configuration 2 register	Section 32.3.26
BCh	C1VSIZE	Channel 1 vertical image size register	Section 32.3.27
		Channel 2	
C0h	C2TLUMA	Channel 2 top field luminance address register	Section 32.3.13
C4h	C2BLUMA	Channel 2 bottom field luminance address register	Section 32.3.14
C8h	C2TCHROMA	Channel 2 top field chrominance address register	Section 32.3.15
CCh	C2BCHROMA	Channel 2 bottom field chrominance address register	Section 32.3.16
D0h	C2THANC	Channel 2 top field horizontal ancillary address register	Section 32.3.17
D4h	C2BHANC	Channel 2 bottom field horizontal ancillary address register	Section 32.3.18
D8h	C2TVANC	Channel 2 top field vertical ancillary address register	Section 32.3.19
DCh	C2BVANC	Channel 2 bottom field vertical ancillary address register	Section 32.3.20
E0h	Reserved	Reserved	
E4h	C2IMGOFFSET	Channel 2 image data address offset register	Section 32.3.21
E8h	C2HANCOFFSET	Channel 2 horizontal ancillary address offset register	Section 32.3.22
ECh	C2HCFG	Channel 2 horizontal data size configuration register	Section 32.3.28

Copyright © 2013–2016, Texas Instruments Incorporated

Table 32-5. Video Port Interface (VPIF) Registers (continued)

Offset	Acronym	Register Description	Section
F0h	C2VCFG0	Channel 2 vertical data size configuration 0 register	Section 32.3.29
F4h	C2VCFG1	Channel 2 vertical data size configuration 1 register	Section 32.3.30
F8h	C2VCFG2	Channel 2 vertical data size configuration 2 register	Section 32.3.31
FCh	C2VSIZE	Channel 2 vertical image size register	Section 32.3.32
100h	C2THANCPOS	Channel 2 top field horizontal ancillary data insertion start position register	Section 32.3.33
104h	C2THANCSIZE	Channel 2 top field horizontal ancillary data size register	Section 32.3.34
108h	C2BHANCPOS	Channel 2 bottom field horizontal ancillary data insertion start position register	Section 32.3.35
10Ch	C2BHANCSIZE	Channel 2 bottom field horizontal ancillary data size register	Section 32.3.36
110h	C2TVANCPOS	Channel 2 top field vertical ancillary data insertion start position register	Section 32.3.37
114h	C2TVANCSIZE	Channel 2 top field vertical ancillary data size register	Section 32.3.38
118h	C2BVANCPOS	Channel 2 bottom field vertical ancillary data insertion start position register	Section 32.3.39
11Ch	C2BVANCSIZE	Channel 2 bottom field vertical ancillary data size register	Section 32.3.40
		Channel 3	
140h	C3TLUMA	Channel 3 top field luminance address register	Section 32.3.13
144h	C3BLUMA	Channel 3 bottom field luminance address register	Section 32.3.14
148h	C3TCHROMA	Channel 3 top field chrominance address register	Section 32.3.15
14Ch	C3BCHROMA	Channel 3 bottom field chrominance address register	Section 32.3.16
150h	C3THANC	Channel 3 top field horizontal ancillary address register	Section 32.3.17
154h	C3BHANC	Channel 3 bottom field horizontal ancillary address register	Section 32.3.18
158h	C3TVANC	Channel 3 top field vertical ancillary address register	Section 32.3.19
15Ch	C3BVANC	Channel 3 bottom field vertical ancillary address register	Section 32.3.20
160h	Reserved	Reserved	
164h	C3IMGOFFSET	Channel 3 image data address offset register	Section 32.3.21
168h	C3HANCOFFSET	Channel 3 horizontal ancillary address offset register	Section 32.3.22
16Ch	C3HCFG	Channel 3 horizontal data size configuration register	Section 32.3.28
170h	C3VCFG0	Channel 3 vertical data size configuration 0 register	Section 32.3.29
174h	C3VCFG1	Channel 3 vertical data size configuration 1 register	Section 32.3.30
178h	C3VCFG2	Channel 3 vertical data size configuration 2 register	Section 32.3.31
17Ch	C3VSIZE	Channel 3 vertical image size register	Section 32.3.32
180h	C3THANCPOS	Channel 3 top field horizontal ancillary data insertion start position register	Section 32.3.33
184h	C3THANCSIZE	Channel 3 top field horizontal ancillary data size register	Section 32.3.34
188h	C3BHANCPOS	Channel 3 bottom field horizontal ancillary data insertion start position register	Section 32.3.35
18Ch	C3BHANCSIZE	Channel 3 bottom field horizontal ancillary data size register	Section 32.3.36
190h	C3TVANCPOS	Channel 3 top field vertical ancillary data insertion start position register	Section 32.3.37
194h	C3TVANCSIZE	Channel 3 top field vertical ancillary data size register	Section 32.3.38
198h	C3BVANCPOS	Channel 3 bottom field vertical ancillary data insertion start position register	Section 32.3.39
19Ch	C3BVANCSIZE	Channel 3 bottom field vertical ancillary data size register	Section 32.3.40

Copyright © 2013–2016, Texas Instruments Incorporated

32.3.1 VPIF Revision Register ID (REVID)

The VPIF peripheral Revision ID Register (REVID) is shown in Figure 32-18 and described in Table 32-6.

Figure 32-18. VPIF Revision ID Register (REVID)

31	0
REV	
R-4C08h 0A01h	

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

Table 32-6. VPIF Revision ID Register (REVID) Field Descriptions

Bit	Field	Value	Description
31-0	REV	4C08 0A01h	VPIF Revision

32.3.2 Channel 0 Control Register (C0CTRL)

The channel 0 control register (C0CTRL) is shown in Figure 32-19 and described in Table 32-7.

Figure 32-19. Channel 0 Control Register (C0CTRL)

31	30	29	28	27	16
CLKEDGE	Rsvd	DATAWIDTH		INTLINE	
R/W-0	R-0	R/W-0		R/W-0	

15	14	13	12	11	10	9	8
FIDINV	VVINV	HVINV	FIELDFRAME	Reserved	INTRPROG	VANC	HANC
R/W-0	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
INTFRAME		FID	Reserved	YCMUX	CAPMODE	Reserved	CHANEN
R/W-0		R-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 32-7. Channel 0 Control Register (C0CTRL) Field Descriptions

Bit	Field	Value	Description
31	CLKEDGE		Clock edge control.
		0	Data is captured on rising edge of input clock.
		1	Data is captured on falling edge of input clock.
30	Reserved	0	Reserved
29-28	DATAWIDTH	0-3h	Data bit width. The DATAWIDTH bit is only used with the CCD/CMOS data capture mode (CAPMODE = 1h).
		0	8 bits/pixel
		1h	10 bits/pixel
		2h	12 bits/pixel
		3h	Reserved
27-16	INTLINE	0-FFFh	Line interrupt interval for CCD/CMOS capture mode. In CCD/CMOS capture mode, the frame interrupt signal is asserted to the CPU at the end of INTLINE interval lines. A value of 0 is invalid in CCD/CMOS capture mode. The INTLINE bit is only used with the CCD/CMOS capture mode (CAPMODE = 1h). Line interrupts are asserted through the FRAME1 signal.
15	FIDINV		Field ID polarity inverting control bit. The FIDINV bit is only used with the CCD/CMOS data capture mode (CAPMODE = 1h).
		0	No inversion.
		1	Invert incoming field ID signal inside the VPIF.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 32-7. Channel 0 Control Register (C0CTRL) Field Descriptions (continued)

