RM46x 16/32-Bit RISC Flash Microcontroller

Technical Reference Manual

Texas Instruments

Literature Number: SPNU514C
March 2018

Contents

Preface **93**

1 Introduction **95**

1.1 Designed for Safety Applications 96

1.2 Family Description 96

1.3 Endianism Considerations 99

1.3.1 RM46x: Little Endian (LE) 99

2 Architecture **100**

2.1 Introduction 101

2.1.1 Architecture Block Diagram 101

2.1.2 Definitions of Terms 102

2.1.3 Bus Master / Slave Access Privileges 104

2.2 Memory Organization 105

2.2.1 Memory-Map Overview 105

2.2.2 Memory-Map Table 106

2.2.3 Flash on Microcontrollers 110

2.2.4 On-Chip SRAM 111

2.3 Exceptions 117

2.3.1 Resets 117

2.3.2 Aborts 117

2.3.3 System Software Interrupts 119

2.4 Clocks 120

2.4.1 Clock Sources 120

2.4.2 Clock Domains 121

2.4.3 Low Power Modes 124

2.4.4 Clock Test Mode 126

2.4.5 Safety Considerations for Clocks 127

2.5 System and Peripheral Control Registers 130

2.5.1 Primary System Control Registers (SYS) 130

2.5.2 Secondary System Control Registers (SYS2) 185

2.5.3 Peripheral Central Resource (PCR) Control Registers 194

3 Power Management Module (PMM) **209**

3.1 Overview 210

3.1.1 Main Features of the Power Management Module (PMM) 210

3.1.2 Block Diagram 210

3.2 Power Domains 212

3.3 PMM Operation 213

3.3.1 Power Switch 213

3.3.2 Power Domain State 213

3.3.3 Default Power Domain State 213

3.3.4 Disabling a Power Domain Permanently 213

3.3.5 Changing Power Domain State 213

3.3.6 Reset Management 214

3.3.7 Diagnostic Power State Controller (PSCON) 214

3.3.8 PSCON Compare Block 214

3.4 PMM Registers 216

3.4.1 LOGICPDPWRCTRL0 Register (Offset = 0h) [reset = X] 217
3.4.2 MEMPDPWRCTRL0 Register (Offset = 10h) [reset = X] 219
3.4.3 PDCLKDIS Register (Offset = 20h) [reset = X] 220
3.4.4 PDCLKDISSET Register (Offset = 24h) [reset = X] 221
3.4.5 PDCLKDISCLR Register (Offset = 28h) [reset = X] 222
3.4.6 LOGICPDPWRSTAT0 Register (Offset = 40h) [reset = X] 223
3.4.7 LOGICPDPWRSTAT1 Register (Offset = 44h) [reset = X] 224
3.4.8 LOGICPDPWRSTAT2 Register (Offset = 48h) [reset = X] 225
3.4.9 LOGICPDPWRSTAT3 Register (Offset = 4Ch) [reset = X] 226
3.4.10 MEMPDPWRSTAT0 Register (Offset = 80h) [reset = X] 227
3.4.11 MEMPDPWRSTAT1 Register (Offset = 84h) [reset = X] 229
3.4.12 GLOBALCTRL1 Register (Offset = A0h) [reset = 0h] 231
3.4.13 GLOBALSTAT Register (Offset = A8h) [reset = 1h] 232
3.4.14 PRCKEYREG Register (Offset = ACh) [reset = 0h] 233
3.4.15 LPDDCSTAT1 Register (Offset = B0h) [reset = 0h] 234
3.4.16 LPDDCSTAT2 Register (Offset = B4h) [reset = 0h] 235
3.4.17 MPDDCSTAT1 Register (Offset = B8h) [reset = 0h] 236
3.4.18 MPDDCSTAT2 Register (Offset = BCh) [reset = 0h] 237
3.4.19 ISODIAGSTAT Register (Offset = C0h) [reset = 0h] 238

4 I/O Multiplexing and Control Module (IOMM) 239

4.1 Overview 240
4.2 Main Features of I/O Multiplexing Module (IOMM) 240
4.3 Control of Multiplexed Outputs 240
4.4 Control of Multiplexed Inputs 241
4.5 Control of Special Multiplexed Options 242
4.5.1 Control of SDRAM clock (EMIF_CLK) 242
4.5.2 Control for other EMIF Outputs 242
4.5.3 Control for Fast Mode for Synchronous and Asynchronous EMIF 243
4.5.4 Control of Ethernet Controller Mode 243
4.5.5 Control of ADC Trigger Events 243
4.5.6 Control for ADC Event Trigger Signal Generation from ePWMx Modules 244
4.5.7 Control for Generating Interrupt Upon External Fault Indication to N2HETx 246
4.5.8 Control for Enabling Clocks to ePWMx Modules 246
4.5.9 Control for Enabling Clocks to eCAPx Modules 247
4.5.10 Control for Enabling Clocks to eQEPx Modules 247
4.5.11 Control for Synchronizing Time Bases for All ePWMx Modules 247
4.5.12 Control for Synchronizing all ePWMx Modules to N2HET1 Module Time-Base 248
4.5.13 Control for Input Connections to ePWMx Modules 249
4.5.14 Control for Input Connections to eCAPx Modules 250
4.5.15 Control for Input Connections to eQEPx Modules 250
4.6 Safety Features 251
4.6.1 Locking Mechanism for Memory-Mapped Registers 251
4.6.2 Error Conditions 251
4.7 IOMM Registers 252
4.7.1 REVISION_REG Register (Offset = 0h) [reset = 4E840102h] 253
4.7.2 ENDIAN_REG Register (Offset = 20h) [reset = 0h] 254
4.7.3 KICK_REG0 Register (Offset = 38h) [reset = 0h] 255
4.7.4 KICK_REG1 Register (Offset = 3Ch) [reset = 0h] 256
4.7.5 ERR_RAW_STATUS_REG Register (Offset = E0h) [reset = 0h] 257
4.7.6 ERR_ENABLED_STATUS_REG Register (Offset = E4h) [reset = 0h] 258
4.7.7 ERR_ENABLE_REG Register (Offset = E8h) [reset = 0h] 259
4.7.8 ERR_ENABLE_CLR_REG Register (Offset = ECh) [reset = 0h] 260
4.7.9 FAULT_ADDRESS_REG Register (Offset = F4h) [reset = 0h] 261

4.7.10 FAULT_STATUS_REG Register (Offset = F8h) [reset = 0h] 262
4.7.11 FAULT_CLEAR_REG Register (Offset = FCh) [reset = 0h] 263
4.7.12 PINMMR_0 to PINMMR_47 Register (Offset = B10h to BCCh) [reset = 1010101h] 264
4.8 Output Multiplexing and Control 265

5 F021 Flash Module Controller (FMC) 267

5.1 Overview 268
5.1.1 Features 268
5.1.2 Definition of Terms 268
5.1.3 F021 Flash Tools 269
5.2 Default Flash Configuration 269
5.3 SECDED 270
5.3.1 SECDED Initialization 270
5.3.2 ECC Encoding 271
5.3.3 Syndrome Table: Decode to Bit in Error 272
5.3.4 Syndrome Table: An Alternate Method 273
5.4 Memory Map 274
5.4.1 Location of Flash ECC Bits 274
5.4.2 OTP Memory 275
5.5 Power On, Power Off, and Reset Considerations 277
5.5.1 Error Checking at Power On 277
5.5.2 Flash Integrity when Reset while Programming or Erasing 277
5.5.3 Flash Integrity at Power Off 278
5.6 Emulation and SIL3 Diagnostic Modes 278
5.6.1 System Emulation 278
5.6.2 Diagnostic Mode 278
5.6.3 Diagnostic Mode Summary 283
5.6.4 Read Margin 284
5.7 Control Registers 284
5.7.1 Flash Option Control Register (FRDCNTL) 286
5.7.2 Flash Error Detection and Correction Control Register 1 (FEDACCTRL1) 287
5.7.3 Flash Error Correction and Correction Control Register 2 (FEDACCTRL2) 289
5.7.4 Flash Correctable Error Count Register (FCOR_ERR_CNT) 289
5.7.5 Flash Correctable Error Address Register (FCOR_ERR_ADD) 290
5.7.6 Flash Correctable Error Position Register (FCOR_ERR_POS) 291
5.7.7 Flash Error Detection and Correction Status Register (FEDACSTATUS) 292
5.7.8 Flash Uncorrectable Error Address Register (FUNC_ERR_ADD) 295
5.7.9 Flash Error Detection and Correction Sector Disable Register (FEDACSDIS) 296
5.7.10 Primary Address Tag Register (FPRIM_ADD_TAG) 297
5.7.11 Duplicate Address Tag Register (FDUP_ADD_TAG) 297
5.7.12 Flash Bank Protection Register (FBPROT) 298
5.7.13 Flash Bank Sector Enable Register (FBSE) 298
5.7.14 Flash Bank Busy Register (FBBUSY) 299
5.7.15 Flash Bank Access Control Register (FBAC) 300
5.7.16 Flash Bank Fallback Power Register (FBFALLBACK) 301
5.7.17 Flash Bank/Pump Ready Register (FBPRDY) 302
5.7.18 Flash Pump Access Control Register 1 (FPAC1) 303
5.7.19 Flash Pump Access Control Register 2 (FPAC2) 304
5.7.20 Flash Module Access Control Register (FMAC) 304
5.7.21 Flash Module Status Register (FMSTAT) 305
5.7.22 EEPROM Emulation Data MSW Register (FEMU_DMSW) 307
5.7.23 EEPROM Emulation Data LSW Register (FEMU_DLSW) 307
5.7.24 EEPROM Emulation ECC Register (FEMU_ECC) 308
5.7.25 EEPROM Emulation Address Register (FEMU_ADDR) 309

5.7.26 Diagnostic Control Register (FDIAGCTRL) 310
5.7.27 Uncorrected Raw Data High Register (FRAW_DATAH) 312
5.7.28 Uncorrected Raw Data Low Register (FRAW_DATAL) 312
5.7.29 Uncorrected Raw ECC Register (FRAW_ECC) 313
5.7.30 Parity Override Register (FPAR_OVR) 314
5.7.31 Flash Error Detection and Correction Sector Disable Register (FEDACSDIS2) 315
5.7.32 FSM Register Write Enable (FSM_WR_ENA) 316
5.7.33 FSM Sector Register (FSM_SECTOR) 316
5.7.34 EEPROM Emulation Configuration Register (EEPROM_CONFIG) 317
5.7.35 EEPROM Emulation Error Detection and Correction Control Register 1 (EE_CTRL1) 318
5.7.36 EEPROM Emulation Error Correction and Correction Control Register 2 (EE_CTRL2) 320
5.7.37 EEPROM Emulation Correctable Error Count Register (EE_COR_ERR_CNT) 320
5.7.38 EEPROM Emulation Correctable Error Address Register (EE_COR_ERR_ADD) 321
5.7.39 EEPROM Emulation Correctable Error Position Register (EE_COR_ERR_POS) 322
5.7.40 EEPROM Emulation Error Status Register (EE_STATUS) 323
5.7.41 EEPROM Emulation Uncorrectable Error Address Register (EE_UNC_ERR_ADD) 324
5.7.42 Flash Bank Configuration Register (FCFG_BANK) 325

6 Tightly-Coupled RAM (TCRAM) Module 326
6.1 Overview 327
6.1.1 B0TCM and B1TCM Connection Diagram 327
6.1.2 Main Features 327
6.2 RAM Memory Map 328
6.3 Safety Features 329
6.3.1 Support for Cortex-R4F CPU's Single-Error-Correction Double-Error-Detection (SECDED) 329
6.3.2 Support for Cortex-R4F CPU's Address and Control Bus Parity Checking 330
6.3.3 Redundant Address Decode 330
6.4 TCRAM Auto-Initialization 330
6.5 Trace Module Support 331
6.6 Emulation / Debug Mode Behavior 331
6.7 TCRAM_MODULE_CONTROL_AND_STATUS Registers 332
6.7.1 RAMCTRL Register (Offset = 0h) [reset = 0005000Ah] 333
6.7.2 RAMTHRESHOLD Register (Offset = 4h) [reset = 0h] 335
6.7.3 RAMOCCUR Register (Offset = 8h) [reset = 0h] 336
6.7.4 RAMINTCTRL Register (Offset = Ch) [reset = 0h] 337
6.7.5 RAMERRSTATUS Register (Offset = 10h) [reset = 0h] 338
6.7.6 RAMSERRADDR Register (Offset = 14h) [reset = 0h] 340
6.7.7 RAMUERRADDR Register (Offset = 1Ch) [reset = 0h] 341
6.7.8 RAMTEST Register (Offset = 30h) [reset = 5h] 342
6.7.9 RAMADDRDECVECT Register (Offset = 38h) [reset = 0h] 344
6.7.10 RAMPERRADDR Register (Offset = 3Ch) [reset = X] 345
6.7.11 INIT_DOMAIN Register (Offset = 40h) [reset = FFh] 346

7 Programmable Built-In Self-Test (PBIST) Module 347
7.1 Overview 348
7.1.1 Features of PBIST 348
7.1.2 PBIST vs. Application Software-Based Testing 348
7.1.3 PBIST Block Diagram 348
7.2 RAM Grouping and Algorithm 349
7.3 PBIST Flow 350
7.3.1 PBIST Sequence 351
7.4 Memory Test Algorithms on the On-chip ROM 352
7.5 PBIST Control Registers 354
7.5.1 RAM Configuration Register (RAMT) 355
7.5.2 Datalogger Register (DLR) 356

7.5.3 PBIST Activate/Clock Enable Register (PACT) 357
7.5.4 PBIST ID Register 358
7.5.5 Override Register (OVER) 359
7.5.6 Fail Status Fail Register (FSRF0) 360
7.5.7 Fail Status Count Registers (FSRC0 and FSRC1) 361
7.5.8 Fail Status Address Registers (FSRA0 and FSRA1) 362
7.5.9 Fail Status Data Registers (FSRDL0 and FSRDL1) 363
7.5.10 ROM Mask Register (ROM) 364
7.5.11 ROM Algorithm Mask Register (ALGO) 364
7.5.12 RAM Info Mask Lower Register (RINFOL) 365
7.5.13 RAM Info Mask Upper Register (RINFOU) 366
7.6 PBIST Configuration Example 367
7.6.1 Example 1 : Configuration of PBIST Controller to Run Self-Test on RAM Group 3 367
7.6.2 Example 2 : Configuration of PBIST Controller to Run Self-Test on ALL RAM Groups 368

8 CPU Self-Test Controller (STC) Module 369
8.1 General Description 370
8.1.1 CPU Self-Test Controller Features 370
8.1.2 STC Block Diagram 370
8.2 Application Self-Test Flow 372
8.2.1 STC Module Configuration 372
8.2.2 Context Saving 372
8.2.3 Entering CPU Idle Mode 372
8.2.4 Self-Test Completion and Error Generation 373
8.3 STC Test Coverage and Duration 374
8.4 STC Control Registers 375
8.4.1 STC Global Control Register 0 (STCGCR0) 376
8.4.2 STC Global Control Register 1 (STCGCR1) 376
8.4.3 Self-Test Run Timeout Counter Preload Register (STCTPR) 377
8.4.4 STC Current ROM Address Register (STC_CADDR) 377
8.4.5 STC Current Interval Count Register (STCCICR) 378
8.4.6 Self-Test Global Status Register (STCGSTAT) 379
8.4.7 Self-Test Fail Status Register (STCFSTAT) 380
8.4.8 CPU1 Current MISR Register (CPU1_CURMISR[3:0]) 381
8.4.9 CPU2_CURMISR[3:0] (CPU2 Current MISR Register) 382
8.4.10 STCSCSCR (Signature Compare Self-Check Register) 383
8.5 STC Configuration Example 384
8.5.1 Example 1: Self-Test Run for 24 Interval 384

9 CPU Compare Module for Cortex-R4F (CCM-R4F) 385
9.1 Main Features 386
9.2 Block Diagram 386
9.3 Module Operation 387
9.3.1 1oo1D Lock Step Mode 387
9.3.2 Self-Test Mode 387
9.3.3 Error Forcing Mode 389
9.3.4 Self-Test Error Forcing Mode 389
9.3.5 Operation During CPU Debug Mode 390
9.4 CCM-R4F Control Registers 390
9.4.1 CCM-R4F Status Register (CCMSR) 391
9.4.2 CCM-R4F Key Register (CCMKEYR) 392