Bit	Field	Value	Description
14	VVINV		Vertical pixel valid signal polarity control. The VVINV bit is only used with the CCD/CMOS data capture mode (CAPMODE = 1h).
		0	No inversion.
		1	Invert incoming vertical pixel valid signal inside the VPIF.
13	HVINV		Horizontal pixel valid signal polarity control. The HVINV bit is only used with the CCD/CMOS data capture mode (CAPMODE = 1h).
		0	No inversion.
		1	Invert incoming horizontal pixel valid signal inside the VPIF.
12	FIELDFRAME		Memory storage mode of input picture. Note that for progressive video, only the frame-based storage method is supported.
		0	Field-based storage
		1	Frame-based storage
11	Reserved	0	Reserved
10	INTRPROG		Output display format.
		0	Interlaced
		1	Progressive
9	VANC		Channel 0 vertical ancillary data enable. Ancillary data is only supported for BT byte streams.
		0	Vertical ancillary data is disabled.
		1	Vertical ancillary data is enabled.
8	HANC		Channel 0 horizontal ancillary data enable. Ancillary data is only supported for BT byte streams.
		0	Horizontal ancillary data is disabled.
		1	Horizontal ancillary data is enabled.
7-6	INTFRAME	0-3h	Channel 0 frame interrupt to CPU for BT/YC and CCD/CMOS capture modes.
		0	Top field V-sync only.
		1h	Bottom field V-sync only. Interlaced format only.
		2h	Top and bottom field V-sync. Interlaced format only.
		3h	Reserved
5	FID		Channel 0 field identification. This bit indicates the active field ID when the FRAME0 interrupt is asserted from VPIF to the CPU.
		0	Top field
		1	Bottom field
4	Reserved	0	Reserved
3	YCMUX		Channel 0 input data format.
		0	Channel 0 Y/C non-multiplexed mode.
		1	Channel 0 Y/C multiplexed mode.
2	CAPMODE		Channel 0 data capture mode. This value must match the channel 1 data capture mode (CAPMODE bit in C1CTRL).
		0	BT video (Y/C format) capture mode.
		1	CCD/CMOS data capture mode.
1	Reserved	0	Reserved
0	CHANEN		VPIF channel 0 enable.
		0	Channel 0 is disabled.
		1	Channel 0 is enabled.

Copyright © 2013–2016, Texas Instruments Incorporated

32.3.3 Channel 1 Control Register (C1CTRL)

The channel 1 control register (C1CTRL) is shown in Figure 32-20 and described in Table 32-8.

Figure 32-20. Channel 1 Control Register (C1CTRL)

31	30	16
CLKEDGE	Reserved	
R/W-0	R-0	

15	11	10	9	8
Reserved		INTRPROG	VANC	HANC
R-0		R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
INTFRAME		FID	Reserved	YCMUX	CAPMODE	Reserved	CHANEN
R/W-0		R-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 32-8. Channel 1 Control Register (C1CTRL) Field Descriptions

Bit	Field	Value	Description
31	CLKEDGE		Clock edge control.
		0	Data is captured on rising edge of input clock.
		1	Data is captured on falling edge of input clock.
30-11	Reserved	0	Reserved
10	INTRPROG		Output display format.
		0	Interlaced
		1	Progressive
9	VANC		Channel 1 vertical ancillary data enable. Ancillary data is only supported for BT byte streams.
		0	Vertical ancillary data is disabled.
		1	Vertical ancillary data is enabled.
8	HANC		Channel 1 horizontal ancillary data enable. Ancillary data is only supported for BT byte streams.
		0	Horizontal ancillary data is disabled.
		1	Horizontal ancillary data is enabled.
7-6	INTFRAME	0-3h	Channel 1 frame interrupt to CPU for BT/YC and CCD/CMOS capture modes.
		0	Top field V-sync only.
		1h	Bottom field V-sync only. Interlaced format only.
		2h	Top and bottom field V-sync. Interlaced format only.
		3h	Reserved
5	FID		Channel 1 field identification. This bit indicates the active field ID when the FRAME1 interrupt is asserted from VPIF to CPU.
		0	Top field.
		1	Bottom field.
4	Reserved	0	Reserved
3	YCMUX		Channel 1 input data format.
		0	Channel 1 Y/C non-multiplexed mode.
		1	Channel 0 Y/C multiplexed mode.
2	CAPMODE		Channel 1 data capture mode. This value must match the channel 0 data capture mode (CAPMODE bit in C0CTRL).
		0	BT video (Y/C format) capture mode.
		1	CCD/CMOS data capture mode.
1	Reserved	0	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

Table 32-8. Channel 1 Control Register (C1CTRL) Field Descriptions (continued)

Bit	Field	Value	Description
0	CHANEN		VPIF channel 1 enable.
		0	Channel 1 is disabled.
		1	Channel 1 is enabled.

32.3.4 Channel 2 Control Register (C2CTRL)

The channel 2 control register (C2CTRL) is shown in Figure 32-21 and described in Table 32-9.

Figure 32-21. Channel 2 Control Register (C2CTRL)

31	30						16
CLKEDGE	Reserved						
R/W-0	R-0						

15	14	13	12	11	10	9	8
Reserved	CLIPANC	CLIPVID	FIELDFRAME	INTRPROG	PIXEL	VANC	HANC
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
INTFRAME		FID	Reserved	YCMUX	Reserved	CLKEN	CHANEN
R/W-0		R-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 32-9. Channel 2 Control Register (C2CTRL) Field Descriptions

Bit	Field	Value	Description
31	CLKEDGE		Clock edge control
		0	Data is output on rising edge of output clock.
		1	Data is output on falling edge of output clock. The output clock controlled by the CLKEN bit will be inverted.
30-15	Reserved	0	Reserved
14	CLIPANC		Activates clipping function of output data in the blanking region for channel 2.
		0	Clipping in the blanking region for channel 2 is disabled.
		1	Clipping in the blanking region for channel 2 is enabled. (0 is clipped to 1; FFh is clipped to FEh)
13	CLIPVID		Activates clipping function of output data in the active region for channel 2.
		0	Clipping in the active region for channel 2 is disabled.
		1	Clipping in the active region for channel 2 is enabled. (0 is clipped to 1; FFh is clipped to FEh)
12	FIELDFRAME		Memory storage mode of output picture. Note that this bit is effective only when the INTRPROG bit is 0 (interlaced). In progressive mode, only the frame-based storage method is supported.
		0	Field-based storage
		1	Frame-based storage
11	INTRPROG		Output display format.
		0	Interlaced
		1	Progressive
10	PIXEL		Enables the display of video pixels.
		0	Pixel data is disabled. Blank pixels are displayed (Y = 10h and C = 80h).
		1	Pixel data is enabled. Pixel data from memory is displayed.
9	VANC		Channel 2 vertical ancillary data enable. Ancillary data is only supported for BT byte streams.
		0	Vertical ancillary data is disabled.
		1	Vertical ancillary data is enabled.

Copyright © 2013–2016, Texas Instruments Incorporated

Table 32-9. Channel 2 Control Register (C2CTRL) Field Descriptions (continued)

Bit	Field	Value	Description
8	HANC		Channel 2 horizontal ancillary data enable. Ancillary data is only supported for BT byte streams.
		0	Horizontal ancillary data is disabled.
		1	Horizontal ancillary data is enabled.
7-6	INTFRAME	0-3h	Channel 2 frame level interrupt to CPU.
		0	Top field V-sync only.
		1h	Bottom field V-sync
		2h	Top and bottom field.
		3h	Reserved
5	FID		Channel 2 field identification. This bit indicates the active field ID when the FRAME2 interrupt is asserted from the VPIF to CPU.
		0	Top field.
		1	Bottom field.
4	Reserved	0	Reserved
3	YCMUX		Channel 2 output data format.
		0	Channel 2 Y/C non-multiplexed mode.
		1	Channel 2 Y/C multiplexed mode (both Y and C are in the byte stream).
2	Reserved	0	Reserved
1	CLKEN		Clock output enable control for VPIF channel 2. This bit should be set before enabling the channel and it should be cleared after deactivating the channel.
		0	VPIF channel 2 clock is disabled.
		1	VPIF channel 2 clock is enabled.
0	CHANEN		VPIF channel 2 enable.
		0	Channel 2 is disabled.
		1	Channel 2 is enabled.