10 Oscillator and PLL 393
10.1 Introduction 394
10.1.1 Features 394
10.2 Quick Start 395

10.3 Oscillator 396
10.3.1 Oscillator Implementation 397
10.3.2 Oscillator Enable 397
10.3.3 Oscillator Disable 397
10.4 Low-Power Oscillator and Clock Detect (LPOCLKDET) 398
10.4.1 Clock Detect 398
10.4.2 Behavior on Oscillator Failure 398
10.4.3 Recovery from Oscillator Failure 399
10.4.4 LPOCLKDET Enable 399
10.4.5 LPOCLKDET Disable 400
10.4.6 Trimming the HF LPO Oscillator 400
10.5 PLL 401
10.5.1 Modulation 403
10.5.2 PLL Output Control 404
10.5.3 Behavior on PLL Fail 407
10.5.4 Recovery from a PLL Failure 408
10.5.5 PLL Modulation Depth Measurement 409
10.5.6 PLL Frequency Measurement Circuit 410
10.5.7 PLL2 410
10.6 PLL Control Registers 411
10.6.1 PLL Modulation Depth Measurement Control Register (SSWPLL1) 412
10.6.2 SSW PLL BIST Control Register 2 (SSWPLL2) 413
10.6.3 SSW PLL BIST Control Register 3 (SSWPLL3) 414
10.7 Phase-Locked Loop Theory of Operation 415
10.7.1 Phase-Frequency Detector 415
10.7.2 Charge Pump and Loop Filter 416
10.7.3 Voltage-Controlled Oscillator 416
10.7.4 Frequency Modulation 417
10.8 Programming Example 417
11 Dual-Clock Comparator (DCC) Module 419
11.1 Introduction 420
11.1.1 Main Features 420
11.1.2 Block Diagram 420
11.2 Module Operation 421
11.2.1 Continuous Monitoring Mode 421
11.2.2 Single-Shot Measurement Mode 424
11.3 Clock Source Selection for Counter0 and Counter1 425
11.4 DCC Control Registers 426
11.4.1 DCC Global Control Register (DCCGCTRL) 427
11.4.2 DCC Revision Id Register (DCCREV) 428
11.4.3 DCC Counter0 Seed Register (DCCCNT0SEED) 428
11.4.4 DCC Valid0 Seed Register (DCCVALID0SEED) 429
11.4.5 DCC Counter1 Seed Register (DCCCNT1SEED) 429
11.4.6 DCC Status Register (DCCSTAT) 430
11.4.7 DCC Counter0 Value Register (DCCCNT0) 431
11.4.8 DCC Valid0 Value Register (DCCVALID0) 432
11.4.9 DCC Counter1 Value Register (DCCCNT1) 432
11.4.10 DCC Counter1 Clock Source Selection Register (DCCCNT1CLKSRC) 433
11.4.11 DCC Counter0 Clock Source Selection Register (DCCCNT0CLKSRC) 434
12 Error Signaling Module (ESM) 435
12.1 Overview 436
12.1.1 Features 436
12.1.2 Block Diagram 436

12.2 Module Operation 438
12.2.1 Reset Behavior 438
12.2.2 $\overline{\text{ERROR}}$ Pin Timing 439
12.2.3 Forcing an Error Condition 441
12.3 Recommended Programming Procedure 442
12.4 Control Registers 443
12.4.1 ESM Enable $\overline{\text{ERROR}}$ Pin Action/Response Register 1 (ESMEEPAPR1) 444
12.4.2 ESM Disable $\overline{\text{ERROR}}$ Pin Action/Response Register 1 (ESMDEPAPR1) 444
12.4.3 ESM Interrupt Enable Set Register 1 (ESMIESR1) 445
12.4.4 ESM Interrupt Enable Clear Register 1 (ESMIECR1) 445
12.4.5 ESM Interrupt Level Set Register 1 (ESMILSR1) 446
12.4.6 ESM Interrupt Level Clear Register 1 (ESMILCR1) 446
12.4.7 ESM Status Register 1 (ESMSR1) 447
12.4.8 ESM Status Register 2 (ESMSR2) 447
12.4.9 ESM Status Register 3 (ESMSR3) 448
12.4.10 ESM $\overline{\text{ERROR}}$ Pin Status Register (ESMEPSR) 448
12.4.11 ESM Interrupt Offset High Register (ESMIOFFHR) 449
12.4.12 ESM Interrupt Offset Low Register (ESMIOFFLR) 450
12.4.13 ESM Low-Time Counter Register (ESMLTCR) 451
12.4.14 ESM Low-Time Counter Preload Register (ESMLTCPR) 451
12.4.15 ESM Error Key Register (ESMEKR) 452
12.4.16 ESM Status Shadow Register 2 (ESMSSR2) 452
12.4.17 ESM Influence $\overline{\text{ERROR}}$ Pin Set Register 4 (ESMIEPSR4) 453
12.4.18 ESM Influence $\overline{\text{ERROR}}$ Pin Clear Register 4 (ESMIEPCR4) 453
12.4.19 ESM Interrupt Enable Set Register 4 (ESMIESR4) 454
12.4.20 ESM Interrupt Enable Clear Register 4 (ESMIECR4) 454
12.4.21 ESM Interrupt Level Set Register 4 (ESMILSR4) 455
12.4.22 ESM Interrupt Level Clear Register 4 (ESMILCR4) 455
12.4.23 ESM Status Register 4 (ESMSR4) 456
13 Real-Time Interrupt (RTI) Module 457
13.1 Overview 458
13.1.1 Features 458
13.1.2 Industry Standard Compliance Statement 458
13.2 Module Operation 459
13.2.1 Counter Operation 459
13.2.2 Interrupt/DMA Requests 461
13.2.3 RTI Clocking 462
13.2.4 Synchronizing Timer Events to Network Time (NTU) 462
13.2.5 Digital Watchdog (DWD) 465
13.2.6 Low Power Modes 468
13.2.7 Halting Debug Mode Behaviour 468
13.3 RTI Control Registers 469
13.3.1 RTI Global Control Register (RTIGCTRL) 470
13.3.2 RTI Timebase Control Register (RTITBCTRL) 471
13.3.3 RTI Capture Control Register (RTICAPCTRL) 472
13.3.4 RTI Compare Control Register (RTICOMPCTRL) 473
13.3.5 RTI Free Running Counter 0 Register (RTIFRC0) 474
13.3.6 RTI Up Counter 0 Register (RTIUC0) 474
13.3.7 RTI Compare Up Counter 0 Register (RTICPUC0) 475
13.3.8 RTI Capture Free Running Counter 0 Register (RTICAFRC0) 475
13.3.9 RTI Capture Up Counter 0 Register (RTICAUC0) 476
13.3.10 RTI Free Running Counter 1 Register (RTIFRC1) 476
13.3.11 RTI Up Counter 1 Register (RTIUC1) 477

13.3.12 RTI Compare Up Counter 1 Register (RTICPUC1) 478
13.3.13 RTI Capture Free Running Counter 1 Register (RTICAFRC1) 479
13.3.14 RTI Capture Up Counter 1 Register (RTICAUC1) 479
13.3.15 RTI Compare 0 Register (RTICOMP0) 480
13.3.16 RTI Update Compare 0 Register (RTIUDCP0) 480
13.3.17 RTI Compare 1 Register (RTICOMP1) 481
13.3.18 RTI Update Compare 1 Register (RTIUDCP1) 481
13.3.19 RTI Compare 2 Register (RTICOMP2) 482
13.3.20 RTI Update Compare 2 Register (RTIUDCP2) 482
13.3.21 RTI Compare 3 Register (RTICOMP3) 483
13.3.22 RTI Update Compare 3 Register (RTIUDCP3) 483
13.3.23 RTI Timebase Low Compare Register (RTITBLCOMP) 484
13.3.24 RTI Timebase High Compare Register (RTITBHCOMP) 484
13.3.25 RTI Set Interrupt Enable Register (RTISETINTENA) 485
13.3.26 RTI Clear Interrupt Enable Register (RTICLEARINTENA) 487
13.3.27 RTI Interrupt Flag Register (RTIINTFLAG) 489
13.3.28 Digital Watchdog Control Register (RTIDWDCTRL) 490
13.3.29 Digital Watchdog Preload Register (RTIDWDPRLD) 491
13.3.30 Watchdog Status Register (RTIWDSTATUS) 492
13.3.31 RTI Watchdog Key Register (RTIWDKEY) 493
13.3.32 RTI Digital Watchdog Down Counter (RTIDWDCNTR) 494
13.3.33 Digital Windowed Watchdog Reaction Control (RTIWWDRXNCTRL) 494
13.3.34 Digital Windowed Watchdog Window Size Control (RTIWWDSIZECTRL) 495
13.3.35 RTI Compare Interrupt Clear Enable Register (RTIINTCLRENABLE) 496
13.3.36 RTI Compare 0 Clear Register (RTICMP0CLR) 497
13.3.37 RTI Compare 1 Clear Register (RTICMP1CLR) 497
13.3.38 RTI Compare 2 Clear Register (RTICMP2CLR) 498
13.3.39 RTI Compare 3 Clear Register (RTICMP3CLR) 498

14 Cyclic Redundancy Check (CRC) Controller Module 499

14.1 Overview 500
14.1.1 Features 500
14.1.2 Block Diagram 501
14.2 Module Operation 502
14.2.1 General Operation 502
14.2.2 CRC Modes of Operation 502
14.2.3 PSA Signature Register 503
14.2.4 PSA Sector Signature Register 504
14.2.5 CRC Value Register 505
14.2.6 Raw Data Register 505
14.2.7 Example DMA Controller Setup 505
14.2.8 Pattern Count Register 507
14.2.9 Sector Count Register/Current Sector Register 507
14.2.10 Interrupt 508
14.2.11 CPU Data Trace 511
14.2.12 Power Down Mode 512
14.2.13 Emulation 512
14.2.14 Peripheral Bus Interface 512
14.3 Example 513
14.3.1 Example: Auto Mode Using Time Based Event Triggering 513
14.3.2 Example: Auto Mode Without Using Time Based Triggering 514
14.3.3 Example: Semi-CPU Mode 515
14.3.4 Example: Full-CPU Mode 515
14.4 CRC Control Registers 516

14.4.1 CRC Global Control Register 0 (CRC_CTRL0) 517
14.4.2 CRC Global Control Register (CRC_CTRL1) 517
14.4.3 CRC Global Control Register 2 (CRC_CTRL2) 518
14.4.4 CRC Interrupt Enable Set Register (CRC_INTS) 519
14.4.5 CRC Interrupt Enable Reset Register (CRC_INTR) 521
14.4.6 CRC Interrupt Status Register (CRC_STATUS) 523
14.4.7 CRC Interrupt Offset (CRC_INT_OFFSET_REG) 524
14.4.8 CRC Busy Register (CRC_BUSY) 526
14.4.9 CRC Pattern Counter Preload Register 1 (CRC_PCOUNT_REG1) 526
14.4.10 CRC Sector Counter Preload Register 1 (CRC_SCOUNT_REG1) 527
14.4.11 CRC Current Sector Register 1 (CRC_CURSEC_REG1) 527
14.4.12 CRC Channel 1 Watchdog Timeout Preload Register A (CRC_WDTOPLD1) 528
14.4.13 CRC Channel 1 Block Complete Timeout Preload Register B (CRC_BCTOPLD1) 528
14.4.14 Channel 1 PSA Signature Low Register (PSA_SIGREGL1) 529
14.4.15 Channel 1 PSA Signature High Register (PSA_SIGREGH1) 529
14.4.16 Channel 1 CRC Value Low Register (CRC_REGL1) 529
14.4.17 Channel 1 CRC Value High Register (CRC_REGH1) 530
14.4.18 Channel 1 PSA Sector Signature Low Register (PSA_SECSIGREGL1) 530
14.4.19 Channel 1 PSA Sector Signature High Register (PSA_SECSIGREGH1) 530
14.4.20 Channel 1 Raw Data Low Register (RAW_DATAREGL1) 531
14.4.21 Channel 1 Raw Data High Register (RAW_DATAREGH1) 531
14.4.22 CRC Pattern Counter Preload Register 2 (CRC_PCOUNT_REG2) 531
14.4.23 CRC Sector Counter Preload Register 2 (CRC_SCOUNT_REG2) 532
14.4.24 CRC Current Sector Register 2 (CRC_CURSEC_REG2) 532
14.4.25 CRC Channel 2 Watchdog Timeout Preload Register A (CRC_WDTOPLD2) 533
14.4.26 CRC Channel 2 Block Complete Timeout Preload Register B (CRC_BCTOPLD2) 533
14.4.27 Channel 2 PSA Signature Low Register (PSA_SIGREGL2) 534
14.4.28 Channel 2 PSA Signature High Register (PSA_SIGREGH2) 534
14.4.29 Channel 2 CRC Value Low Register (CRC_REGL2) 534
14.4.30 Channel 2 CRC Value High Register (CRC_REGH2) 535
14.4.31 Channel 2 PSA Sector Signature Low Register (PSA_SECSIGREGL2) 535
14.4.32 Channel 2 PSA Sector Signature High Register (PSA_SECSIGREGH2) 535
14.4.33 Channel 2 Raw Data Low Register (RAW_DATAREGL2) 536
14.4.34 Channel 2 Raw Data High Register (RAW_DATAREGH2) 536
14.4.35 Data Bus Selection Register (CRC_TRACE_BUS_SEL) 537

15 Vectored Interrupt Manager (VIM) Module 538
15.1 Overview 539
15.2 Device Level Interrupt Management 539
15.2.1 Interrupt Generation at the Peripheral 540
15.2.2 Interrupt Handling at the CPU 540
15.2.3 Software Interrupt Handling Options 541
15.3 Interrupt Handling Inside VIM 542
15.3.1 VIM Interrupt Channel Mapping 543
15.3.2 VIM Input Channel Management 545
15.4 Interrupt Vector Table (VIM RAM) 546
15.4.1 Interrupt Vector Table Operation 546
15.4.2 Enabling and Controlling the VIM Parity 547
15.4.3 Interrupt Vector Table Initialization 547
15.4.4 Interrupt Vector Table Parity Testing 548
15.5 VIM Wakeup Interrupt 549
15.6 Capture Event Sources 550
15.7 Examples 550
15.7.1 Examples - Configure CPU to Receive Interrupts 550

15.7.2 Examples - Register Vector Interrupt and Index Interrupt Handling 551
15.8 VIM Registers 553
15.8.1 Interrupt Vector Table Parity Flag Register (PARFLG) 554
15.8.2 Interrupt Vector Table Parity Control Register (PARCTL).. 554
15.8.3 Address Parity Error Register (ADDERR) 555
15.8.4 Fall-Back Address Parity Error Register (FBPARERR)... 555
15.8.5 VIM Offset Vector Registers.. 556
15.8.6 FIQ/IRQ Program Control Registers (FIRQPR[0:3]) 558
15.8.7 Pending Interrupt Read Location Registers (INTREQ[0:3]) 560
15.8.8 Interrupt Enable Set Registers (REQENASET[0:3]) 562
15.8.9 Interrupt Enable Clear Registers (REQENACLR[0:3]).. 564
15.8.10 Wake-Up Enable Set Registers (WAKEENASET[0:3]).. 566
15.8.11 Wake-Up Enable Clear Registers (WAKEENACLR[0:3]) 568
15.8.12 IRQ Interrupt Vector Register (IRQVECREG) 570
15.8.13 FIQ Interrupt Vector Register (FIQVECREG) 570
15.8.14 Capture Event Register (CAPEVT) 571
15.8.15 VIM Interrupt Control Registers (CHANCTRL[0:31]) 572

16 Direct Memory Access Controller (DMA) Module 574

16.1 Overview 575
16.1.1 Main Features 575
16.2 Module Operation... 576
16.2.1 Memory Space 576
16.2.2 DMA Data Access 576
16.2.3 Addressing Modes.. 577
16.2.4 DMA Channel Control Packets 577
16.2.5 Priority Queue 581
16.2.6 Data Packing and Unpacking 583
16.2.7 DMA Request 586
16.2.8 Auto-Initiation.. 588
16.2.9 Interrupts... 588
16.2.10 Debugging.. 590
16.2.11 Power Management 590
16.2.12 FIFO Buffer... 590
16.2.13 Channel Chaining 591
16.2.14 Memory Protection 592
16.2.15 Parity Checking.. 593
16.2.16 Parity Testing 594
16.2.17 Initializing RAM with Parity... 594
16.3 Control Registers and Control Packets 595
16.3.1 Global Configuration Registers 597
16.3.2 Channel Configuration 642