Copyright © 2013–2016, Texas Instruments Incorporated

32.3.5 Channel 3 Control Register (C3CTRL)

The channel 3 control register (C3CTRL) is shown in Figure 32-22 and described in Table 32-10.

Figure 32-22. Channel 3 Control Register (C3CTRL)

31	30	16
CLKEDGE	Reserved	
R/W-0	R-0	

15	14	13	12	11	10	9	8
Reserved	CLIPANC	CLIPVID	Reserved	INTRPROG	PIXEL	VANC	HANC
R-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
INTFRAME		FID	Reserved	YCMUX	Reserved	CLKEN	CHANEN
R/W-0		R-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 32-10. Channel 3 Control Register (C3CTRL) Field Descriptions

Bit	Field	Value	Description
31	CLKEDGE		Clock edge control
		0	Data is output on rising edge of output clock.
		1	Data is output on falling edge of output clock. The output clock controlled by the CLKEN bit will be inverted.
30-15		0	Reserved
14	CLIPANC		Activates clipping function of output data in the blanking region for channel 3.
		0	Clipping in the blanking region for channel 3 is disabled.
		1	Clipping in the blanking region for channel 3 is enabled. (0 is clipped to 1; FFh is clipped to FEh)
13	CLIPVID		Activates clipping function of output data in the active region for channel 3.
		0	Clipping in the active region for channel 3 is disabled.
		1	Clipping in the active region for channel 3 is enabled. (0 is clipped to 1; FFh is clipped to FEh)
12	Reserved	0	Reserved
11	INTRPROG		Output display format.
		0	Interlaced
		1	Progressive
10	PIXEL		Enables the display of video pixels.
		0	Pixel data is disabled. Blank pixels are displayed (Y = 10h and C = 80h).
		1	Pixel data is enabled. Pixel data from memory is displayed.
9	VANC		Channel 3 vertical ancillary data enable. Ancillary data is only supported for BT byte streams.
		0	Vertical ancillary data is disabled.
		1	Vertical ancillary data is enabled.
8	HANC		Channel 3 horizontal ancillary data enable. Ancillary data is only supported for BT byte streams.
		0	Horizontal ancillary data is disabled.
		1	Horizontal ancillary data is enabled.
7-6	INTFRAME	0-3h	Channel 3 frame level interrupt to CPU.
		0	Top field V-sync only.
		1h	Bottom field V-sync.
		2h	Top and bottom field.
		3h	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

Table 32-10. Channel 3 Control Register (C3CTRL) Field Descriptions (continued)

Bit	Field	Value	Description
5	FID		Channel 3 field identification. This bit indicates the active field ID when the FRAME3 interrupt is asserted from the VPIF to CPU.
		0	Top field.
		1	Bottom field.
4	Reserved	0	Reserved
3	YCMUX		Channel 3 output data format.
		0	Channel 3 Y/C non-multiplexed mode.
		1	Channel 3 Y/C multiplexed mode (both Y and C are in the byte stream).
2	Reserved	0	Reserved
1	CLKEN		Clock output enable control for VPIF channel 3. This bit should be set before enabling the channel and it should be cleared after deactivating the channel.
		0	VPIF channel 3 clock is disabled.
		1	VPIF channel 3 clock is enabled.
0	CHANEN		VPIF channel 3 enable.
		0	Channel 3 is disabled.
		1	Channel 3 is enabled.

Copyright © 2013–2016, Texas Instruments Incorporated

32.3.6 Interrupt Enable Register (INTEN)

The interrupt enable register (INTEN) is shown in Figure 32-23 and described in Table 32-11.

Figure 32-23. Interrupt Enable Register (INTEN)

31	16
Reserved	
R-0	

15	8
Reserved	
R-0	

7	4	3	2	1	0
Reserved	ERROR	FRAME3	FRAME2	FRAME1	FRAME0
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 32-11. Interrupt Enable Register (INTEN) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4	ERROR		VPIF error interrupt enable. The error interrupt is asserted when an problem is detected in any input/output byte stream data.
		0	Error interrupt is disabled.
		1	Error interrupt is enabled. The ERROR bit in the INTSET register must also be set to activate the ERROR interrupt.
3	FRAME3		Channel 3 frame interrupt enable. Enables the frame interrupt from the VPIF to CPU.
		0	Channel 3 frame interrupt is disabled.
		1	Channel 3 frame interrupt is enabled. The FRAME3 bit in the INTSET register must also be set to activate the FRAME3 interrupt.
2	FRAME2		Channel 2 frame interrupt enable. Enables the frame interrupt from the VPIF to CPU.
		0	Channel 2 frame interrupt is disabled.
		1	Channel 2 frame interrupt is enabled. The FRAME2 bit in the INTSET register must also be set to activate the FRAME2 interrupt.
1	FRAME1		Channel 1 frame interrupt enable. Enables the frame interrupt in BT/YC capture mode, or enables the line interval interrupt in CCD/CMOS capture mode.
		0	Channel 1 frame interrupt is disabled.
		1	Channel 1 frame interrupt is enabled. The FRAME1 bit in the INTSET register must also be set to activate the FRAME1 interrupt.
0	FRAME0		Channel 0 frame interrupt enable. Enables the frame interrupt from VPIF to CPU for BT/YC and CCD/CMOS capture modes.
		0	Channel 0 frame interrupt is disabled.
		1	Channel 0 frame interrupt is enabled. The FRAME0 bit in the INTSET register must also be set to activate the FRAME0 interrupt.

Copyright © 2013–2016, Texas Instruments Incorporated

32.3.7 Interrupt Enable Set Register (INTSET)

The interrupt enable set register (INTSET) is shown in Figure 32-24 and described in Table 32-12.

Figure 32-24. Interrupt Enable Set Register (INTSET)

31	16
Reserved	
R-0	

15	8
Reserved	
R-0	

7	4	3	2	1	0
Reserved	ERROR1	FRAME3	FRAME2	FRAME1	FRAME0
R-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0

LEGEND: R/WS = Read/Write 1 to Set; R = Read only; -*n* = value after reset

Table 32-12. Interrupt Enable Set Register (INTSET) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4	ERROR		Error interrupt enable set.
		0	Interrupt on ERROR is masked (write 0 has no effect).
		1	Interrupt on ERROR is activated.
3	FRAME3		Channel 3 frame interrupt enable set.
		0	Interrupt on FRAME3 is masked (write 0 has no effect).
		1	Interrupt on FRAME3 is activated.
2	FRAME2		Channel 2 frame interrupt enable set.
		0	Interrupt on FRAME2 is masked (write 0 has no effect).
		1	Interrupt on FRAME2 is activated.
1	FRAME1		Channel 1 frame interrupt enable set.
		0	Interrupt on FRAME1 is masked (write 0 has no effect).
		1	Interrupt on FRAME1 is activated.
0	FRAME0		Channel 0 frame interrupt enable set.
		0	Interrupt on FRAME0 is masked (write 0 has no effect).
		1	Interrupt on FRAME0 is activated.

Copyright © 2013–2016, Texas Instruments Incorporated

32.3.8 Interrupt Enable Clear Register (INTCLR)

The interrupt enable clear register (INTCLR) is shown in Figure 32-25 and described in Table 32-13.