17 External Memory Interface (EMIF) 648

17.1 Introduction 649
17.1.1 Purpose of the Peripheral 649
17.1.2 Features.. 649
17.1.3 Functional Block Diagram 650
17.2 EMIF Module Architecture 651
17.2.1 EMIF Clock Control... 651
17.2.2 EMIF Requests.. 651
17.2.3 EMIF Signal Descriptions.. 651
17.2.4 EMIF Signal Multiplexing Control... 652
17.2.5 SDRAM Controller and Interface 653
17.2.6 Asynchronous Controller and Interface 665

17.2.7 Data Bus Parking 677
17.2.8 Reset and Initialization Considerations 678
17.2.9 Interrupt Support 678
17.2.10 DMA Event Support 679
17.2.11 EMIF Signal Multiplexing 679
17.2.12 Memory Map 679
17.2.13 Priority and Arbitration 680
17.2.14 System Considerations 681
17.2.15 Power Management 682
17.2.16 Emulation Considerations 682
17.3 EMIF Registers 683
17.3.1 Module ID Register (MIDR) 683
17.3.2 Asynchronous Wait Cycle Configuration Register (AWCC) 684
17.3.3 SDRAM Configuration Register (SDCR) 685
17.3.4 SDRAM Refresh Control Register (SDRCR) 686
17.3.5 Asynchronous *n* Configuration Registers (CE2CFG-CE5CFG) 687
17.3.6 SDRAM Timing Register (SDTIMR) 688
17.3.7 SDRAM Self Refresh Exit Timing Register (SDSRETR) 689
17.3.8 EMIF Interrupt Raw Register (INTRAW) 690
17.3.9 EMIF Interrupt Masked Register (INTMSK) 691
17.3.10 EMIF Interrupt Mask Set Register (INTMSKSET) 692
17.3.11 EMIF Interrupt Mask Clear Register (INTMSKCLR) 693
17.3.12 Page Mode Control Register (PMCR) 694
17.4 Example Configuration 695
17.4.1 Hardware Interface 695
17.4.2 Software Configuration 695
18 Parameter Overlay Module (POM) 704
18.1 Introduction 705
18.1.1 Main Features 705
18.1.2 Parameter Overlay Module (POM) Considerations 705
18.1.3 Block Diagram 706
18.2 Module Operation 706
18.2.1 Decode Regions 707
18.2.2 Bus Errors on Accesses via POM 707
18.3 POM Control Registers 708
18.3.1 POM Global Control Register (POMGLBCTRL) 709
18.3.2 POM Revision ID (POMREV) 710
18.3.3 POM Clock Gate Control Register (POMCLKCTRL) 710
18.3.4 POM Status Register (POMFLG) 711
18.3.5 POM Program Region Start Address Register x (POMPROGSTARTx) 712
18.3.6 POM Overlay Region Start Address Register x (POMOVLSTARTx) 712
18.3.7 POM Region Size Register x (POMREGSIZEx) 713
18.3.8 POM Integration Control Register (POMITCTRL) 713
18.3.9 POM Claim Set Register (POMCLAIMSET) 714
18.3.10 POM Claim Clear Register (POMCLAIMCLR) 715
18.3.11 POM Lock Access Register (POMLOCKACCESS) 716
18.3.12 POM Lock Status Register (POMLOCKSTATUS) 716
18.3.13 POM Authentication Status Register (POMAUTHSTATUS) 716
18.3.14 POM Device ID Register (POMDEVID) 717
18.3.15 POM Device Type Register (POMDEVTYPE) 717
18.3.16 POM Peripheral ID 4 Register (POMPERIPHERALID4) 718
18.3.17 POM Peripheral ID 5 Register (POMPERIPHERALID5) 718
18.3.18 POM Peripheral ID 6 Register (POMPERIPHERALID6) 719

18.3.19 POM Peripheral ID 7 Register (POMPERIPHERALID7) 719
18.3.20 POM Peripheral ID 0 Register (POMPERIPHERALID0) 720
18.3.21 POM Peripheral ID 1 Register (POMPERIPHERALID1) 720
18.3.22 POM Peripheral ID 2 Register (POMPERIPHERALID2) 721
18.3.23 POM Peripheral ID 3 Register (POMPERIPHERALID3) 721
18.3.24 POM Component ID 0 Register (POMCOMPONENTID0) 722
18.3.25 POM Component ID 1 Register (POMCOMPONENTID1) 722
18.3.26 POM Component ID 2 Register (POMCOMPONENTID2) 723
18.3.27 POM Component ID 3 Register (POMCOMPONENTID3) 723

19 Enhanced Pulse Width Modulator (ePWM) Module 724

19.1 Introduction 725
19.1.1 Submodule Overview 725
19.1.2 Register Mapping 728
19.2 ePWM Submodules 729
19.2.1 Overview 729
19.2.2 Time-Base (TB) Submodule 731
19.2.3 Counter-Compare (CC) Submodule 742
19.2.4 Action-Qualifier (AQ) Submodule 748
19.2.5 Dead-Band Generator (DB) Submodule 762
19.2.6 PWM-Chopper (PC) Submodule 767
19.2.7 Trip-Zone (TZ) Submodule 771
19.2.8 Event-Trigger (ET) Submodule 777
19.2.9 Digital Compare (DC) Submodule 782
19.2.10 Proper Interrupt Initialization Procedure 788
19.3 Application Examples 788
19.3.1 Overview of Multiple Modules 788
19.3.2 Key Configuration Capabilities 789
19.3.3 Controlling Multiple Buck Converters With Independent Frequencies 790
19.3.4 Controlling Multiple Buck Converters With Same Frequencies 793
19.3.5 Controlling Multiple Half H-Bridge (HHB) Converters 796
19.3.6 Controlling Dual 3-Phase Inverters for Motors (ACI and PMSM) 798
19.3.7 Practical Applications Using Phase Control Between PWM Modules 801
19.4 EPWM_MODULE_CONTROL_AND_STATUS_REGISTER Registers 803
19.4.1 TBCTL Register (Offset = 0h) [reset = 83h] 805
19.4.2 TBSTS Register (Offset = 2h) [reset = 1h] 807
19.4.3 TBPHS Register (Offset = 6h) [reset = 0h] 808
19.4.4 TBCTR Register (Offset = 8h) [reset = 0h] 809
19.4.5 TBPRD Register (Offset = Ah) [reset = 0h] 810
19.4.6 CMPCTL Register (Offset = Eh) [reset = 0h] 811
19.4.7 CMPA Register (Offset = 12h) [reset = 0h] 813
19.4.8 CMPB Register (Offset = 14h) [reset = 0h] 814
19.4.9 AQCTLA Register (Offset = 16h) [reset = 0h] 815
19.4.10 AQCTLB Register (Offset = 18h) [reset = 0h] 817
19.4.11 AQSFRC Register (Offset = 1Ah) [reset = 0h] 819
19.4.12 AQCSFRC Register (Offset = 1Ch) [reset = 0h] 820
19.4.13 DBCTL Register (Offset = 1Eh) [reset = 0h] 821
19.4.14 DBRED Register (Offset = 20h) [reset = 0h] 823
19.4.15 DBFED Register (Offset = 22h) [reset = 0h] 824
19.4.16 TZSEL Register (Offset = 24h) [reset = 0h] 825
19.4.17 TZDCSEL Register (Offset = 26h) [reset = 0h] 827
19.4.18 TZCTL Register (Offset = 28h) [reset = 0h] 828
19.4.19 TZEINT Register (Offset = 2Ah) [reset = 0h] 829
19.4.20 TZFLG Register (Offset = 2Ch) [reset = 0h] 830

19.4.21 TZCLR Register (Offset = 2Eh) [reset = 0h] 831
19.4.22 TZFRC Register (Offset = 30h) [reset = 0h] 832
19.4.23 ETSEL Register (Offset = 32h) [reset = 0h] 833
19.4.24 ETPS Register (Offset = 34h) [reset = 0h] 835
19.4.25 ETFLG Register (Offset = 36h) [reset = 0h] 837
19.4.26 ETCLR Register (Offset = 38h) [reset = 0h] 838
19.4.27 ETFRC Register (Offset = 3Ah) [reset = 0h] 839
19.4.28 PCCTL Register (Offset = 3Ch) [reset = 0h] 840
19.4.29 DCTRIPSEL Register (Offset = 60h) [reset = 0h] 841
19.4.30 DCACTL Register (Offset = 62h) [reset = 0h] 842
19.4.31 DCBCTL Register (Offset = 64h) [reset = 0h] 843
19.4.32 DCFCTL Register (Offset = 66h) [reset = 0h] 844
19.4.33 DCCAPCTL Register (Offset = 68h) [reset = 0h] 845
19.4.34 DCFOFFSET Register (Offset = 6Ah) [reset = 0h] 846
19.4.35 DCFOFFSETCNT Register (Offset = 6Ch) [reset = 0h] 847
19.4.36 DCFWINDOW Register (Offset = 6Eh) [reset = 0h] 848
19.4.37 DCFWINDOWCNT Register (Offset = 70h) [reset = 0h] 849
19.4.38 DCCAP Register (Offset = 72h) [reset = 0h] 850

20 Enhanced Capture (eCAP) Module 851

20.1 Introduction 852
20.2 Description 852
20.3 Capture and APWM Operating Mode 853
20.4 Capture Mode Description 854
20.4.1 Event Prescaler 855
20.4.2 Edge Polarity Select and Qualifier 855
20.4.3 Continuous/One-Shot Control 856
20.4.4 32-Bit Counter and Phase Control 856
20.4.5 CAP1-CAP4 Registers 857
20.4.6 Interrupt Control 857
20.4.7 Shadow Load and Lockout Control 858
20.4.8 APWM Mode Operation 859
20.5 ECAP Registers 860
20.5.1 TSCTR Register (Offset = 0h) [reset = 0h] 861
20.5.2 CTRPHS Register (Offset = 4h) [reset = 0h] 862
20.5.3 CAP1 Register (Offset = 8h) [reset = 0h] 863
20.5.4 CAP2 Register (Offset = Ch) [reset = 0h] 864
20.5.5 CAP3 Register (Offset = 10h) [reset = 0h] 865
20.5.6 CAP4 Register (Offset = 14h) [reset = 0h] 866
20.5.7 ECCTL1 Register (Offset = 28h) [reset = 0h] 867
20.5.8 ECCTL2 Register (Offset = 2Ah) [reset = 2h] 869
20.5.9 ECEINT Register (Offset = 2Ch) [reset = 0h] 871
20.5.10 ECFLG Register (Offset = 2Eh) [reset = 0h] 872
20.5.11 ECCLR Register (Offset = 30h) [reset = 0h] 873
20.5.12 ECFRC Register (Offset = 32h) [reset = 0h] 874
20.6 Application of the ECAP Module 875
20.6.1 Example 1 - Absolute Time-Stamp Operation Rising Edge Trigger 876
20.6.2 Example 2 - Absolute Time-Stamp Operation Rising and Falling Edge Trigger 878
20.6.3 Example 3 - Time Difference (Delta) Operation Rising Edge Trigger 880
20.6.4 Example 4 - Time Difference (Delta) Operation Rising and Falling Edge Trigger 882
20.7 Application of the APWM Mode 884
20.7.1 Simple PWM Generation (Independent Channel/s) 884

21 Enhanced Quadrature Encoder Pulse (eQEP) Module 885

21.1 Introduction 886

21.2 Description 888
21.2.1 EQEP Inputs 888
21.2.2 Functional Description 889
21.2.3 eQEP Memory Map 890
21.3 Quadrature Decoder Unit (QDU) 891
21.3.1 Position Counter Input Modes 892
21.3.2 eQEP Input Polarity Selection 894
21.3.3 Position-Compare Sync Output 894
21.4 Position Counter and Control Unit (PCCU) 894
21.4.1 Position Counter Operating Modes 894
21.4.2 Position Counter Latch 897
21.4.3 Position Counter Initialization 899
21.4.4 eQEP Position-compare Unit 899
21.5 eQEP Edge Capture Unit 901
21.6 eQEP Watchdog 904
21.7 Unit Timer Base 904
21.8 eQEP Interrupt Structure 905
21.9 EQEP Registers 906
21.9.1 QPOSCNT Register (Offset = 0h) [reset = 0h] 907
21.9.2 QPOSINIT Register (Offset = 4h) [reset = 0h] 908
21.9.3 QPOSMAX Register (Offset = 8h) [reset = 0h] 909
21.9.4 QPOSCMP Register (Offset = Ch) [reset = 0h] 910
21.9.5 QPOSILAT Register (Offset = 10h) [reset = 0h] 911
21.9.6 QPOSSLAT Register (Offset = 14h) [reset = 0h] 912
21.9.7 QPOSLAT Register (Offset = 18h) [reset = 0h] 913
21.9.8 QUTMR Register (Offset = 1Ch) [reset = 0h] 914
21.9.9 QUPRD Register (Offset = 20h) [reset = 0h] 915
21.9.10 QWDTMR Register (Offset = 24h) [reset = 0h] 916
21.9.11 QWDPRD Register (Offset = 26h) [reset = 0h] 917
21.9.12 QDECCTL Register (Offset = 28h) [reset = 0h] 918
21.9.13 QEPCTL Register (Offset = 2Ah) [reset = 0h] 919
21.9.14 QCAPCTL Register (Offset = 2Ch) [reset = 0h] 921
21.9.15 QPOSCTL Register (Offset = 2Eh) [reset = 0h] 922
21.9.16 QEINT Register (Offset = 30h) [reset = 0h] 923
21.9.17 QFLG Register (Offset = 32h) [reset = 0h] 924
21.9.18 QCLR Register (Offset = 34h) [reset = 0h] 926
21.9.19 QFRC Register (Offset = 36h) [reset = 0h] 928
21.9.20 QEPSTS Register (Offset = 38h) [reset = 8Eh] 929
21.9.21 QCTMR Register (Offset = 3Ah) [reset = 0h] 930
21.9.22 QCPRD Register (Offset = 3Ch) [reset = 0h] 931
21.9.23 QCTMRLAT Register (Offset = 3Eh) [reset = 0h] 932
21.9.24 QCPRDLAT Register (Offset = 40h) [reset = 0h] 933
22 Analog To Digital Converter (ADC) Module 934
22.1 Overview 935
22.2 Introduction 936
22.2.1 Input Multiplexor 937
22.2.2 Self-Test and Calibration Cell 937
22.2.3 Analog-to-Digital Converter Core 937
22.2.4 Sequencer 938
22.2.5 Conversion Groups 938
22.3 Basic Features and Usage of the ADC 939
22.3.1 How to Select Between 12-bit and 10-bit Resolutions 939
22.3.2 How to Set Up the ADCLK Speed 939

22.3.3 How to Set Up the Input Channel Acquisition Time 939
22.3.4 How to Select an Input Channel for Conversion 939
22.3.5 How to Select Between Single Conversion Sequence or Continuous Conversions 940
22.3.6 How to Start a Conversion 940
22.3.7 How to Know When the Group Conversion is Completed 940
22.3.8 How Results are Stored in the Results' Memory 941
22.3.9 How to Read the Results from the Results' Memory 941
22.3.10 How to Stop a Conversion 945
22.3.11 Example Sequence for Basic Configuration of ADC Module 945
22.4 Advanced Conversion Group Configuration Options 946
22.4.1 Group Trigger Options 947
22.4.2 Analog Input Channel Selection Mode Options 947
22.4.3 Single or Continuous Conversion Modes 951
22.4.4 Conversion Group Freeze Capability 951
22.4.5 Conversion Group Memory Overrun Option 952
22.4.6 Response on Writing Non-Zero Value to Conversion Group's Channel Select Register 952
22.4.7 Conversion Result Size on Reading: 8-bit, 10-bit, or 12-bit 953
22.4.8 Option to Read Group Channel Id Along With Conversion Result 953
22.5 ADC Module Basic Interrupts 954
22.5.1 Group Conversion End Interrupt 954
22.5.2 Group Memory Threshold Interrupt 954
22.5.3 Group Memory Overrun Interrupt 954
22.6 ADC Module DMA Requests 955
22.6.1 DMA Request for Each Conversion Result Written to the Results' Memory 955
22.6.2 DMA Request for a Fixed Number of Conversion Results 955
22.7 ADC Magnitude Threshold Interrupts 956
22.7.1 Magnitude Threshold Interrupt Configuration 956
22.7.2 Magnitude Threshold Interrupt Comparison Mask Configuration 956
22.7.3 Magnitude Threshold Interrupt Enable / Disable Control 956
22.7.4 Magnitude Threshold Interrupt Flags 956
22.7.5 Magnitude Threshold Interrupt Offset Register 956
22.8 ADC Special Modes 957
22.8.1 ADC Error Calibration Mode 957
22.8.2 ADC Self-Test Mode 961
22.8.3 ADC Power-Down Mode 962
22.8.4 ADC Sample Capacitor Discharge Mode 963
22.9 ADC Results' RAM Special Features 964
22.9.1 ADC Results' RAM Auto-Initialization 964
22.9.2 ADC Results' RAM Test Mode 964
22.9.3 ADC Results' RAM Parity 964
22.10 ADEVT Pin General Purpose I/O Functionality 965
22.10.1 GPIO Functionality 965
22.10.2 Summary 966
22.11 ADC_CONTROL Registers 967
22.11.1 ADRSTCR Register (Offset = 0h) [reset = 0h] 970
22.11.2 ADOPMODECR Register (Offset = 4h) [reset = 00140000h] 971
22.11.3 ADCLOCKCR Register (Offset = 8h) [reset = 0h] 973
22.11.4 ADCALCR Register (Offset = Ch) [reset = 0h] 974
22.11.5 ADEVMODECR Register (Offset = 10h) [reset = 0h] 976
22.11.6 ADG1MODECR Register (Offset = 14h) [reset = 0h] 979
22.11.7 ADG2MODECR Register (Offset = 18h) [reset = 0h] 982
22.11.8 ADEVSRC Register (Offset = 1Ch) [reset = 0h] 985
22.11.9 ADG1SRC Register (Offset = 20h) [reset = 0h] 986

22.11.10 ADG2SRC Register (Offset = 24h) [reset = 0h] 987
22.11.11 ADEVINTENA Register (Offset = 28h) [reset = 0h] 988
22.11.12 ADG1INTENA Register (Offset = 2Ch) [reset = 0h]........ 989
22.11.13 ADG2INTENA Register (Offset = 30h) [reset = 0h] 990
22.11.14 ADEVINTFLG Register (Offset = 34h) [reset = 4h] 991
22.11.15 ADG1INTFLG Register (Offset = 38h) [reset = 4h] 992
22.11.16 ADG2INTFLG Register (Offset = 3Ch) [reset = 4h]........ 993
22.11.17 ADEVTHRINTCR Register (Offset = 40h) [reset = 0h] 994
22.11.18 ADG1THRINTCR Register (Offset = 44h) [reset = 0h] 995
22.11.19 ADG2THRINTCR Register (Offset = 48h) [reset = 0h] 996
22.11.20 ADEVDMACR Register (Offset = 4Ch) [reset = 0h]........ 997
22.11.21 ADG1DMACR Register (Offset = 50h) [reset = 0h] 999
22.11.22 ADG2DMACR Register (Offset = 54h) [reset = 0h]........ 1001
22.11.23 ADBNDCR Register (Offset = 58h) [reset = 0h]........ 1003
22.11.24 ADBNDEND Register (Offset = 5Ch) [reset = 0h] 1004
22.11.25 ADEVSAMP Register (Offset = 60h) [reset = 0h]........ 1005
22.11.26 ADG1SAMP Register (Offset = 64h) [reset = 0h]........ 1006
22.11.27 ADG2SAMP Register (Offset = 68h) [reset = 0h]........ 1007
22.11.28 ADEVSR Register (Offset = 6Ch) [reset = 8h] 1008
22.11.29 ADG1SR Register (Offset = 70h) [reset = 8h]........ 1009
22.11.30 ADG2SR Register (Offset = 74h) [reset = 8h]........ 1010
22.11.31 ADEVSEL Register (Offset = 78h) [reset = 0h] 1011
22.11.32 ADG1SEL Register (Offset = 7Ch) [reset = 0h]........ 1012
22.11.33 ADG2SEL Register (Offset = 80h) [reset = 0h] 1013
22.11.34 ADCALR Register (Offset = 84h) [reset = 0h]........ 1014
22.11.35 ADSMSTATE Register (Offset = 88h) [reset = 0h] 1015
22.11.36 ADLASTCONV Register (Offset = 8Ch) [reset = X] 1016
22.11.37 ADEVBUFFER_0 to ADEVBUFFER_7 Register (Offset = 90h to ACh) [reset = 80000000h] ... 1017
22.11.38 ADG1BUFFER_0 to ADG1BUFFER_7 Register (Offset = B0h to CCh) [reset = X] 1019
22.11.39 ADG2BUFFER_0 to ADG2BUFFER_7 Register (Offset = D0h to ECh) [reset = X] 1021
22.11.40 ADEVEMUBUFFER Register (Offset = F0h) [reset = X]........ 1023
22.11.41 ADG1EMUBUFFER Register (Offset = F4h) [reset = X]........ 1025
22.11.42 ADG2EMUBUFFER Register (Offset = F8h) [reset = X]........ 1027
22.11.43 ADEVTDIR Register (Offset = FCh) [reset = 0h]........ 1029
22.11.44 ADEVTOUT Register (Offset = 100h) [reset = X]........ 1030
22.11.45 ADEVTIN Register (Offset = 104h) [reset = X]........ 1031
22.11.46 ADEVTSET Register (Offset = 108h) [reset = X] 1032
22.11.47 ADEVTCLR Register (Offset = 10Ch) [reset = X]........ 1033
22.11.48 ADEVTPDR Register (Offset = 110h) [reset = 0h]........ 1034
22.11.49 ADEVTPDIS Register (Offset = 114h) [reset = 0h] 1035
22.11.50 ADEVTPSEL Register (Offset = 118h) [reset = 0h] 1036
22.11.51 ADEVSAMPDISEN Register (Offset = 11Ch) [reset = 0h]........ 1037
22.11.52 ADG1SAMPDISEN Register (Offset = 120h) [reset = 0h] 1038
22.11.53 ADG2SAMPDISEN Register (Offset = 124h) [reset = 0h] 1039
22.11.54 ADMAGINT1CR Register (Offset = 128h) [reset = 0h]........ 1040
22.11.55 ADMAGINT1MASK Register (Offset = 12Ch) [reset = 0h]........ 1042
22.11.56 ADMAGINT2CR Register (Offset = 130h) [reset = 0h]........ 1043
22.11.57 ADMAGINT2MASK Register (Offset = 134h) [reset = 0h] 1045
22.11.58 ADMAGINT3CR Register (Offset = 138h) [reset = 0h]........ 1046
22.11.59 ADMAGINT3MASK Register (Offset = 13Ch) [reset = 0h]........ 1048
22.11.60 ADMAGINTENASET Register (Offset = 158h) [reset = 0h] 1049
22.11.61 ADMAGINTENACLR Register (Offset = 15Ch) [reset = 0h]........ 1050
22.11.62 ADMAGINTFLG Register (Offset = 160h) [reset = 0h]........ 1051

22.11.63 ADMAGINTOFF Register (Offset = 164h) [reset = 0h] 1052
22.11.64 ADEVFIFORESETCR Register (Offset = 168h) [reset = 0h] 1053
22.11.65 ADG1FIFORESETCR Register (Offset = 16Ch) [reset = 0h] 1054
22.11.66 ADG2FIFORESETCR Register (Offset = 170h) [reset = 0h] 1055
22.11.67 ADEVRAMWRADDR Register (Offset = 174h) [reset = 0h] 1056
22.11.68 ADG1RAMWRADDR Register (Offset = 178h) [reset = 0h] 1057
22.11.69 ADG2RAMWRADDR Register (Offset = 17Ch) [reset = 0h] 1058
22.11.70 ADPARCR Register (Offset = 180h) [reset = 5h] 1059
22.11.71 ADPARADDR Register (Offset = 184h) [reset = X] 1060
22.11.72 ADPWRUPDLYCTRL Register (Offset = 188h) [reset = 0h] 1061
22.11.73 ADEVCHNSELMODECTRL Register (Offset = 190h) [reset = 5h] 1062
22.11.74 ADG1CHNSELMODECTRL Register (Offset = 194h) [reset = 5h] 1063
22.11.75 ADG2CHNSELMODECTRL Register (Offset = 198h) [reset = 5h] 1064
22.11.76 ADEVCURRCOUNT Register (Offset = 19Ch) [reset = 0h] 1065
22.11.77 ADEVMAXCOUNT Register (Offset = 1A0h) [reset = 0h] 1066
22.11.78 ADG1CURRCOUNT Register (Offset = 1A4h) [reset = 0h] 1067
22.11.79 ADG1MAXCOUNT Register (Offset = 1A8h) [reset = 0h] 1068
22.11.80 ADG2CURRCOUNT Register (Offset = 1ACh) [reset = 0h] 1069
22.11.81 ADG2MAXCOUNT Register (Offset = 1B0h) [reset = 0h] 1070

23 High-End Timer (N2HET) Module 1071

23.1 Overview 1072
23.1.1 Features 1072
23.1.2 Major Advantages 1072
23.1.3 Block Diagram 1073
23.1.4 Timer Module Structure and Execution 1074
23.1.5 Performance 1075
23.1.6 N2HET Compared to NHET 1075
23.1.7 NHET and N2HET Compared to HET 1075
23.1.8 Instructions Features 1076
23.1.9 Program Usage 1076
23.2 N2HET Functional Description 1076
23.2.1 Specialized Timer Micromachine 1076
23.2.2 N2HET RAM Organization 1081
23.2.3 Time Base 1084
23.2.4 Host Interface 1087
23.2.5 I/O Control 1088
23.2.6 Suppression Filters 1104
23.2.7 Interrupts and Exceptions 1105
23.2.8 Hardware Priority Scheme 1106
23.2.9 N2HET Requests to DMA and HTU 1108
23.3 Angle Functions 1108
23.3.1 Software Angle Generator 1108
23.3.2 Hardware Angle Generator (HWAG) 1113
23.4 N2HET Control Registers 1135
23.4.1 Global Configuration Register (HETGCR) 1136
23.4.2 Prescale Factor Register (HETPFR) 1138
23.4.3 N2HET Current Address Register (HETADDR) 1139
23.4.4 Offset Index Priority Level 1 Register (HETOFF1) 1139
23.4.5 Offset Index Priority Level 2 Register (HETOFF2) 1140
23.4.6 Interrupt Enable Set Register (HETINTENAS) 1141
23.4.7 Interrupt Enable Clear Register (HETINTENAC) 1141
23.4.8 Exception Control Register 1 (HETEXC1) 1142
23.4.9 Exception Control Register 2 (HETEXC2) 1143

23.4.10 Interrupt Priority Register (HETPRY) 1144
23.4.11 Interrupt Flag Register (HETFLG) 1144
23.4.12 AND Share Control Register (HETAND) 1145
23.4.13 HR Share Control Register (HETHRSH) 1146
23.4.14 XOR Share Control Register (HETXOR) 1147
23.4.15 Request Enable Set Register (HETREQENS) 1148
23.4.16 Request Enable Clear Register (HETREQENC) 1148
23.4.17 Request Destination Select Register (HETREQDS) 1149
23.4.18 NHET Direction Register (HETDIR) 1150
23.4.19 N2HET Data Input Register (HETDIN) 1151
23.4.20 N2HET Data Output Register (HETDOUT) 1151
23.4.21 NHET Data Set Register (HETDSET) 1152
23.4.22 N2HET Data Clear Register (HETDCLR) 1152
23.4.23 N2HET Open Drain Register (HETPDR) 1153
23.4.24 N2HET Pull Disable Register (HETPULDIS) 1153
23.4.25 N2HET Pull Select Register (HETPSL) 1154
23.4.26 Parity Control Register (HETPCR) 1155
23.4.27 Parity Address Register (HETPAR) 1156
23.4.28 Parity Pin Register (HETPPR) 1157
23.4.29 Suppression Filter Preload Register (HETSFPRLD) 1158
23.4.30 Suppression Filter Enable Register (HETSFENA) 1158
23.4.31 Loop Back Pair Select Register (HETLBPSEL) 1159
23.4.32 Loop Back Pair Direction Register (HETLBPDIR) 1160
23.4.33 N2HET Pin Disable Register (HETPINDIS) 1161
23.5 HWAG Registers 1162
23.5.1 HWAG Pin Select Register (HWAPINSEL) 1163
23.5.2 HWAG Global Control Register 0 (HWAGCR0) 1164
23.5.3 HWAG Global Control Register 1 (HWAGCR1) 1164
23.5.4 HWAG Global Control Register 2 (HWAGCR2) 1165
23.5.5 HWAG Interrupt Enable Set Register (HWAENASET) 1166
23.5.6 HWAG Interrupt Enable Clear Register (HWAENACLR) 1167
23.5.7 HWAG Interrupt Level Set Register (HWALVLSET) 1168
23.5.8 HWAG Interrupt Level Clear Register (HWALVLCLR) 1168
23.5.9 HWAG Interrupt Flag Register (HWAFLG) 1169
23.5.10 HWAG Interrupt Offset Register 0 (HWAOFF0) 1170
23.5.11 HWAG Interrupt Offset Register 1 (HWAOFF1) 1171
23.5.12 HWAG Angle Value Register (HWAACNT) 1172
23.5.13 HWAG Previous Tooth Period Value Register (HWAPCNT1) 1173
23.5.14 HWAG Current Tooth Period Value Register (HWAPCNT) 1173
23.5.15 HWAG Step Width Register (HWASTWD) 1174
23.5.16 HWAG Teeth Number Register (HWATHNB) 1175
23.5.17 HWAG Current Teeth Number Register (HWATHVL) 1175
23.5.18 HWAG Filter Register (HWAFIL) 1176
23.5.19 HWAG Filter Register 2 (HWAFIL2) 1176
23.5.20 HWAG Angle Increment Register (HWAANGI) 1177
23.6 Instruction Set 1178
23.6.1 Instruction Summary 1178
23.6.2 Abbreviations, Encoding Formats and Bits 1180
23.6.3 Instruction Description 1183
24 High-End Timer Transfer Unit (HTU) Module 1249
24.1 Overview 1250
24.1.1 Features 1250
24.2 Module Operation 1251