Figure 32-25. Interrupt Enable Clear Register (INTCLR)

31	16
Reserved	
R-0	

15	8
Reserved	
R-0	

7	4	3	2	1	0
Reserved	ERROR	FRAME3	FRAME2	FRAME1	FRAME0
R-0	W1C-0	W1C-0	W1C-0	W1C-0	W1C-0

LEGEND: R = Read only; W1C = Write 1 to Clear; -*n* = value after reset

Table 32-13. Interrupt Enable Clear Register (INTCLR) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4	ERROR		Error interrupt enable clear.
		0	No change (write 0 has no effect.).
		1	Interrupt on ERROR is masked
3	FRAME3		Channel 3 frame interrupt enable clear.
		0	No change (write 0 has no effect).
		1	Interrupt on FRAME3 is masked.
2	FRAME2		Channel 2 frame interrupt enable clear.
		0	No change (write 0 has no effect).
		1	Interrupt on FRAME2 is masked.
1	FRAME1		Channel 1 frame interrupt enable clear.
		0	No change (write 0 has no effect).
		1	Interrupt on FRAME1 is masked.
0	FRAME0		Channel 0 frame interrupt enable clear.
		0	No change (write 0 has no effect).
		1	Interrupt on FRAME0 is masked.

Copyright © 2013–2016, Texas Instruments Incorporated

32.3.9 Interrupt Status Register (INTSTAT)

The interrupt status register (INTSTAT) is shown in Figure 32-26 and described in Table 32-14.

Figure 32-26. Interrupt Status Register (INTSTAT)

31	16
Reserved	
R-0	

15	8
Reserved	
R-0	

7	4	3	2	1	0
Reserved	ERROR	FRAME3	FRAME2	FRAME1	FRAME0
R-0	R-0	R-0	R-0	R-0	R-0

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

Table 32-14. Interrupt Status Register (INTSTAT) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4	ERROR		Error interrupt status. This bit is effective even if the ERROR bit in INTEN is disabled. Use the INTSTATCLR register to clear the status.
		0	No error detected.
		1	Error detected.
3	FRAME3		Channel 3 frame interrupt status. This bit is effective even if the FRAME3 bit in INTEN is disabled. Use the INTSTATCLR register to clear the status.
		0	Frame sync on channel 3 not detected.
		1	Frame sync on channel 3 detected.
2	FRAME2		Channel 2 frame interrupt status. This bit is effective even if the FRAME2 bit in INTEN is disabled. Use the INTSTATCLR register to clear the status.
		0	Frame sync on channel 2 not detected.
		1	Frame sync on channel 2 detected.
1	FRAME1		Channel 1 frame or line interval interrupt status. This bit is effective even if the FRAME1 bit in INTEN is disabled. Use the INTSTATCLR register to clear the status.
		0	Frame sync or line interval on channel 1 not detected.
		1	Frame sync or line interval on channel 1 detected.
0	FRAME0		Channel 0 frame or line interval interrupt status. This bit is effective even if the FRAME0 bit in INTEN is disabled. Use the INTSTATCLR register to clear the status.
		0	Frame sync or line interval on channel 0 not detected.
		1	Frame sync or line interval on channel 0 detected.

Copyright © 2013–2016, Texas Instruments Incorporated

32.3.10 Interrupt Status Clear Register (INTSTATCLR)

The interrupt status clear register (INTSTATCLR) is shown in Figure 32-27 and described in Table 32-15.

Figure 32-27. Interrupt Status Clear Register (INTSTATCLR)

31	16
Reserved	
R-0	

15	8
Reserved	
R-0	

7	4	3	2	1	0
Reserved	ERROR	FRAME3	FRAME2	FRAME1	FRAME0
R-0	W1C-0	W1C-0	W1C-0	W1C-0	W1C-0

LEGEND: R = Read only; W1C = Write 1 to Clear; -*n* = value after reset

Table 32-15. Interrupt Status Clear Register (INTSTATCLR) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4	ERROR		Error interrupt status clear.
		0	No change
		1	Clear ERROR bit in INTSTAT.
3	FRAME3		Channel 3 frame interrupt status clear.
		0	No change
		1	Clear FRAME3 bit in INTSTAT.
2	FRAME2		Channel 2 frame interrupt status clear.
		0	No change
		1	Clear FRAME2 bit in INTSTAT.
1	FRAME1		Channel 1 frame or line interval interrupt status clear.
		0	No change
		1	Clear FRAME1 bit in INTSTAT.
0	FRAME0		Channel 0 frame or line interval interrupt status clear.
		0	No change
		1	Clear FRAME0 bit in INTSTAT.

Copyright © 2013–2016, Texas Instruments Incorporated

32.3.11 Emulation Suspend Control Register (EMUCTRL)

The emulation suspend control register (EMUCTRL) is shown in Figure 32-28 and described in Table 32-16.

Figure 32-28. Emulation Suspend Control Register (EMUCTRL)

31	16
Reserved	
R-0	

15	1	0
Reserved		FREE
R-0		R/W-0

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

Table 32-16. Emulation Suspend Control Register (EMUCTRL) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	FREE		Controls whether the peripheral responds to the emulation suspend signal.
		0	Respond to monitoring emulation suspend signal.
		1	Ignores any emulation suspend signal (non-stop).

32.3.12 DMA Size Control Register (REQSIZE)

The DMA size control register (REQSIZE) is shown in Figure 32-29 and described in Table 32-17.

Figure 32-29. DMA Size Control Register (REQSIZE)

31	16
Reserved	
R-0	

15	9	8	0
Reserved		BYTES	
R-0		R/W-0	

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

Table 32-17. DMA Size Control Register (REQSIZE) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8-0	BYTES	0-1FF	Request size for DMA data transfer from/to VPIF.
		0-1Fh	Reserved
		20h	32 bytes
		21h-3Fh	Reserved
		40h	64 bytes
		41h-7Fh	Reserved
		80h	128 bytes
		81h-FFh	Reserved
		100h	256 bytes
		101h-1FFh	Reserved

Copyright © 2013–2016, Texas Instruments Incorporated

32.3.13 Channel *n* Top Field Luminance Address Register (CnTLUMA)

The Channel *n* Top Field Luminance Address Register (C*n*TLUMA) is shown in Figure 32-30 and described in Table 32-18.

NOTE: C1TLUMA and C3TLUMA registers are only used in Y/C multiplexed mode.

C0TLUMA is used for the top field of CCD/CMOS raw capture mode.

Figure 32-30. Channel *n* Top Field Luminance Address Register (C*n*TLUMA)

31	0
C*n*TLUMA	
R/W-0	

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

Table 32-18. Channel *n* Top Field Luminance Address Register (C*n*TLUMA) Field Descriptions

Bit	Field	Value	Description
31-0	C*n*TLUMA	0-FFFF FFFFh	Memory address of the top field luminance buffer. The register value points to the beginning of the buffer, and the address must be a multiple of 8 (C*n*TLUMA[2:0] = 0).

32.3.14 Channel *n* Bottom Field Luminance Address Register (CnBLUMA)

The Channel *n* Bottom Field Luminance Address Register (C*n*BLUMA) is shown in Figure 32-31 and described in Table 32-19.

NOTE: C*n*BLUMA registers are not used in progressive video mode. C1BLUMA and C3BLUMA registers are only used in Y/C multiplexed mode.

C0BLUMA is used for the bottom field of CCD/CMOS raw capture mode.