24.2.1 Data Transfers between Main RAM and N2HET RAM 1253
24.2.2 Arbitration of HTU Elements and Frames 1257
24.2.3 Conditions for Frame Transfer Interruption 1258
24.2.4 HTU Overload and Request Lost Detection 1258
24.2.5 Memory Protection 1261
24.2.6 Control Packet RAM Parity Checking 1261
24.3 Use Cases 1263
24.3.1 Example: Single Element Transfer with One Trigger Request 1263
24.3.2 Example: Multiple Element Transfer with One Trigger Request 1263
24.3.3 Example: 64-Bit-Transfer of Control Field and Data Fields 1265
24.4 HTU Control Registers 1266
24.4.1 Global Control Register (HTU GC) 1267
24.4.2 Control Packet Enable Register (HTU CPENA) 1268
24.4.3 Control Packet (CP) Busy Register 0 (HTU BUSY0) 1269
24.4.4 Control Packet (CP) Busy Register 1 (HTU BUSY1) 1270
24.4.5 Control Packet (CP) Busy Register 2 (HTU BUSY2) 1270
24.4.6 Control Packet (CP) Busy Register 3 (HTU BUSY3) 1271
24.4.7 Active Control Packet and Error Register (HTU ACPE) 1271
24.4.8 Request Lost and Bus Error Control Register (HTU RLBECTRL) 1273
24.4.9 Buffer Full Interrupt Enable Set Register (HTU BFINTS) 1274
24.4.10 Buffer Full Interrupt Enable Clear Register (HTU BFINTC) 1274
24.4.11 Interrupt Mapping Register (HTU INTMAP) 1275
24.4.12 Interrupt Offset Register 0 (HTU INTOFF0) 1276
24.4.13 Interrupt Offset Register 1 (HTU INTOFF1) 1277
24.4.14 Buffer Initialization Mode Register (HTU BIM) 1278
24.4.15 Request Lost Flag Register (HTU RLOSTFL) 1280
24.4.16 Buffer Full Interrupt Flag Register (HTU BFINTFL) 1280
24.4.17 BER Interrupt Flag Register (HTU BERINTFL) 1281
24.4.18 Memory Protection 1 Start Address Register (HTU MP1S) 1282
24.4.19 Memory Protection 1 End Address Register (HTU MP1E) 1282
24.4.20 Debug Control Register (HTU DCTRL) 1283
24.4.21 Watch Point Register (HTU WPR) 1284
24.4.22 Watch Mask Register (HTU WMR) 1284
24.4.23 Module Identification Register (HTU ID) 1285
24.4.24 Parity Control Register (HTU PCR) 1286
24.4.25 Parity Address Register (HTU PAR) 1287
24.4.26 Memory Protection Control and Status Register (HTU MPCS) 1288
24.4.27 Memory Protection Start Address Register 0 (HTU MP0S) 1291
24.4.28 Memory Protection End Address Register (HTU MP0E) 1291
24.5 Double Control Packet Configuration Memory 1292
24.5.1 Initial Full Address A Register (HTU IFADDRA) 1293
24.5.2 Initial Full Address B Register (HTU IFADDRB) 1293
24.5.3 Initial N2HET Address and Control Register (HTU IHADDRCT) 1294
24.5.4 Initial Transfer Count Register (HTU ITCOUNT) 1295
24.5.5 Current Full Address A Register (HTU CFADDRA) 1296
24.5.6 Current Full Address B Register (HTU CFADDRB) 1297
24.5.7 Current Frame Count Register (HTU CFCOUNT) 1298
24.6 Examples 1299
24.6.1 Application Examples for Setting the Transfer Modes of CP A and B of a DCP 1299
24.6.2 Software Example Sequence Assuming Circular Mode for Both CP A and B 1299
24.6.3 Example of an Interrupt Dispatch Flow for a Request Lost Interrupt 1300
25 General-Purpose Input/Output (GIO) Module 1301
25.1 Overview 1302

25.2 Quick Start Guide 1303
25.3 Functional Description of GIO Module 1305
25.3.1 I/O Functions 1305
25.3.2 Interrupt Function 1306
25.3.3 GIO Block Diagram 1306
25.4 Device Modes of Operation 1308
25.4.1 Emulation Mode 1308
25.4.2 Power-Down Mode (Low-Power Mode) 1308
25.5 GIO Control Registers 1309
25.5.1 GIO Global Control Register (GIOGCR0) 1310
25.5.2 GIO Interrupt Detect Register (GIOINTDET) 1311
25.5.3 GIO Interrupt Polarity Register (GIOPOL) 1312
25.5.4 GIO Interrupt Enable Registers (GIOENASET and GIOENACLR) 1313
25.5.5 GIO Interrupt Priority Registers (GIOLVLSET and GIOLVLCLR) 1315
25.5.6 GIO Interrupt Flag Register (GIOFLG) 1318
25.5.7 GIO Offset Register 1 (GIOOFF1) 1319
25.5.8 GIO Offset B Register (GIOOFF2) 1320
25.5.9 GIO Emulation A Register (GIOEMU1) 1321
25.5.10 GIO Emulation B Register (GIOEMU2) 1322
25.5.11 GIO Data Direction Registers (GIODIR[A-B]) 1323
25.5.12 GIO Data Input Registers (GIODIN[A-B]) 1323
25.5.13 GIO Data Output Registers (GIODOUT[A-B]) 1324
25.5.14 GIO Data Set Registers (GIODSET[A-B]) 1324
25.5.15 GIO Data Clear Registers (GIODCLR[A-B]) 1325
25.5.16 GIO Open Drain Registers (GIOPDR[A-B]) 1325
25.5.17 GIO Pull Disable Registers (GIOPULDIS[A-B]) 1326
25.5.18 GIO Pull Select Registers (GIOPSL[A-B]) 1326
25.6 I/O Control Summary 1327

26 Controller Area Network (DCAN) Module 1328

26.1 Overview 1329
26.1.1 Features 1329
26.1.2 Functional Description 1329
26.2 CAN Blocks 1330
26.2.1 CAN Core 1330
26.2.2 Message RAM 1330
26.2.3 Message Handler 1330
26.2.4 Message RAM Interface 1331
26.2.5 Register and Message Object Access 1331
26.2.6 Dual Clock Source 1331
26.3 CAN Bit Timing 1332
26.3.1 Bit Time and Bit Rate 1332
26.3.2 DCAN Bit Timing Registers 1334
26.4 CAN Module Configuration 1336
26.4.1 DCAN RAM Initialization through Hardware 1336
26.4.2 CAN Module Initialization 1336
26.5 Message RAM 1338
26.5.1 Structure of Message Objects 1338
26.5.2 Addressing Message Objects in RAM 1340
26.5.3 Message RAM Representation in Debug/Suspend Mode 1341
26.5.4 Message RAM Representation in Direct Access Mode 1341
26.6 Message Interface Register Sets 1342
26.6.1 Message Interface Register Sets 1 and 2 1342
26.6.2 Using Message Interface Register Sets 1 and 2 1343

26.6.3 Message Interface Register 3 1344
26.7 Message Object Configurations 1345
26.7.1 Configuration of a Transmit Object for Data Frames 1345
26.7.2 Configuration of a Transmit Object for Remote Frames 1345
26.7.3 Configuration of a Single Receive Object for Data Frames 1345
26.7.4 Configuration of a Single Receive Object for Remote Frames 1346
26.7.5 Configuration of a FIFO Buffer 1346
26.7.6 Reconfiguration of Message Objects for the Reception of Frames 1346
26.7.7 Reconfiguration of Message Objects for the Transmission of Frames 1346
26.8 Message Handling 1347
26.8.1 Message Handler Overview 1347
26.8.2 Receive/Transmit Priority 1347
26.8.3 Transmission of Messages in Event Driven CAN Communication 1348
26.8.4 Updating a Transmit Object 1348
26.8.5 Changing a Transmit Object 1348
26.8.6 Acceptance Filtering of Received Messages 1349
26.8.7 Reception of Data Frames 1349
26.8.8 Reception of Remote Frames 1349
26.8.9 Reading Received Messages 1349
26.8.10 Requesting New Data for a Receive Object 1350
26.8.11 Storing Received Messages in FIFO Buffers 1350
26.8.12 Reading from a FIFO Buffer 1350
26.9 CAN Message Transfer 1352
26.9.1 Automatic Retransmission 1352
26.9.2 Auto-Bus-On 1353
26.10 Interrupt Functionality 1353
26.10.1 Message Object Interrupts 1353
26.10.2 Status Change Interrupts 1354
26.10.3 Error Interrupts 1354
26.11 Global Power-Down Mode 1355
26.11.1 Entering Global Power-Down Mode 1355
26.11.2 Wakeup From Global Power-Down Mode 1355
26.12 Local Power-Down Mode 1356
26.12.1 Entering Local Power-Down Mode 1356
26.12.2 Wakeup From Local Power-Down Mode 1356
26.13 GIO Support 1357
26.14 Test Modes 1358
26.14.1 Silent Mode 1358
26.14.2 Loop Back Mode 1359
26.14.3 External Loop Back Mode 1360
26.14.4 Loop Back Combined with Silent Mode 1361
26.14.5 Software Control of CAN_TX Pin 1361
26.15 Parity Check Mechanism 1362
26.15.1 Behavior on Parity Error 1362
26.15.2 Parity Testing 1362
26.16 Debug/Suspend Mode 1363
26.17 DCAN Control Registers 1363
26.17.1 CAN Control Register (DCAN CTL) 1365
26.17.2 Error and Status Register (DCAN ES) 1367
26.17.3 Error Counter Register (DCAN ERRC) 1369
26.17.4 Bit Timing Register (DCAN BTR) 1370
26.17.5 Interrupt Register (DCAN INT) 1371
26.17.6 Test Register (DCAN TEST) 1372

26.17.7 Parity Error Code Register (DCAN PERR) 1373
26.17.8 Core Release Register (DCAN REL) 1373
26.17.9 Auto-Bus-On Time Register (DCAN ABOTR) 1374
26.17.10 Transmission Request X Register (DCAN TXRQ X) 1374
26.17.11 Transmission Request Registers (DCAN TXRQ12 to DCAN TXRQ78) 1375
26.17.12 New Data X Register (DCAN NWDAT X) 1376
26.17.13 New Data Registers (DCAN NWDAT12 to DCAN NWDAT78) 1377
26.17.14 Interrupt Pending X Register (DCAN INTPND X) 1378
26.17.15 Interrupt Pending Registers (DCAN INTPND12 to DCAN INTPND78) 1379
26.17.16 Message Valid X Register (DCAN MSGVAL X) 1380
26.17.17 Message Valid Registers (DCAN MSGVAL12 to DCAN MSGVAL78) 1381
26.17.18 Interrupt Multiplexer Registers (DCAN INTMUX12 to DCAN INTMUX78) 1382
26.17.19 IF1/IF2 Command Registers (DCAN IF1CMD, DCAN IF2CMD) 1383
26.17.20 IF1/IF2 Mask Registers (DCAN IF1MSK, DCAN IF2MSK) 1386
26.17.21 IF1/IF2 Arbitration Registers (DCAN IF1ARB, DCAN IF2ARB) 1387
26.17.22 IF1/IF2 Message Control Registers (DCAN IF1MCTL, DCAN IF2MCTL) 1388
26.17.23 IF1/IF2 Data A and Data B Registers (DCAN IF1DATA/DATB, DCAN IF2DATA/DATB) 1390
26.17.24 IF3 Observation Register (DCAN IF3OBS) 1391
26.17.25 IF3 Mask Register (DCAN IF3MSK) 1393
26.17.26 IF3 Arbitration Register (DCAN IF3ARB) 1394
26.17.27 IF3 Message Control Register (DCAN IF3MCTL) 1395
26.17.28 IF3 Data A and Data B Registers (DCAN IF3DATA/DATB) 1396
26.17.29 IF3 Update Enable Registers (DCAN IF3UPD12 to IF3UPD78) 1397
26.17.30 CAN TX IO Control Register (DCAN TIOC) 1398
26.17.31 CAN RX IO Control Register (DCAN RIOC) 1399

27 Multi-Buffered Serial Peripheral Interface Module (MibSPI) with Parallel Pin Option (MibSPIP) . 1401

27.1 Overview 1402
27.1.1 Word Format Options 1402
27.1.2 Multi-buffering (Mib) Support 1402
27.1.3 Transmission Lock (Multi-Buffer Mode Master Only) 1403
27.2 Operating Modes 1403
27.2.1 Pin Configurations 1404
27.2.2 Data Handling 1404
27.2.3 Operation with $\overline{\text{SPICS}}$ 1407
27.2.4 Operation with $\overline{\text{SPIENA}}$ 1408
27.2.5 Five-Pin Operation (Hardware Handshaking) 1409
27.2.6 Data Formats 1410
27.2.7 Clocking Modes 1411
27.2.8 Data Transfer Example 1413
27.2.9 Decoded and Encoded Chip Select (Master Only) 1414
27.2.10 Variable Chip Select Setup and Hold Timing (Master Only) 1414
27.2.11 Hold Chip-Select Active 1414
27.2.12 Detection of Slave Desynchronization (Master Only) 1415
27.2.13 ENA Signal Time-Out (Master Only) 1416
27.2.14 Data-Length Error 1416
27.2.15 Parallel Mode (Multiple SIMO/SOMI Support, not available on all devices) 1416
27.2.16 Continuous Self-Test (Master/Slave) 1424
27.2.17 Half Duplex Mode 1424
27.3 Test Features 1424
27.3.1 Internal Loop-Back Test Mode (Master Only) 1424
27.3.2 Input/Output Loopback Test Mode 1425
27.4 General-Purpose I/O 1426
27.5 Low-Power Mode 1426

27.6 Interrupts 1427
27.6.1 Interrupts in Multi-Buffer Mode 1427
27.7 DMA Interface 1429
27.7.1 DMA in Multi-Buffer Mode 1429
27.8 Module Configuration 1430
27.8.1 Compatibility (SPI) Mode Configuration 1430
27.8.2 MibSPI Mode Configuration 1431
27.9 Control Registers 1432
27.9.1 SPI Global Control Register 0 (SPIGCR0) 1433
27.9.2 SPI Global Control Register 1 (SPIGCR1) 1434
27.9.3 SPI Interrupt Register (SPIINT0) 1435
27.9.4 SPI Interrupt Level Register (SPILVL) 1437
27.9.5 SPI Flag Register (SPIFLG) 1438
27.9.6 SPI Pin Control Register 0 (SPIPC0) 1441
27.9.7 SPI Pin Control Register 1 (SPIPC1) 1442
27.9.8 SPI Pin Control Register 2 (SPIPC2) 1444
27.9.9 SPI Pin Control Register 3 (SPIPC3) 1445
27.9.10 SPI Pin Control Register 4 (SPIPC4) 1446
27.9.11 SPI Pin Control Register 5 (SPIPC5) 1448
27.9.12 SPI Pin Control Register 6 (SPIPC6) 1449
27.9.13 SPI Pin Control Register 7 (SPIPC7) 1451
27.9.14 SPI Pin Control Register 8 (SPIPC8) 1452
27.9.15 SPI Transmit Data Register 0 (SPIDAT0) 1453
27.9.16 SPI Transmit Data Register 1 (SPIDAT1) 1455
27.9.17 SPI Receive Buffer Register (SPIBUF) 1458
27.9.18 SPI Emulation Register (SPIEMU) 1460
27.9.19 SPI Delay Register (SPIDELAY) 1460
27.9.20 SPI Default Chip Select Register (SPIDEF) 1463
27.9.21 SPI Data Format Registers (SPIFMT) 1464
27.9.22 Interrupt Vector 0 (INTVECT0) 1466
27.9.23 Interrupt Vector 1 (INTVECT1) 1467
27.9.24 SPI Pin Control Register 9 (SPIPC9) 1469
27.9.25 Parallel/Modulo Mode Control Register (SPIPMCTRL) 1470
27.9.26 Multi-buffer Mode Enable Register (MIBSPIE) 1473
27.9.27 TG Interrupt Enable Set Register (TGITENST) 1474
27.9.28 TG Interrupt Enable Clear Register (TGITENCR) 1475
27.9.29 Transfer Group Interrupt Level Set Register (TGITLVST) 1476
27.9.30 Transfer Group Interrupt Level Clear Register (TGITLVCR) 1477
27.9.31 Transfer Group Interrupt Flag Register (TGINTFLG) 1478
27.9.32 Tick Count Register (TICKCNT) 1479
27.9.33 Last TG End Pointer (LTGPEND) 1480
27.9.34 TGx Control Registers (TGxCTRL) 1481
27.9.35 DMA Channel Control Register (DMAxCTRL) 1484
27.9.36 DMAxCOUNT Register (ICOUNT) 1486
27.9.37 DMA Large Count (DMACNTLEN) 1487
27.9.38 Multi-buffer RAM Uncorrectable Parity Error Control Register (UERRCTRL) 1487
27.9.39 Multi-buffer RAM Uncorrectable Parity Error Status Register (UERRSTAT) 1488
27.9.40 RXRAM Uncorrectable Parity Error Address Register (UERRADDR1) 1489
27.9.41 TXRAM Uncorrectable Parity Error Address Register (UERRADDR0) 1490
27.9.42 RXRAM Overrun Buffer Address Register (RXOVRN_BUF_ADDR) 1491
27.9.43 I/O-Loopback Test Control Register (IOLPBKTSTCR) 1492
27.9.44 SPI Extended Prescale Register 1 (EXTENDED_PRESCALE1 for SPIFMT0 and SPIFMT1) 1494
27.9.45 SPI Extended Prescale Register 2 (EXTENDED_PRESCALE2 for SPIFMT2 and SPIFMT3) 1496

27.10 Multi-Buffer RAM 1498
27.10.1 Multi-Buffer RAM Auto Initialization 1499
27.10.2 Multi-Buffer RAM Register Summary 1499
27.10.3 Multi-Buffer RAM Transmit Data Register (TXRAM) 1500
27.10.4 Multi-Buffer RAM Receive Buffer Register (RXRAM) 1503
27.11 Parity Memory 1505
27.11.1 Example of Parity Memory Organization 1507
27.12 MibSPI Pin Timing Parameters 1508
27.12.1 Master Mode Timings for SPI/MibSPI 1508
27.12.2 Slave Mode Timings for SPI/MibSPI 1510
27.12.3 Master Mode Timing Parameter Details 1511
27.12.4 Slave Mode Timing Parameter Details 1511
28 Serial Communication Interface (SCI)/ Local Interconnect Network (LIN) Module 1512
28.1 Introduction and Features 1513
28.1.1 SCI Features 1513
28.1.2 LIN Features 1514
28.1.3 Block Diagram 1515
28.2 SCI Communication Formats 1518
28.2.1 SCI Frame Formats 1518
28.2.2 SCI Timing Mode 1519
28.2.3 SCI Baud Rate 1519
28.2.4 SCI Multiprocessor Communication Modes 1522
28.2.5 SCI Multi-Buffered Mode 1524
28.3 SCI Interrupts 1526
28.3.1 Transmit Interrupt 1527
28.3.2 Receive Interrupt 1527
28.3.3 WakeUp Interrupt 1527
28.3.4 Error Interrupts 1528
28.4 SCI DMA Interface 1529
28.4.1 Receive DMA Requests 1529
28.4.2 Transmit DMA Requests 1529
28.5 SCI Configurations 1530
28.5.1 Receiving Data 1530
28.5.2 Transmitting Data 1531
28.6 SCI Low-Power Mode 1532
28.6.1 Sleep Mode for Multiprocessor Communication 1532
28.7 LIN Communication Formats 1533
28.7.1 LIN Standards 1533
28.7.2 Message Frame 1534
28.7.3 Synchronizer 1536
28.7.4 Baud Rate 1536
28.7.5 Header Generation 1538
28.7.6 Extended Frames Handling 1542
28.7.7 Timeout Control 1543
28.7.8 TXRX Error Detector (TED) 1544
28.7.9 Message Filtering and Validation 1547
28.7.10 Receive Buffers 1549
28.7.11 Transmit Buffers 1550
28.8 LIN Interrupts 1551
28.9 LIN DMA Interface 1551
28.9.1 LIN Receive DMA Requests 1551
28.9.2 LIN Transmit DMA Requests 1551
28.10 LIN Configurations 1552

28.10.1 Receiving Data 1552
28.10.2 Transmitting Data 1553
28.11 Low-Power Mode 1554
28.11.1 Entering Sleep Mode 1554
28.11.2 Wakeup 1555
28.11.3 Wakeup Timeouts 1556
28.12 Emulation Mode 1556
28.13 SCI/LIN Control Registers 1557
28.13.1 SCI Global Control Register 0 (SCIGCR0) 1558
28.13.2 SCI Global Control Register 1 (SCIGCR1) 1559
28.13.3 SCI Global Control Register 2 (SCIGCR2) 1563
28.13.4 SCI Set Interrupt Register (SCISETINT) 1565
28.13.5 SCI Clear Interrupt Register (SCICLEARINT) 1568
28.13.6 SCI Set Interrupt Level Register (SCISETINTLVL) 1572
28.13.7 SCI Clear Interrupt Level Register (SCICLEARINTLVL) 1575
28.13.8 SCI Flags Register (SCIFLR) 1578
28.13.9 SCI Interrupt Vector Offset 0 (SCIINTVECT0) 1585
28.13.10 SCI Interrupt Vector Offset 1 (SCIINTVECT1) 1585
28.13.11 SCI Format Control Register (SCIFORMAT) 1586
28.13.12 Baud Rate Selection Register (BRS) 1587
28.13.13 SCI Data Buffers (SCIED, SCIRD, SCITD) 1588
28.13.14 SCI Pin I/O Control Register 0 (SCIPIO0) 1590
28.13.15 SCI Pin I/O Control Register 1 (SCIPIO1) 1591
28.13.16 SCI Pin I/O Control Register 2 (SCIPIO2) 1592
28.13.17 SCI Pin I/O Control Register 3 (SCIPIO3) 1593
28.13.18 SCI Pin I/O Control Register 4 (SCIPIO4) 1594
28.13.19 SCI Pin I/O Control Register 5 (SCIPIO5) 1595
28.13.20 SCI Pin I/O Control Register 6 (SCIPIO6) 1596
28.13.21 SCI Pin I/O Control Register 7 (SCIPIO7) 1597
28.13.22 SCI Pin I/O Control Register 8 (SCIPIO8) 1597
28.13.23 LIN Compare Register (LINCOMPARE) 1598
28.13.24 LIN Receive Buffer 0 Register (LINRD0) 1599
28.13.25 LIN Receive Buffer 1 Register (LINRD1) 1599
28.13.26 LIN Mask Register (LINMASK) 1600
28.13.27 LIN Identification Register (LINID) 1601
28.13.28 LIN Transmit Buffer 0 Register (LINTD0) 1602
28.13.29 LIN Transmit Buffer 1 Register (LINTD1) 1602
28.13.30 Maximum Baud Rate Selection Register (MBRS) 1603
28.13.31 Input/Output Error Enable (IODFTCTRL) Register 1604
28.14 GPIO Functionality 1606
28.14.1 GPIO Functionality 1606
28.14.2 Under Reset 1606
28.14.3 Out of Reset 1607
28.14.4 Open-Drain Feature Enabled on a Pin 1607
28.14.5 Summary 1607
29 Serial Communication Interface (SCI) Module 1608
29.1 Introduction 1609
29.1.1 SCI Features 1609
29.1.2 Block Diagram 1609
29.2 SCI Communication Formats 1611
29.2.1 SCI Frame Formats 1611
29.2.2 SCI Timing Mode 1611
29.2.3 SCI Baud Rate 1613

29.2.4 SCI Multiprocessor Communication Modes 1613
29.3 SCI Interrupts 1616
29.3.1 Transmit Interrupt 1617
29.3.2 Receive Interrupt 1617
29.3.3 WakeUp Interrupt 1617
29.3.4 Error Interrupts 1618
29.4 SCI DMA Interface 1619
29.4.1 Receive DMA Requests 1619
29.4.2 Transmit DMA Requests 1619
29.5 SCI Configurations 1620
29.5.1 Receiving Data 1620
29.5.2 Transmitting Data 1621
29.6 SCI Low-Power Mode 1621
29.6.1 Sleep Mode for Multiprocessor Communication 1622
29.7 SCI Control Registers 1623
29.7.1 SCI Global Control Register 0 (SCIGCR0) 1624
29.7.2 SCI Global Control Register 1 (SCIGCR1) 1625
29.7.3 SCI Set Interrupt Register (SCISETINT) 1628
29.7.4 SCI Clear Interrupt Register (SCICLEARINT) 1630
29.7.5 SCI Set Interrupt Level Register (SCISETINTLVL) 1632
29.7.6 SCI Clear Interrupt Level Register (SCICLEARINTLVL) 1633
29.7.7 SCI Flags Register (SCIFLR) 1635
29.7.8 SCI Interrupt Vector Offset 0 (SCIINTVECT0) 1639
29.7.9 SCI Interrupt Vector Offset 1 (SCIINTVECT1) 1639
29.7.10 SCI Format Control Register (SCIFORMAT) 1640
29.7.11 Baud Rate Selection Register (BRS) 1641
29.7.12 SCI Data Buffers (SCIED, SCIRD, SCITD) 1642
29.7.13 SCI Pin I/O Control Register 0 (SCIPIO0) 1643
29.7.14 SCI Pin I/O Control Register 1 (SCIPIO1) 1644
29.7.15 SCI Pin I/O Control Register 2 (SCIPIO2) 1645
29.7.16 SCI Pin I/O Control Register 3 (SCIPIO3) 1646
29.7.17 SCI Pin I/O Control Register 4 (SCIPIO4) 1647
29.7.18 SCI Pin I/O Control Register 5 (SCIPIO5) 1648
29.7.19 SCI Pin I/O Control Register 6 (SCIPIO6) 1649
29.7.20 SCI Pin I/O Control Register 7 (SCIPIO7) 1650
29.7.21 SCI Pin I/O Control Register 8 (SCIPIO8) 1650
29.7.22 Input/Output Error Enable (IODFTCTRL) Register 1651
29.8 GPIO Functionality 1653
29.8.1 GPIO Functionality 1653
29.8.2 Under Reset 1653
29.8.3 Out of Reset 1654
29.8.4 Open-Drain Feature Enabled on a Pin 1654
29.8.5 Summary 1654

30 Inter-Integrated Circuit (I2C) Module 1655
30.1 Overview 1656
30.1.1 Introduction to the I2C Module 1656
30.1.2 Functional Overview 1657
30.1.3 Clock Generation 1659
30.2 I2C Module Operation 1660
30.2.1 Input and Output Voltage Levels 1660
30.2.2 I2C Module Reset Conditions 1660
30.2.3 I2C Module Data Validity 1660
30.2.4 I2C Module Start and Stop Conditions 1661

30.2.5 Serial Data Formats 1661
30.2.6 NACK Bit Generation 1663
30.3 I2C Operation Modes 1664
30.3.1 Master Transmitter Mode 1664
30.3.2 Master Receiver Mode 1664
30.3.3 Slave Transmitter Mode 1664
30.3.4 Slave Receiver Mode 1664
30.3.5 Low Power Mode 1665
30.3.6 Free Run Mode 1665
30.3.7 Ignore NACK Mode 1665
30.4 I2C Module Integrity 1666
30.4.1 Arbitration 1666
30.4.2 I2C Clock Generation and Synchronization 1667
30.4.3 Prescaler 1667
30.4.4 Noise Filter 1667
30.5 Operational Information 1668
30.5.1 I2C Module Interrupts 1668
30.5.2 DMA Controller Events 1669
30.5.3 I2C Enable/Disable 1669
30.5.4 General Purpose I/O 1669
30.5.5 Pull Up/Pull Down Function 1670
30.5.6 Open Drain Function 1670
30.6 I2C Control Registers 1671
30.6.1 I2C Own Address Manager (I2COAR) 1672
30.6.2 I2C Interrupt Mask Register (I2CIMR) 1673
30.6.3 I2C Status Register (I2CSTR) 1674
30.6.4 I2C Clock Divider Low Register (I2CCKL) 1677
30.6.5 I2C Clock Control High Register (I2CCKH) 1677
30.6.6 I2C Data Count Register (I2CCNT) 1678
30.6.7 I2C Data Receive Register (I2CDRR) 1678
30.6.8 I2C Slave Address Register (I2CSAR) 1679
30.6.9 I2C Data Transmit Register (I2CDXR) 1679
30.6.10 I2C Mode Register (I2CMDR) 1680
30.6.11 I2C Interrupt Vector Register (I2CIVR) 1683
30.6.12 I2C Extended Mode Register (I2CEMDR) 1684
30.6.13 I2C Prescale Register (I2CPSC) 1684
30.6.14 I2C Peripheral ID Register 1 (I2CPID1) 1685
30.6.15 I2C Peripheral ID Register 2 (I2CPID2) 1685
30.6.16 I2C DMA Control Register (I2CDMACR) 1686
30.6.17 I2C Pin Function Register (I2CPFNC) 1686
30.6.18 I2C Pin Direction Register (I2CPDIR) 1687
30.6.19 I2C Data Input Register (I2CDIN) 1687
30.6.20 I2C Data Output Register (I2CDOUT) 1688
30.6.21 I2C Data Set Register (I2CDSET) 1688
30.6.22 I2C Data Clear Register (I2CDCLR) 1689
30.6.23 I2C Pin Open Drain Register (I2CPDR) 1689
30.6.24 I2C Pull Disable Register (I2CPDIS) 1690
30.6.25 I2C Pull Select Register (I2CPSEL) 1690
30.6.26 I2C Pins Slew Rate Select Register (I2CSRS) 1691
30.7 Sample Waveforms 1692
31 EMAC/MDIO Module 1693
31.1 Introduction 1694
31.1.1 Purpose of the Peripheral 1694

31.1.2 Features 1694
31.1.3 Functional Block Diagram 1695
31.1.4 Industry Standard(s) Compliance Statement 1696
31.2 Architecture 1696
31.2.1 Clock Control 1696
31.2.2 Memory Map 1696
31.2.3 Signal Descriptions 1697
31.2.4 MII / RMII Signal Multiplexing Control 1700
31.2.5 Ethernet Protocol Overview 1701
31.2.6 Programming Interface 1702
31.2.7 EMAC Control Module 1716
31.2.8 MDIO Module 1717
31.2.9 EMAC Module 1722
31.2.10 MAC Interface 1724
31.2.11 Packet Receive Operation 1728
31.2.12 Packet Transmit Operation 1733
31.2.13 Receive and Transmit Latency 1734
31.2.14 Transfer Node Priority 1734
31.2.15 Reset Considerations 1735
31.2.16 Initialization 1736
31.2.17 Interrupt Support 1738
31.2.18 Power Management 1742
31.2.19 Emulation Considerations 1742
31.3 EMAC Control Module Registers 1743
31.3.1 EMAC Control Module Revision ID Register (REVID) 1744
31.3.2 EMAC Control Module Software Reset Register (SOFTRESET) 1744
31.3.3 EMAC Control Module Interrupt Control Register (INTCONTROL) 1745
31.3.4 EMAC Control Module Receive Threshold Interrupt Enable Registers (C0RXTHRESHEN) 1746
31.3.5 EMAC Control Module Receive Interrupt Enable Registers (C0RXEN) 1747
31.3.6 EMAC Control Module Transmit Interrupt Enable Registers (C0TXEN) 1748
31.3.7 EMAC Control Module Miscellaneous Interrupt Enable Registers (C0MISCEN) 1749
31.3.8 EMAC Control Module Receive Threshold Interrupt Status Registers (C0RXTHRESHSTAT) 1750
31.3.9 EMAC Control Module Receive Interrupt Status Registers (C0RXSTAT) 1751
31.3.10 EMAC Control Module Transmit Interrupt Status Registers (C0TXSTAT) 1752
31.3.11 EMAC Control Module Miscellaneous Interrupt Status Registers (C0MISCSTAT) 1753
31.3.12 EMAC Control Module Receive Interrupts Per Millisecond Registers (C0RXIMAX) 1754
31.3.13 EMAC Control Module Transmit Interrupts Per Millisecond Registers (C0TXIMAX) 1755
31.4 MDIO Registers 1756
31.4.1 MDIO Revision ID Register (REVID) 1756
31.4.2 MDIO Control Register (CONTROL) 1757
31.4.3 PHY Acknowledge Status Register (ALIVE) 1758
31.4.4 PHY Link Status Register (LINK) 1758
31.4.5 MDIO Link Status Change Interrupt (Unmasked) Register (LINKINTRAW) 1759
31.4.6 MDIO Link Status Change Interrupt (Masked) Register (LINKINTMASKED) 1760
31.4.7 MDIO User Command Complete Interrupt (Unmasked) Register (USERINTRAW) 1761
31.4.8 MDIO User Command Complete Interrupt (Masked) Register (USERINTMASKED) 1762
31.4.9 MDIO User Command Complete Interrupt Mask Set Register (USERINTMASKSET) 1763
31.4.10 MDIO User Command Complete Interrupt Mask Clear Register (USERINTMASKCLEAR) 1764
31.4.11 MDIO User Access Register 0 (USERACCESS0) 1765
31.4.12 MDIO User PHY Select Register 0 (USERPHYSEL0) 1766
31.4.13 MDIO User Access Register 1 (USERACCESS1) 1767
31.4.14 MDIO User PHY Select Register 1 (USERPHYSEL1) 1768
31.5 EMAC Module Registers 1769