Figure 32-31. Channel *n* Bottom Field Luminance Address Register (C*n*BLUMA)

31	0
C*n*BLUMA	
R/W-0	

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

Table 32-19. Channel *n* Bottom Field Luminance Address Register (C*n*BLUMA) Field Descriptions

Bit	Field	Value	Description
31-0	C*n*BLUMA	0-FFFF FFFFh	Memory address of the bottom field luminance buffer. The register value points to the beginning of the buffer, and the address must be a multiple of 8 (C*n*BLUMA[2:0] = 0).

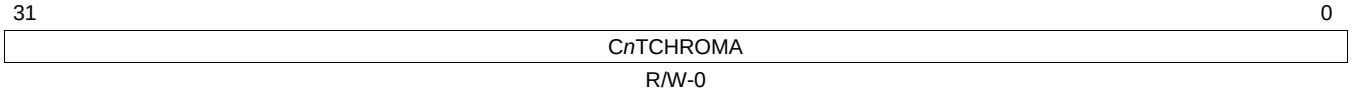
Copyright © 2013–2016, Texas Instruments Incorporated

32.3.15 Channel *n* Top Field Chrominance Address Register (C*n*TCHROMA)

The Channel *n* Top Field Chrominance Address Register (C*n*TCHROMA) is shown in Figure 32-32 and described in Table 32-20.

NOTE: C0TCHROMA and C2TCHROMA registers are only used in Y/C multiplexed mode.

Figure 32-32. Channel *n* Top Field Chrominance Address Register (C*n*TCHROMA)



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

Table 32-20. Channel *n* Top Field Chrominance Address Register (C*n*TCHROMA) Field Descriptions

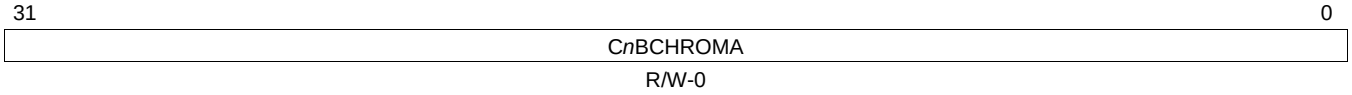
Bit	Field	Value	Description
31-0	C*n*TCHROMA	0-FFFF FFFFh	Memory address of the top field chrominance buffer. The register value points to the beginning of the buffer, and the address must be a multiple of 8 (C*n*TCHROMA[2:0] = 0).

32.3.16 Channel *n* Bottom Field Chrominance Address Register (C*n*BCHROMA)

The Channel *n* Bottom Field Chrominance Address Register (C*n*BCHROMA) is shown in Figure 32-33 and described in Table 32-21.

NOTE: C*n*BCHROMA registers are not used in progressive video mode. C0BCHROMA and C2BCHROMA registers are only used in Y/C multiplexed mode.

Figure 32-33. Channel *n* Bottom Field Chrominance Address Register (C*n*BCHROMA)



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

Table 32-21. Channel *n* Bottom Field Chrominance Address Register (C*n*BCHROMA) Field Descriptions

Bit	Field	Value	Description
31-0	C*n*BCHROMA	0-FFFF FFFFh	Memory address of the bottom field chrominance buffer. The register value points to the beginning of the buffer, and the address must be a multiple of 8 (C*n*BCHROMA[2:0] = 0).

Copyright © 2013–2016, Texas Instruments Incorporated

32.3.17 Channel *n* Top Field Horizontal Ancillary Address Register (C*n*THANC)

The Channel *n* Top Field Horizontal Ancillary Address Register (C*n*THANC) is shown in Figure 32-34 and described in Table 32-22.

Figure 32-34. Channel *n* Top Field Horizontal Ancillary Address Register (C*n*THANC)

31	0
C*n*THANC	
R/W-0	

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

Table 32-22. Channel *n* Top Field Horizontal Ancillary Address Register (C*n*THANC) Field Descriptions

Bit	Field	Value	Description
31-0	C*n*THANC	0-FFFF FFFFh	Memory address of the top field horizontal ancillary buffer. The register value points to the beginning of the buffer, and the address must be a multiple of 8 (C*n*THANC[2:0] = 0).

32.3.18 Channel *n* Bottom Field Horizontal Ancillary Address Register (C*n*BHANC)

The Channel *n* Bottom Field Horizontal Ancillary Address Register (C*n*BHANC) is shown in Figure 32-35 and described in Table 32-23.

NOTE: C*n*BHANC registers are not used in progressive video mode.

Figure 32-35. Channel *n* Bottom Field Horizontal Ancillary Address Register (C*n*BHANC)

31	0
C*n*BHANC	
R/W-0	

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

Table 32-23. Channel *n* Bottom Field Horizontal Ancillary Address Register (C*n*BHANC) Field Descriptions

Bit	Field	Value	Description
31-0	C*n*BHANC	0-FFFF FFFFh	Memory address of the bottom field horizontal ancillary buffer. The register value points to the beginning of the buffer, and the address must be a multiple of 8 (C*n*BHANC[2:0] = 0).

Copyright © 2013–2016, Texas Instruments Incorporated

32.3.19 Channel *n* Top Field Vertical Ancillary Address Register (C*n*TVANC)

The Channel *n* Top Field Vertical Ancillary Address Register (C*n*TVANC) is shown in Figure 32-36 and described in Table 32-24.

Figure 32-36. Channel *n* Top Field Vertical Ancillary Address Register (C*n*TVANC)

31	0
C*n*TVANC	
R/W-0	

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

Table 32-24. Channel *n* Top Field Vertical Ancillary Address Register (C*n*TVANC) Field Descriptions

Bit	Field	Value	Description
31-0	C*n*TVANC	0-FFFF FFFFh	Memory address of the top field vertical ancillary buffer. The register value points to the beginning of the buffer, and the address must be a multiple of 8 (C*n*TVANC[2:0] = 0).

32.3.20 Channel *n* Bottom Field Vertical Ancillary Address Register (C*n*BVANC)

The Channel *n* Bottom Field Vertical Ancillary Address Register (C*n*BVANC) is shown in Figure 32-37 and described in Table 32-25.

NOTE: C*n*BVANC registers are not used in progressive video mode.

Figure 32-37. Channel *n* Bottom Field Vertical Ancillary Data Buffer Start Address Register (C*n*BVANC)

31	0
C*n*BVANC	
R/W-0	

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

Table 32-25. Channel *n* Bottom Field Vertical Ancillary Data Buffer Start Address Register (C*n*BVANC) Field Descriptions

Bit	Field	Value	Description
31-0	C*n*BVANC	0-FFFF FFFFh	Memory address of the bottom field vertical ancillary buffer. The register value points to the beginning of the buffer, and the address must be a multiple of 8 (C*n*BVANC[2:0] = 0).

Copyright © 2013–2016, Texas Instruments Incorporated

32.3.21 Channel *n* Image Address Offset Register (C*n*IMGOFFSET)

The Channel *n* Image Address Offset Register (C*n*IMGOFFSET) is shown in Figure 32-38 and described in Table 32-26.

Figure 32-38. Channel *n* Image Address Offset Register (C*n*IMGOFFSET)

31	0
C*n*IMGOFFSET	
R/W-0	

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

Table 32-26. Channel *n* Image Address Offset Register (C*n*IMGOFFSET) Field Descriptions

Bit	Field	Value	Description
31-0	C*n*IMGOFFSET	0-FFFF FFFFh	Memory address offset of each line of image data. Value is given in bytes, note that the 3 LSBs must be 0. Address calculation method on line[x] is written in the following equation: Address = C*n*IMG_STRT_ADD + C*n*IMGOFFSET × line[x] line[x] = current_line – L3: top field or L9: bottom field Note that this value is valid only if SDRAM storage mode is raster scanning format. In sub-picture mode, this value is ignored. For vertical ancillary data, the address line offset value is automatically calculated by hardware with the configuration of the C*n*YC_MUX bit in C*n*CTRL. In Y/C mux mode, the actual address offset is C*n*IMGOFFSET × 2. In Y/C separate mode, the actual address offset is C*n*IMGOFFSET. See Storage Format for more information.

32.3.22 Channel *n* Horizontal Ancillary Address Offset Register (C*n*HANCOFFSET)

The Channel *n* Horizontal Ancillary Address Offset Register (C*n*HANCOFFSET) is shown in Figure 32-39 and described in Table 32-27.

Figure 32-39. Channel *n* Horizontal Ancillary Address Offset Register (C*n*HANCOFFSET)

31	0
C*n*HANCOFFSET	
R/W-0	

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

Table 32-27. Channel *n* Horizontal Ancillary Address Offset Register (C*n*HANCOFFSET) Field Descriptions

Bit	Field	Value	Description
31-0	C*n*HANCOFFSET	0-FFFF FFFFh	Memory address line-offset for the horizontal ancillary buffers. C*n*HANCOFFSET is given in bytes. Hanc_Address[Line] = Hanc_Buffer_Start_Address + (C*n*HANCOFFSET × Line) The offset value must be a multiple of 8 (C*n*HANCOFFSET[2:0] = 0).

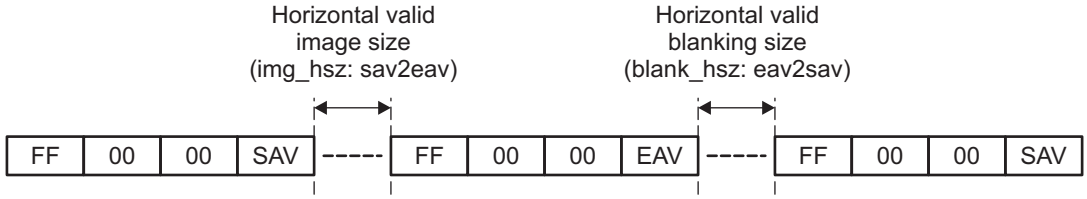
Copyright © 2013–2016, Texas Instruments Incorporated

32.3.23 Channel *n* Horizontal Size Configuration Register (C0HCFG and C1HCFG)

The Channel *n* Horizontal Size Configuration Register (C*n*HCFG) is shown in Figure 32-40 and described in Table 32-28.

The C*n*HCFG Register determines the number of bytes in the active (SAV2EAV) and inactive (EAV2SAV) video regions. In the BT/YC mode (Horizontal Distance in Y/C Mode), the video regions are delimited by the embedded EAV and SAV codes (the region size does not include the 4-byte EAV and SAV codes). In the CCD/CMOS mode (Horizontal Distance in CCD/CMOS Mode), the regions are delimited by the synchronous raw_h_valid capture signal.

Horizontal Distance in Y/C Mode



Horizontal Distance in CCD/CMOS Mode

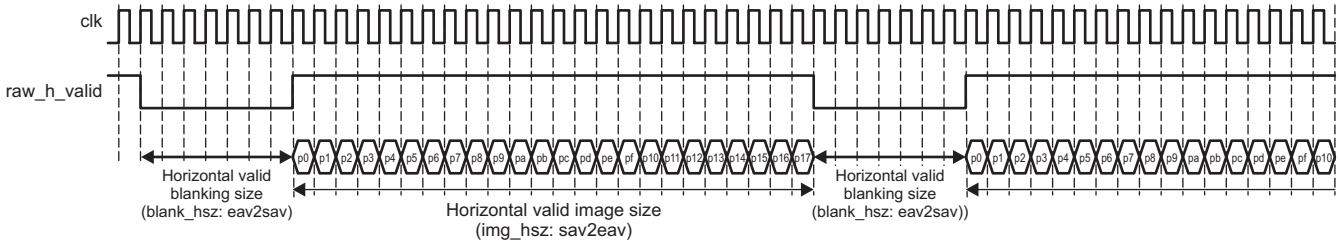


Figure 32-40. Channel *n* Horizontal Size Configuration Register (C*n*HCFG)

31	29	28	16
Reserved		EAV2SAV	
R-0		R/W-0	

15	13	12	0
Reserved		SAV2EAV	
R-0		R/W-0	

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

Table 32-28. Channel *n* Horizontal Size Configuration Register (C*n*HCFG) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Reserved
28-16	EAV2SAV	0-1FFFh	Number of bytes in the EAV2SAV region. The EAV2SAV value must be even (EAV2SAV[0:0] = 0)
15-13	Reserved	0	Reserved
12-0	SAV2EAV	0-1FFFh	Number of bytes in the SAV2EAV region. The SAV2EAV value must be even (SAV2EAV[0:0] = 0)

Copyright © 2013–2016, Texas Instruments Incorporated

32.3.24 Channel *n* Vertical Size Configuration 0 Register (C0VCFG0 and C1VCFG0)

The Channel *n* Vertical Size Configuration 0 Register (C*n*VCFG0) is shown in Figure 32-41 and described in Table 32-29.

Figure 32-41. Channel *n* Vertical Size Configuration 0 Register (C*n*VCFG0)

31	28	27	16
Reserved		L1	
R-0		R/W-0	

15	12	11	0
Reserved		L3	
R-0		R/W-0	

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

Table 32-29. Channel *n* Vertical Size Configuration 0 Register (C*n*VCFG0) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reserved
27-16	L1	0-FFFh	Enumerated line number for the L1 field position (see Interlaced and Progressive Video).
15-12	Reserved	0	Reserved
11-0	L3	0-FFFh	Enumerated line number for the L3 field position (see Interlaced and Progressive Video).

32.3.25 Channel *n* Vertical Size Configuration 1 Register (C0VCFG1 and C1VCFG1)

The Channel *n* Vertical Size Configuration 1 Register (C*n*VCFG1) is shown in Figure 32-42 and described in Table 32-30.

Figure 32-42. Channel *n* Vertical Data Size Configuration 1 Register (C*n*VCFG1)

31	28	27	16
Reserved		L5	
R-0		R/W-0	

15	12	11	0
Reserved		L7	
R-0		R/W-0	

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

Table 32-30. Channel *n* Vertical Size Configuration 1 Register (C*n*VCFG1) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reserved
27-16	L5	0-FFFh	Enumerated line number for the L5 field position (see Interlaced and Progressive Video).
15-12	Reserved	0	Reserved
11-0	L7	0-FFFh	Enumerated line number for the L7 field position (see Interlaced and Progressive Video). Note that L7 is not used with the progressive video mode.

Copyright © 2013–2016, Texas Instruments Incorporated

32.3.26 Channel *n* Vertical Size Configuration 2 Register (C0VCFG2 and C1VCFG2)

NOTE: The C*n*VCFG2 registers are not used with the progressive video mode.

The Channel *n* Vertical Size Configuration 2 Register (C*n*VCFG2) is shown in Figure 32-43 and described in Table 32-31.

Figure 32-43. Channel *n* Vertical Size Configuration 2 Register (C*n*VCFG2)

31	28	27	16
Reserved		L9	
R-0		R/W-0	

15	12	11	0
Reserved		L11	
R-0		R/W-0	

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

Table 32-31. Channel *n* Vertical Size Configuration 2 Register (C*n*VCFG2) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reserved
27-16	L9	0-FFFh	Enumerated line number for the L9 field position (see Interlaced and Progressive Video).
15-12	Reserved	0	Reserved
11-0	L11	0-FFFh	Enumerated line number for the L11 field position (see Interlaced and Progressive Video).