31.5.1 Transmit Revision ID Register (TXREVID) 1772
31.5.2 Transmit Control Register (TXCONTROL)........ 1772
31.5.3 Transmit Teardown Register (TXTEARDOWN)........ 1773
31.5.4 Receive Revision ID Register (RXREVID)........ 1773
31.5.5 Receive Control Register (RXCONTROL) 1774
31.5.6 Receive Teardown Register (RXTEARDOWN) 1774
31.5.7 Transmit Interrupt Status (Unmasked) Register (TXINTSTATRAW)........ 1775
31.5.8 Transmit Interrupt Status (Masked) Register (TXINTSTATMASKED) 1776
31.5.9 Transmit Interrupt Mask Set Register (TXINTMASKSET) 1777
31.5.10 Transmit Interrupt Mask Clear Register (TXINTMASKCLEAR) 1778
31.5.11 MAC Input Vector Register (MACINVECTOR) 1779
31.5.12 MAC End Of Interrupt Vector Register (MACEOIVECTOR) 1780
31.5.13 Receive Interrupt Status (Unmasked) Register (RXINTSTATRAW)........ 1781
31.5.14 Receive Interrupt Status (Masked) Register (RXINTSTATMASKED) 1782
31.5.15 Receive Interrupt Mask Set Register (RXINTMASKSET) 1783
31.5.16 Receive Interrupt Mask Clear Register (RXINTMASKCLEAR)........ 1784
31.5.17 MAC Interrupt Status (Unmasked) Register (MACINTSTATRAW) 1785
31.5.18 MAC Interrupt Status (Masked) Register (MACINTSTATMASKED)........ 1785
31.5.19 MAC Interrupt Mask Set Register (MACINTMASKSET)........ 1786
31.5.20 MAC Interrupt Mask Clear Register (MACINTMASKCLEAR) 1786
31.5.21 Receive Multicast/Broadcast/Promiscuous Channel Enable Register (RXMBPENABLE)........ 1787
31.5.22 Receive Unicast Enable Set Register (RXUNICASTSET) 1789
31.5.23 Receive Unicast Clear Register (RXUNICASTCLEAR) 1790
31.5.24 Receive Maximum Length Register (RXMAXLEN)........ 1790
31.5.25 Receive Buffer Offset Register (RXBUFFEROFFSET)........ 1791
31.5.26 Receive Filter Low Priority Frame Threshold Register (RXFILTERLOWTHRESH)........ 1791
31.5.27 Receive Channel Flow Control Threshold Registers (RX0FLOWTHRESH-RX7FLOWTHRESH) 1792
31.5.28 Receive Channel Free Buffer Count Registers (RX0FREEBUFFER-RX7FREEBUFFER)........ 1792
31.5.29 MAC Control Register (MACCONTROL) 1793
31.5.30 MAC Status Register (MACSTATUS)........ 1795
31.5.31 Emulation Control Register (EMCONTROL) 1797
31.5.32 FIFO Control Register (FIFOCONTROL)........ 1797
31.5.33 MAC Configuration Register (MACCONFIG) 1798
31.5.34 Soft Reset Register (SOFTRESET)........ 1798
31.5.35 MAC Source Address Low Bytes Register (MACSRCADDRLO) 1799
31.5.36 MAC Source Address High Bytes Register (MACSRCADDRHI)........ 1799
31.5.37 MAC Hash Address Register 1 (MACHASH1)........ 1800
31.5.38 MAC Hash Address Register 2 (MACHASH2)........ 1800
31.5.39 Back Off Test Register (BOFFTEST) 1801
31.5.40 Transmit Pacing Algorithm Test Register (TPACETEST) 1801
31.5.41 Receive Pause Timer Register (RXPAUSE) 1802
31.5.42 Transmit Pause Timer Register (TXPAUSE) 1802
31.5.43 MAC Address Low Bytes Register (MACADDRLO) 1803
31.5.44 MAC Address High Bytes Register (MACADDRHI)........ 1804
31.5.45 MAC Index Register (MACINDEX) 1804
31.5.46 Transmit Channel DMA Head Descriptor Pointer Registers (TX0HDP-TX7HDP) 1805
31.5.47 Receive Channel DMA Head Descriptor Pointer Registers (RX0HDP-RX7HDP) 1805
31.5.48 Transmit Channel Completion Pointer Registers (TX0CP-TX7CP) 1806
31.5.49 Receive Channel Completion Pointer Registers (RX0CP-RX7CP) 1806
31.5.50 Network Statistics Registers 1807

32 Universal Serial Bus (USB)........ 1816

32.1 Overview........ 1817
32.2 USB Host Controller 1817

32.2.1 USB Open Host Controller Interface Functionality 1817
32.2.2 USB Host Controller Differences From OHCI Specification for USB 1818
32.2.3 Implementation of OHCI Specification for USB 1819
32.2.4 USB Host Controller Registers 1820
32.2.5 USB Host Controller Reserved Registers and Reserved Bit Fields 1843
32.2.6 USB Host Controller Registers, USB Reset, and USB Clocking 1843
32.2.7 OHCI Interrupts 1843
32.2.8 USB Host Controller Access to System Memory 1844
32.2.9 Physical Addressing 1844
32.2.10 USB Host Controller Bus Addressing and OHCI Data Structure Pointers 1844
32.2.11 NULL Pointers 1844
32.2.12 USB Host Controller Power Management 1844
32.3 USB Device Controller 1844
32.3.1 USB Device Controller Registers 1845
32.3.2 USB Device Transactions 1877
32.3.3 Non-Isochronous, Non-Setup OUT (USB HOST→CPU) Transactions 1877
32.3.4 Non-Isochronous IN (CPU→USB HOST) Transactions 1881
32.3.5 Isochronous OUT (USB HOST→CPU) Transactions 1884
32.3.6 Isochronous IN (CPU→USB HOST) Transactions 1886
32.3.7 Control Transfers on Endpoint 0 1887
32.3.8 USB Device Initialization 1895
32.3.9 Preparing for Transfers 1898
32.3.10 USB Device Interrupt Service Routine (ISR) Flowcharts 1901
32.3.11 Important Note on USB Device Interrupts 1901
32.3.12 Parsing General USB Device Interrupt 1901
32.3.13 Setup Interrupt Handler 1902
32.3.14 Endpoint 0 RX Interrupt Handler 1905
32.3.15 Endpoint 0 TX Interrupt Handler 1906
32.3.16 Device States Changed Handler 1909
32.3.17 Device States Attached/Unattached Handler 1912
32.3.18 Device State Configuration Changed Handler 1913
32.3.19 Device State Address Changed Handler 1914
32.3.20 USB Device Reset Interrupt Handler 1914
32.3.21 Suspend/Resume Interrupt Handler 1915
32.3.22 Parsing Non-ISO Endpoint-Specific Interrupt 1916
32.3.23 Non-ISO, Non-Control OUT Endpoint Receive Interrupt Handler 1917
32.3.24 Non-ISO, Non-Control IN Endpoint Transmit Interrupt Handler 1920
32.3.25 SOF Interrupt Handler 1922
32.3.26 Summary of USB Device Controller Interrupts 1925
32.3.27 DMA Operation 1926
32.3.28 Power Management 1935
32.4 USB Connectivity 1938
32.4.1 Transceiver Interface 1938
32.4.2 Selecting and Configuring USB Connectivity 1938
32.4.3 Transceiver Signaling 1939
32.4.4 Host Controller Connectivity With USB Transceivers 1940
32.4.5 USB Function Controller Connectivity With USB Transceivers 1941
32.4.6 USB Hardware Considerations 1942

33 eFuse Controller 1943
33.1 Overview 1944
33.2 Introduction 1944
33.3 eFuse Controller Testing 1944
33.3.1 eFuse Controller Connections to ESM 1944

33.3.2 Checking for eFuse Errors After Power Up 1944
33.4 eFuse Controller Registers 1947
33.4.1 EFC Boundary Control Register (EFCBOUND) 1947
33.4.2 EFC Pins Register (EFCPINS) 1949
33.4.3 EFC Error Status Register (EFCERRSTAT) 1950
33.4.4 EFC Self Test Cycles Register (EFCSTCY) 1950
33.4.5 EFC Self Test Signature Register (EFCSTSIG) 1951
Revision History 1952

List of Figures

1-1. Block Diagram 98
1-2. Example: SPIDELAY – 0xFFF7F448 99
2-1. Architectural Block Diagram 101
2-2. Memory-Map 105
2-3. Hardware Memory Initialization Protocol 115
2-4. LPO and Clock Detection, Untrimmed HF LPO 127
2-5. SYS Pin Control Register 1 (SYSPC1) [offset = 00] 132
2-6. SYS Pin Control Register 2 (SYSPC2) [offset = 04h] 132
2-7. SYS Pin Control Register 3 (SYSPC3) [offset = 08h] 133
2-8. SYS Pin Control Register 4 (SYSPC4) [offset = 0Ch] 133
2-9. SYS Pin Control Register 5 (SYSPC5) [offset = 10h] 134
2-10. SYS Pin Control Register 6 (SYSPC6) [offset = 14h] 134
2-11. SYS Pin Control Register 7 (SYSPC7) [offset = 18h] 135
2-12. SYS Pin Control Register 8 (SYSPC8) [offset = 1Ch] 135
2-13. SYS Pin Control Register 9 (SYSPC9) [offset = 20h] 136
2-14. Clock Source Disable Register (CSDIS) [offset = 30h] 137
2-15. Clock Source Disable Set Register (CSDISSET) [offset = 34h] 138
2-16. Clock Source Disable Clear Register (CSDISCLR) [offset = 38h] 139
2-17. Clock Domain Disable Register (CDDIS) [offset = 3Ch] 140
2-18. Clock Domain Disable Set Register (CDDISSET) [offset = 40h] 142
2-19. Clock Domain Disable Clear Register (CDDISCLR) [offset = 44h] 144
2-20. GCLK, HCLK, VCLK, and VCLK2 Source Register (GHVSRC) [offset = 48h] 146
2-21. Peripheral Asynchronous Clock Source Register (VCLKASRC) [offset = 4Ch] 147
2-22. RTI Clock Source Register (RCLKSRC) [offset = 50h] 148
2-23. Clock Source Valid Status Register (CSVSTAT) [offset = 54h] 149
2-24. Memory Self-Test Global Control Register (MSTGCR) [offset = 58h] 150
2-25. Memory Hardware Initialization Global Control Register (MINITGCR) [offset = 5Ch] 151
2-26. MBIST Controller/Memory Initialization Enable Register (MSINENA) [offset = 60h] 152
2-27. Memory Self-Test Fail Status Register (MSTFAIL) [offset = 64h] 153
2-28. MSTC Global Status Register (MSTCGSTAT) [offset = 68h] 154
2-29. Memory Hardware Initialization Status Register (MINISTAT) [offset = 6Ch] 155
2-30. PLL Control Register 1 (PLLCTL1) [offset = 70h] 155
2-31. PLL Control Register 2 (PLLCTL2) [offset = 74h] 157
2-32. SYS Pin Control Register 10 (SYSPC10) [offset = 78h] 158
2-33. Die Identification Register, Lower Word (DIEIDL) [offset = 7Ch] 159
2-34. Die Identification Register, Upper Word (DIEIDH) [offset = 80h] 159
2-35. LPO/Clock Monitor Control Register (LPOMONCTL) [offset = 88h] 160
2-36. Clock Test Register (CLKTEST) [offset = 8Ch] 163
2-37. DFT Control Register (DFTCTRLREG) [offset = 90h] 165
2-38. DFT Control Register 2 (DFTCTRLREG2) [offset = 94h] 166
2-39. General Purpose Register (GPREG1) [offset = A0h] 167
2-40. Imprecise Fault Status Register (IMPFASTS) [offset = A8h] 169
2-41. Imprecise Fault Write Address Register (IMPFTADD) [offset = ACh] 170
2-42. System Software Interrupt Request 1 Register (SSIR1) [offset = B0h] 171
2-43. System Software Interrupt Request 2 Register (SSIR2) [offset = B4h] 171
2-44. System Software Interrupt Request 3 Register (SSIR3) [offset = B8h] 172
2-45. System Software Interrupt Request 4 Register (SSIR4) [offset = BCh] 172

2-46. RAM Control Register (RAMGCR) [offset = C0h] 173
2-47. Bus Matrix Module Control Register 1 (BMMCR) [offset = C4h] 174
2-48. CPU Reset Control Register (CPURSTCR) [offset = CCh] 174
2-49. Clock Control Register (CLKCNTRL) [offset = D0h] 176
2-50. ECP Control Register (ECPCNTL) [offset = D4h] 177
2-51. DEV Parity Control Register 1 (DEVCR1) [offset = DCh] 178
2-52. System Exception Control Register (SYSECR) [offset = E0h] 178
2-53. System Exception Status Register (SYSESR) [offset = E4h] 179
2-54. System Test Abort Status Register (SYSTASR) [offset = E8h] 180
2-55. Global Status Register (GLBSTAT) [offset = ECh] 181
2-56. Device Identification Register (DEVID) [offset = F0h] 182
2-57. Software Interrupt Vector Register (SSIVEC) [offset = F4h] 183
2-58. System Software Interrupt Flag Register (SSIF) [offset = F8h] 184
2-59. PLL Control Register 3 (PLLCTL3) [offset = 00] 186
2-60. CPU Logic BIST Clock Prescaler (STCLKDIV) [offset = 08h] 187
2-61. Clock 2 Control Register (CLK2CNTRL) [offset = 3Ch] 187
2-62. Peripheral Asynchronous Clock Configuration 1 Register (VCLKACON1) [offset = 40h] 188
2-63. Clock Slip Register (CLKSLIP) [offset = 70h] 190
2-64. EFUSE Controller Control Register (EFC_CTLREG) [offset = ECh] 191
2-65. Die Identification Register, Lower Word (DIEIDL_REG0) [offset = F0h] 191
2-66. Die Identification Register, Upper Word (DIEIDH_REG1) [offset = F4h] 192
2-67. Die Identification Register, Lower Word (DIEIDL_REG2) [offset = F8h] 192
2-68. Die Identification Register, Upper Word (DIEIDH_REG3) [offset = FCh] 192
2-69. Peripheral Memory Protection Set Register 0 (PMPROTSET0) [offset = 00] 195
2-70. Peripheral Memory Protection Set Register 1 (PMPROTSET1) [offset = 04h] 195
2-71. Peripheral Memory Protection Clear Register 0 (PMPROTCLR0) [offset = 10h] 196
2-72. Peripheral Memory Protection Clear Register 1 (PMPROTCLR1) [offset = 14h] 196
2-73. Peripheral Protection Set Register 0 (PPROTSET0) [offset = 20h] 197
2-74. Peripheral Protection Set Register 1 (PPROTSET1) [offset = 24h] 198
2-75. Peripheral Protection Set Register 2 (PPROTSET2) [offset = 28h] 198
2-76. Peripheral Protection Set Register 3 (PPROTSET3) [offset = 2Ch] 199
2-77. Peripheral Protection Clear Register 0 (PPROTCLR0) [offset = 40h] 199
2-78. Peripheral Protection Clear Register 1 (PPROTCLR1) [offset = 44h] 200
2-79. Peripheral Protection Clear Register 2 (PPROTCLR2) [offset = 48h] 200
2-80. Peripheral Protection Clear Register 3 (PPROTCLR3) [offset = 4Ch] 201
2-81. Peripheral Memory Power-Down Set Register 0 (PCSPWRDWNSET0) [offset = 60h] 201
2-82. Peripheral Memory Power-Down Set Register 1 (PCSPWRDWNSET1) [offset = 64h] 202
2-83. Peripheral Memory Power-Down Clear Register 0 (PCSPWRDWNCLR0) [offset = 70h] 202
2-84. Peripheral Memory Power-Down Clear Register 1 (PCSPWRDWNCLR1) [offset = 74h] 203
2-85. Peripheral Power-Down Set Register 0 (PSPWRDWNSET0) [offset = 80h] 204
2-86. Peripheral Power-Down Set Register 1 (PSPWRDWNSET1) [offset = 84h] 205
2-87. Peripheral Power-Down Set Register 2 (PSPWRDWNSET2) [offset = 88h] 205
2-88. Peripheral Power-Down Set Register 3 (PSPWRDWNSET3) [offset = 8Ch] 206
2-89. Peripheral Power-Down Clear Register 0 (PSPWRDWNCLR0) [offset = A0h] 206
2-90. Peripheral Power-Down Clear Register 1 (PSPWRDWNCLR1) [offset = A4h] 207
2-91. Peripheral Power-Down Clear Register 2 (PSPWRDWNCLR2) [offset = A8h] 207
2-92. Peripheral Power-Down Clear Register 3 (PSPWRDWNCLR) [offset = ACh] 208
3-1. PMM Block Diagram 211
3-2. Core Power Domains 212