32.3.27 Channel *n* Vertical Image Size Register (C0VSIZE and C1VSIZE)

The Channel *n* Vertical Image Size Register (C*n*VSIZE) is shown in Figure 32-44 and described in Table 32-32.

Figure 32-44. Channel *n* Vertical Image Size Register (C*n*VSIZE)

31	16
Reserved	
R-0	

15	12	11	0
Reserved		VSIZE	
R-0		R/W-0	

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

Table 32-32. Channel *n* Vertical Image Size Register (C*n*VSIZE) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reserved
11-0	VSIZE	0-FFFh	Vertical size of image (total number of lines).

Copyright © 2013–2016, Texas Instruments Incorporated

32.3.28 Channel *n* Horizontal Size Configuration Register (C2HCFG and C3HCFG)

The Channel *n* Horizontal Size Configuration Register (C*n*HCFG) is shown in Figure 32-45 and described in Table 32-33.

The C*n*HCFG Register determines the number of bytes in the active (SAV2EAV) and inactive (EAV2SAV) video regions. In the BT/YC mode (Horizontal Distance in Y/C Mode), the video regions are delimited by the embedded EAV and SAV codes (the region size does not include the 4-byte EAV and SAV codes). In the CCD/CMOS mode (Horizontal Distance in CCD/CMOS Mode), the regions are delimited by the synchronous raw_h_valid capture signal.

Figure 32-45. Channel *n* Horizontal Size Configuration Register (C*n*HCFG)

31	27	26	16
Reserved		EAV2SAV	
R-0		R/W-0	

15	11	10	0
Reserved		SAV2EAV	
R-0		R/W-0	

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

Table 32-33. Channel *n* Horizontal Size Configuration Register (C*n*HCFG) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reserved
26-16	EAV2SAV	0-7FFh	Number of bytes in the EAV2SAV region. The EAV2SAV value must be even (EAV2SAV[0:0] = 0)
15-11	Reserved	0	Reserved
10-0	SAV2EAV	0-7FFh	Number of bytes in the SAV2EAV region. The SAV2EAV value must be even (SAV2EAV[0:0] = 0)

Copyright © 2013–2016, Texas Instruments Incorporated

32.3.29 Channel *n* Vertical Size Configuration 0 Register (C2VCFG0 and C3VCFG0)

The Channel *n* Vertical Size Configuration 0 Register (C*n*VCFG0) is shown in Figure 32-46 and described in Table 32-34.

Figure 32-46. Channel *n* Vertical Size Configuration 0 Register (C*n*VCFG0)

31	27	26	16
Reserved		L1	
R-0		R/W-0	

15	11	10	0
Reserved		L3	
R-0		R/W-0	

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

Table 32-34. Channel *n* Vertical Size Configuration 0 Register (C*n*VCFG0) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reserved
26-16	L1	0-7FFh	Enumerated line number for the L1 field position (see Interlaced and Progressive Video).
15-11	Reserved	0	Reserved
10-0	L3	0-7FFh	Enumerated line number for the L3 field position (see Interlaced and Progressive Video).

32.3.30 Channel *n* Vertical Size Configuration 1 Register (C2VCFG1 and C3VCFG1)

The Channel *n* Vertical Size Configuration 1 Register (C*n*VCFG1) is shown in Figure 32-47 and described in Table 32-35.

Figure 32-47. Channel *n* Vertical Size Configuration 1 Register (C*n*VCFG1)

31	27	26	16
Reserved		L5	
R-0		R/W-0	

15	11	10	0
Reserved		L7	
R-0		R/W-0	

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

Table 32-35. Channel *n* Vertical Size Configuration 1 Register (C*n*VCFG1) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reserved
26-16	L5	0-7FFh	Enumerated line number for the L5 field position (see Interlaced and Progressive Video).
15-11	Reserved	0	Reserved
10-0	L7	0-7FFh	Enumerated line number for the L5 field position (see Interlaced and Progressive Video). Note that L7 is not used with the progressive video mode.

Copyright © 2013–2016, Texas Instruments Incorporated

32.3.31 Channel *n* Vertical Size Configuration 2 Register (C2VCFG2 and C3VCFG2)

NOTE: The C*n*VCFG2 registers are not used with progressive video mode.

The Channel *n* Vertical Size Configuration 2 Register (C*n*VCFG2) is shown in Figure 32-48 and described in Table 32-36.

Figure 32-48. Channel *n* Vertical Size Configuration 2 Register (C*n*VCFG2)

31	27	26	16
Reserved		L9	
R-0		R/W-0	

15	11	10	0
Reserved		L11	
R-0		R/W-0	

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

Table 32-36. Channel *n* Vertical Size Configuration 2 Register (C*n*VCFG2) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reserved
26-16	L9	0-7FFh	Enumerated line number for the L9 field position (see Interlaced and Progressive Video).
15-11	Reserved	0	Reserved
10-0	L11	0-7FFh	Enumerated line number for the L11 field position (see Interlaced and Progressive Video).

32.3.32 Channel *n* Vertical Image Size Register (C2VSIZE and C3VSIZE)

The Channel *n* Vertical Image Size Register (C*n*VSIZE) is shown in Figure 32-49 and described in Table 32-37.

Figure 32-49. Channel *n* Vertical Image Size Register (C*n*VSIZE)

31	16
Reserved	
R-0	

15	11	10	0
Reserved		VSIZE	
R-0		R/W-0	

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

Table 32-37. Channel *n* Vertical Image Size Register (C*n*VSIZE) Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	0	Reserved
10-0	VSIZE	0-7FFh	Vertical size of image (total number of lines)

Copyright © 2013–2016, Texas Instruments Incorporated

32.3.33 Channel *n* Top Field Horizontal Ancillary Position Register (C2THANCPOS and C3THANCPOS)

The Channel *n* Top Field Horizontal Ancillary Position Register (C*n*THANCPOS) is shown in Figure 32-50 and described in Table 32-38.

Figure 32-50. Channel *n* Top Field Horizontal Ancillary Position Register (C*n*THANCPOS)

31	27	26	16
Reserved		VPOS	
R-0		R/W-0	

15	11	10	0
Reserved		HPOS	
R-0		R/W-0	

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

Table 32-38. Channel *n* Top Field Horizontal Ancillary Position Register (C*n*THANCPOS) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reserved
26-16	VPOS	0-7FFh	Vertical position (line-count) of valid data within the top field horizontal ancillary blanking region. Line positions are enumerated beginning with 1.
15-11	Reserved	0	Reserved
10-0	HPOS	0-7FFh	Horizontal position (byte-count) of valid data within the top field horizontal ancillary blanking region. Byte positions are enumerated beginning with 0. The value of HPOS must be a multiple of 8 (HPOS[2:0] = 0).

Copyright © 2013–2016, Texas Instruments Incorporated

32.3.34 Channel *n* Top Field Horizontal Ancillary Size Register (C2THANCSIZE and C3THANCSIZE)

The Channel *n* Top Field Horizontal Ancillary Size Register (C*n*THANCSIZE) is shown in Figure 32-51 and described in Table 32-39.

Figure 32-51. Channel *n* Top Field Horizontal Ancillary Size Register (C*n*THANCSIZE)

31	27	26	16
Reserved		VSIZE	
R-0		R/W-0	

15	11	10	0
Reserved		HSIZE	
R-0		R/W-0	

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

Table 32-39. Channel *n* Top Field Horizontal Ancillary Size Register (C*n*THANCSIZE) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reserved
26-16	VSIZE	0-7FFh	Vertical size (line-count) of valid top field horizontal ancillary data beginning at VPOS of the C*n*THANCPOS register.
15-11	Reserved	0	Reserved
10-0	HSIZE	0-7FFh	Horizontal size (byte-count) of valid top field horizontal ancillary data beginning at HPOS of the C*n*THANCPOS register. The value of HSIZE must be a multiple of 8 (HPOS[2:0] = 0).

Copyright © 2013–2016, Texas Instruments Incorporated

32.3.35 Channel *n* Bottom Field Horizontal Ancillary Position Register (C2BHANCPOS and C3BHANCPOS)

The Channel *n* Bottom Field Horizontal Ancillary Position Register (C*n*BHANCPOS) is shown in Figure 32-52 and described in Table 32-40.

Figure 32-52. Channel *n* Bottom Field Horizontal Ancillary Position Register (C*n*BHANCPOS)

31	27	26	16
Reserved		VPOS	
R-0		R/W-0	

15	11	10	0
Reserved		HPOS	
R-0		R/W-0	

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

Table 32-40. Channel *n* Bottom Field Horizontal Ancillary Position Register (C*n*BHANCPOS) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reserved
26-16	VPOS	0-7FFh	Vertical position (line-count) of valid data within the bottom field horizontal ancillary blanking region. Line positions are enumerated beginning with 1.
15-11	Reserved	0	Reserved
10-0	HPOS	0-7FFh	Horizontal position (byte-count) of valid data within the bottom field horizontal ancillary blanking region. Byte positions are enumerated beginning with 0. The value of HPOS must be a multiple of 8 (HPOS[2:0] = 0).

Copyright © 2013–2016, Texas Instruments Incorporated

32.3.36 Channel *n* Bottom Field Horizontal Ancillary Size Register (C2BHANCSIZE and C3BHANCSIZE)

The Channel *n* Bottom Field Horizontal Ancillary Size Register (C*n*BHANCSIZE) is shown in Figure 32-53 and described in Table 32-41.

Figure 32-53. Channel *n* Bottom Field Horizontal Ancillary Size Register (C*n*BHANCSIZE)

31	27	26	16
Reserved		VSIZE	
R-0		R/W-0	

15	11	10	0
Reserved		HSIZE	
R-0		R/W-0	

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

Table 32-41. Channel *n* Bottom Field Horizontal Ancillary Size Register (C*n*BHANCSIZE) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reserved
26-16	VSIZE	0-7FFh	Vertical size (line-count) of valid bottom field horizontal ancillary data beginning at VPOS of the C*n*BHANCPOS register.
15-11	Reserved	0	Reserved
10-0	HSIZE	0-7FFh	Horizontal size (byte-count) of valid bottom field horizontal ancillary data beginning at HPOS of the C*n*BHANCPOS register. The value of HSIZE must be a multiple of 8 (HSIZE[2:0] = 0).

Copyright © 2013–2016, Texas Instruments Incorporated

32.3.37 Channel *n* Top Field Vertical Ancillary Position Register (C2TVANCPOS and C3TVANCPOS)

The Channel *n* Top Field Vertical Ancillary Position Register (C*n*TVANCPOS) is shown in Figure 32-54 and described in Table 32-42.

Figure 32-54. Channel *n* Top Field Vertical Ancillary Position Register (C*n*TVANCPOS)

31	27	26	16
Reserved		VPOS	
R-0		R/W-0	

15	11	10	0
Reserved		HPOS	
R-0		R/W-0	

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

Table 32-42. Channel *n* Top Field Vertical Ancillary Position Register (C*n*TVANCPOS) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reserved
26-16	VPOS	0-7FFh	Vertical position (line-count) of valid data within the top field vertical ancillary blanking region. Line positions are enumerated beginning with 1.
15-11	Reserved	0	Reserved
10-0	HPOS	0-7FFh	Horizontal position (byte-count) of valid data within the top field vertical ancillary blanking region. Byte positions are enumerated beginning with 0. The value of HPOS must be a multiple of 8 (HPOS[2:0] = 0).

Copyright © 2013–2016, Texas Instruments Incorporated

32.3.38 Channel *n* Top Field Vertical Ancillary Size Register (C2TVANCSIZE and C3TVANCSIZE)

The Channel *n* Top Field Vertical Ancillary Size register (C*n*TVANCSIZE) is shown in Figure 32-55 and described in Table 32-43.

Figure 32-55. Channel *n* Top Field Vertical Ancillary Size Register (C*n*TVANCSIZE)

31	27	26	16
Reserved		VSIZE	
R-0		R/W-0	

15	11	10	0
Reserved		HSIZE	
R-0		R/W-0	

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

Table 32-43. Channel *n* Top Field Vertical Ancillary Size Register (C*n*TVANCSIZE) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reserved
26-16	VSIZE	0-7FFh	Vertical size (line-count) of valid top field verticall ancillary data beginning at VPOS of the C*n*TVANCPOS register.
15-11	Reserved	0	Reserved
10-0	HSIZE	0-7FFh	Horizontal size (byte-count) of valid top field vertical ancillary data beginning at HPOS of the C*n*TVANCPOS register. The value of HSIZE must be a multiple of 8 (HSIZE[2:0] = 0).

Copyright © 2013–2016, Texas Instruments Incorporated

32.3.39 Channel *n* Bottom Field Vertical Ancillary Position Register (C2BVANCPOS and C3BVANCPOS)

The Channel *n* Bottom Field Vertical Ancillary Position Register (C*n*BVANCPOS) is shown in Figure 32-56 and described in Table 32-44.

Figure 32-56. Channel *n* Bottom Field Vertical Ancillary Position Register (C*n*BVANCPOS)

31	27	26	16
Reserved		VPOS	
R-0		R/W-0	

15	11	10	0
Reserved		HPOS	
R-0		R/W-0	

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

Table 32-44. Channel *n* Bottom Field Vertical Ancillary Position Register (C*n*BVANCPOS) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reserved
26-16	VPOS	0-7FFh	Vertical position (line-count) of valid data within the bottom field vertical ancillary blanking region. Line positions are enumerated beginning with 1.
15-11	Reserved	0	Reserved
10-0	HPOS	0-7FFh	Horizontal position (byte-count) of valid data within the bottom field vertical ancillary blanking region. Byte positions are enumerated beginning with 0. The value of HPOS must be a multiple of 8 (HPOS[2:0] = 0).

Copyright © 2013–2016, Texas Instruments Incorporated

32.3.40 Channel *n* Bottom Field Vertical Ancillary Size Register (C2BVANCSIZE and C3BVANCSIZE)

The Channel *n* Bottom Field Vertical Ancillary Size Register (C*n*BVANCSIZE) is shown in Figure 32-57 and described in Table 32-45.

Figure 32-57. Channel *n* Bottom Field Vertical Ancillary Size Register (C*n*BVANCSIZE)

31	27	26	16
Reserved		VSIZE	
R-0		R/W-0	

15	11	10	0
Reserved		HSIZE	
R-0		R/W-0	

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

Table 32-45. Channel *n* Bottom Field Vertical Ancillary Size Register (C*n*BVANCSIZE) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reserved
26-16	VSIZE	0-7FFh	Vertical size (line-count) of valid bottom field verticall ancillary data beginning at VPOS of the C*n*BVANCPOS register.
15-11	Reserved	0	Reserved
10-0	HSIZE	0-7FFh	Horizontal size (byte-count) of valid bottom field vertical ancillary data beginning at HPOS of the C*n*BVANCPOS register. The value of HSIZE must be a multiple of 8 (HSIZE[2:0] = 0).

Copyright © 2013–2016, Texas Instruments Incorporated

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from July 10, 2016 to September 12, 2016 (from B Revision (July 2016) to C Revision)	Page
• Updated Chapter 13 entirely	301

Copyright © 2013–2016, Texas Instruments Incorporated

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2016, Texas Instruments Incorporated