3-3. LOGICPDPWRCTRL0 Register 217
3-4. MEMPDPWRCTRL0 Register........ 219
3-5. PDCLKDIS Register 220
3-6. PDCLKDISSET Register........ 221
3-7. PDCLKDISCLR Register........ 222
3-8. LOGICPDPWRSTAT0 Register........ 223
3-9. LOGICPDPWRSTAT1 Register........ 224
3-10. LOGICPDPWRSTAT2 Register........ 225
3-11. LOGICPDPWRSTAT3 Register........ 226
3-12. MEMPDPWRSTAT0 Register........ 227
3-13. MEMPDPWRSTAT1 Register........ 229
3-14. GLOBALCTRL1 Register 231
3-15. GLOBALSTAT Register........ 232
3-16. PRCKEYREG Register 233
3-17. LPDDCSTAT1 Register........ 234
3-18. LPDDCSTAT2 Register........ 235
3-19. MPDDCSTAT1 Register 236
3-20. MPDDCSTAT2 Register 237
3-21. ISODIAGSTAT Register 238
4-1. PINMMR10 Control Register, Address = FFFF EB38h........ 240
4-2. Output Multiplexing Example........ 241
4-3. Input Multiplexing Example........ 242
4-4. ADC Trigger Event Signal Generation from ePWMx........ 245
4-5. Sychronizing ePWMx Modules to N2HET1 Time-Base........ 248
4-6. REVISION_REG Register 253
4-7. ENDIAN_REG Register........ 254
4-8. KICK_REG0 Register 255
4-9. KICK_REG1 Register 256
4-10. ERR_RAW_STATUS_REG Register 257
4-11. ERR_ENABLED_STATUS_REG Register 258
4-12. ERR_ENABLE_REG Register 259
4-13. ERR_ENABLE_CLR_REG Register 260
4-14. FAULT_ADDRESS_REG Register........ 261
4-15. FAULT_STATUS_REG Register 262
4-16. FAULT_CLEAR_REG Register........ 263
4-17. PINMMR_0 to PINMMR_47 Register 264
5-1. ECC Organization for Program Flash (144-Bits Wide) 274
5-2. TI OTP Bank 0 Sector Information 275
5-3. TI OTP Bank 0 Package and Memory Size Information (F008 015Ch) 276
5-4. TI OTP Bank 0 LPO Trim and Max HCLK Information (F008 01B4h)........ 276
5-5. TI OTP Bank 0 Symbolization Information (F008 01E0h-F008 01FFh) 277
5-6. TI OTP Bank 0 Deliberate ECC Error Information (F008 03F0h-F008 03FFh)........ 277
5-7. ECC Malfunction Test Logic........ 280
5-8. Flash Option Control Register (FRDCNTL) [offset = 00h] 286
5-9. Flash Error Detection and Correction Control Register 1 (FEDACCTRL1) [offset = 08h]........ 287
5-10. Flash Error Correction and Correction Control Register 2 (FEDACCTRL2) [offset = 0Ch] 289
5-11. Flash Correctable Error Count Register (FCOR_ERR_CNT) [offset = 10h] 289
5-12. Flash Correctable Error Address Register (FCOR_ERR_ADD) [offset = 14h] 290
5-13. Flash Correctable Error Position Register (FCOR_ERR_POS) [offset = 18h]........ 291

5-14. Flash Error Detection and Correction Status Register (FEDACSTATUS) [offset = 1Ch] 292
5-15. Flash Uncorrectable Error Address Register (FUNC_ERR_ADD) [offset = 20h] 295
5-16. Flash Error Detection and Correction Sector Disable Register (FEDACSDIS) [offset = 24h] 296
5-17. Primary Address Tag Register (FPRIM_ADD_TAG) [offset = 28h] 297
5-18. Duplicate Address Tag Register (FDUP_ADD_TAG) [offset = 2Ch] 297
5-19. Flash Bank Protection Register (FBPROT) [offset = 30h] 298
5-20. Flash Bank Sector Enable Register (FBSE) [offset = 34h] 298
5-21. Flash Bank Busy Register (FBBUSY) [offset = 38h] 299
5-22. Flash Bank Access Control Register (FBAC) [offset = 3Ch] 300
5-23. Flash Bank Fallback Power Register (FBFALLBACK) [offset = 40h] 301
5-24. Flash Bank/Pump Ready Register (FBPRDY) [offset = 44h] 302
5-25. Flash Pump Access Control Register 1 (FPAC1) [offset = 48h] 303
5-26. Flash Pump Access Control Register 2 (FPAC2) [offset = 4Ch] 304
5-27. Flash Module Access Control Register (FMAC) [offset = 50h] 304
5-28. Flash Module Status Register (FMSTAT) [offset = 54h] 305
5-29. EEPROM Emulation Data MSW Register (FEMU_DMSW) [offset = 58h] 307
5-30. EEPROM Emulation Data LSW Register (FEMU_DLSW) [offset = 5Ch] 307
5-31. EEPROM Emulation ECC Register (FEMU_ECC) [offset = 60h] 308
5-32. EEPROM Emulation Address Register (FEMU_ADDR) [offset = 68h] 309
5-33. Diagnostic Control Register (FDIAGCTRL) [offset = 6Ch] 310
5-34. Uncorrected Raw Data High Register (FRAW_DATAH) [offset = 70h] 312
5-35. Uncorrected Raw Data Low Register (FRAW_DATAL) [offset = 74h] 312
5-36. Uncorrected Raw ECC Register (FRAW_ECC) [offset = 78h] 313
5-37. Parity Override Register (FPAR_OVR) [offset = 7Ch] 314
5-38. Flash Error Detection and Correction Sector Disable Register (FEDACSDIS2) [offset = C0h] 315
5-39. FSM Register Write Enable (FSM_WR_ENA) [offset = 288h] 316
5-40. FSM Sector Register (FSM_SECTOR) [offset = 2A4h] 316
5-41. EEPROM Emulation Configuration Register (EEPROM_CONFIG) [offset = 2B8h] 317
5-42. EEPROM Emulation Error Detection and Correction Control Register 1 (EE_CTRL1) [offset = 308h] 318
5-43. EEPROM Emulation Error Correction and Correction Control Register 2 (EE_CTRL2) [offset = 30Ch] 320
5-44. EEPROM Emulation Error Correctable Error Count Register (EE_COR_ERR_CNT) [offset = 310h] 320
5-45. EEPROM Emulation Correctable Error Address Register (EE_COR_ERR_ADD) [offset = 314h] 321
5-46. EEPROM Emulation Correctable Error Position Register (EE_COR_ERR_POS) [offset = 318h] 322
5-47. EEPROM Emulation Error Status Register (EE_STATUS) [offset = 31Ch] 323
5-48. EEPROM Emulation Uncorrectable Error Address Register (EE_UNC_ERR_ADD) [offset = 320h] 324
5-49. Flash Bank Configuration Register (FCFG_BANK) [offset = 400h] 325
6-1. TCRAM Module Connections 327
6-2. RAM Memory Map 328
6-3. RAMCTRL Register 333
6-4. RAMTHRESHOLD Register 335
6-5. RAMOCCUR Register 336
6-6. RAMINTCTRL Register 337
6-7. RAMERRSTATUS Register 338
6-8. RAMSERRADDR Register 340
6-9. RAMUERRADDR Register 341
6-10. RAMTEST Register 342
6-11. RAMADDRDECVECT Register 344
6-12. RAMPERRADDR Register 345
6-13. INIT_DOMAIN Register 346

7-1. PBIST Block Diagram 348
7-2. PBIST Memory Self-Test Flow Diagram 350
7-3. RAM Configuration Register (RAMT) [offset = 0160h] 355
7-4. Datalogger Register (DLR) [offset = 0164h] 356
7-5. PBIST Activate/ROM Clock Enable Register (PACT) [offset = 0180h] 357
7-6. PBIST ID Register [offset = 184h] 358
7-7. Override Register (OVER) [offset = 0188h] 359
7-8. Fail Status Fail Register 0 (FSRF0) [offset = 0190h] 360
7-9. Fail Status Count 0 Register (FSRC0) [offset = 0198h] 361
7-10. Fail Status Count Register 1 (FSRC1) [offset = 019Ch] 361
7-11. Fail Status Address 0 Register (FSRA0) [offset = 01A0h] 362
7-12. Fail Status Address 1 Register (FSRA1) [offset = 01A4h] 362
7-13. Fail Status Data Register 0 (FSRDL0) [offset = 01A8h] 363
7-14. Fail Status Data Register 1 (FSRDL1) [offset = 01B0h] 363
7-15. ROM Mask Register (ROM) [offset = 01C0h] 364
7-16. ROM Algorithm Mask Register (ALGO) [offset = 01C4h] 364
7-17. RAM Info Mask Lower Register (RINFOL) [offset = 01C8h] 365
7-18. RAM Info Mask Upper Register (RINFOU) [offset = 01CCh] 366
8-1. STC Block Diagram 371
8-2. Application Self-Test Flow Chart 373
8-3. STC Global Control Register 0 (STCGCR0) [offset = 00] 376
8-4. STC Global Control Register 1 (STCGCR1) [offset = 04h] 376
8-5. Self-Test Run Timeout Counter Preload Register (STCTPR) [offset = 08h] 377
8-6. STC Current ROM Address Register (STC_CADDR) [offset = 0Ch] 377
8-7. STC Current Interval Count Register (STCCICR) [offset = 10h] 378
8-8. Self-Test Global Status Register (STCGSTAT) [offset = 14h] 379
8-9. Self-Test Fail Status Register (STCFSTAT) [offset = 18h] 380
8-10. CPU1 Current MISR Register (CPU1_CURMISR3) [offset = 1Ch] 381
8-11. CPU1 Current MISR Register (CPU1_CURMISR2) [offset = 20h] 381
8-12. CPU1 Current MISR Register (CPU1_CURMISR1) [offset = 24h] 381
8-13. CPU1 Current MISR Register (CPU1_CURMISR0) [offset = 28h] 381
8-14. CPU2 Current MISR Register (CPU2_CURMISR3) [offset = 2Ch] 382
8-15. CPU2 Current MISR Register (CPU2_CURMISR2) [offset = 30h] 382
8-16. CPU2 Current MISR Register (CPU2_CURMISR1) [offset = 34h] 382
8-17. CPU2 Current MISR Register (CPU2_CURMISR0) [offset = 38h] 382
8-18. Signature Compare Self-Check Register (STCSCSCR) [offset = 3Ch] 383
9-1. Block Diagram 386
9-2. CCM-R4F Status Register (CCMSR) (Address = FFFF F600h) 391
9-3. CCM-R4F Key Register (CCMKEYR) (Address = FFFF F604h) 392
10-1. Clock Path From Oscillator Through PLL To Device 395
10-2. Clock Generation Path 396
10-3. Oscillator Implementation 397
10-4. Operation of the FM-PLL Module 401
10-5. PLL Slip Detection and Reset/Bypass Block Diagram 407
10-6. SSW PLL BIST Control Register 1 (SSWPLL1) [offset = FF24h] 412
10-7. SSW PLL BIST Control Register 2 (SSWPLL2) [offset = FF28h] 413
10-8. SSW PLL BIST Control Register 3 (SSWPLL3) [offset = FF2Ch] 414
10-9. Basic PLL Circuit 415
10-10. PFD Timing 415

10-11. PLL Modulation Block Diagram 416
10-12. Frequency vs. Time 417
11-1. DCC Operation 420
11-2. Counter Relationship 422
11-3. Clock1 Slower Than Clock0 - Results in an Error and Stops Counting 422
11-4. Clock1 Faster Than Clock0 - Results in an Error and Stops Counting 423
11-5. Clock1 Not Present - Results in an Error and Stops Counting 423
11-6. Clock0 Not Present - Results in an Error and Stops Counting 424
11-7. DCC Global Control Register (DCCGCTRL) [offset = 00] 427
11-8. DCC Revision Id Register (DCCREV) [offset = 4h] 428
11-9. DCC Counter0 Seed Register (DCCCNT0SEED) [offset = 8h] 428
11-10. DCC Valid0 Seed Register (DCCVALID0SEED) [offset = Ch] 429
11-11. DCC Counter1 Seed Register (DCCCNT1SEED) [offset = 10h] 429
11-12. DCC Status Register (DCCSTAT) [offset = 14h] 430
11-13. DCC Counter0 Value Register (DCCCNT0) [offset = 18h] 431
11-14. DCC Valid0 Value Register (DCCVALID0) [offset = 1Ch] 432
11-15. DCC Counter1 Value Register (DCCCNT1) [offset = 20h] 432
11-16. DCC Counter1 Clock Source Selection Register (DCCCNT1CLKSRC) [offset = 24h] 433
11-17. DCC Counter0 Clock Source Selection Register (DCCCNT0CLKSRC) [offset = 28h] 434
12-1. Block Diagram 436
12-2. Interrupt Response Handling 437
12-3. $\overline{\text{ERROR}}$ Pin Response Handling 437
12-4. $\overline{\text{ERROR}}$ Pin Timing - Example 1 439
12-5. $\overline{\text{ERROR}}$ Pin Timing - Example 2 439
12-6. $\overline{\text{ERROR}}$ Pin Timing - Example 3 439
12-7. $\overline{\text{ERROR}}$ Pin Timing - Example 4 440
12-8. $\overline{\text{ERROR}}$ Pin Timing - Example 5 440
12-9. $\overline{\text{ERROR}}$ Pin Timing - Example 7 441
12-10. ESM Initialization 442
12-11. ESM Enable $\overline{\text{ERROR}}$ Pin Action/Response Register 1 (ESMEEPAPR1) [address = FFFF F500h] 444
12-12. ESM Disable $\overline{\text{ERROR}}$ Pin Action/Response Register 1 (ESMDEPAPR1) [address = FFFF F504h] 444
12-13. ESM Interrupt Enable Set Register 1 (ESMIESR1) [address = FFFF F508h] 445
12-14. ESM Interrupt Enable Clear Register 1 (ESMIECR1) [address = FFFF F50Ch] 445
12-15. ESM Interrupt Level Set Register 1 (ESMILSR1) [address = FFFF F510h] 446
12-16. ESM Interrupt Level Clear Register 1 (ESMILCR1) [address = FFFF F514h] 446
12-17. ESM Status Register 1 (ESMSR1) [address = FFFF F518h] 447
12-18. ESM Status Register 2 (ESMSR2) [address = FFFF F51Ch] 447
12-19. ESM Status Register 3 (ESMSR3) [address = FFFF F520h] 448
12-20. ESM $\overline{\text{ERROR}}$ Pin Status Register (ESMEPSR) [address = FFFF F524h] 448
12-21. ESM Interrupt Offset High Register (ESMIOFFHR) [address = FFFF F528h] 449
12-22. ESM Interrupt Offset Low Register (ESMIOFFLR) [address = FFFF F52Ch] 450
12-23. ESM Low-Time Counter Register (ESMLTCR) [address = FFFF F530h] 451
12-24. ESM Low-Time Counter Preload Register (ESMLTCPR) [address = FFFF F534h] 451
12-25. ESM Error Key Register (ESMEKR) [address = FFFF F538h] 452
12-26. ESM Status Shadow Register 2 (ESMSSR2) [address = FFFF F53Ch] 452
12-27. ESM Influence $\overline{\text{ERROR}}$ Pin Set Register 4 (ESMIEPSR4) [address = FFFF F540h] 453
12-28. ESM Influence $\overline{\text{ERROR}}$ Pin Clear Register 4 (ESMIEPCR4) [address = FFFF F544h] 453
12-29. ESM Interrupt Enable Set Register 4 (ESMIESR4) [address = FFFF F548h] 454
12-30. ESM Interrupt Enable Clear Register 4 (ESMIECR4) [address = FFFF F54Ch] 454

12-31. ESM Interrupt Level Set Register 4 (ESMILSR4) [address = FFFF F550h] 455
12-32. ESM Interrupt Level Clear Register 4 (ESMILCR4) [address = FFFF F554h] 455
12-33. ESM Status Register 4 (ESMSR4) [address = FFFF F558h] 456
13-1. RTI Block Diagram 459
13-2. Counter Block Diagram 460
13-3. Compare Unit Block Diagram (shows only 1 of 4 blocks for simplification) 462
13-4. Timebase Control 463
13-5. Clock Detection Scheme 463
13-6. Switch to NTUx 464
13-7. Missing NTUx Signal Example 465
13-8. Digital Watchdog 465
13-9. DWD Operation 466
13-10. Digital Windowed Watchdog Timing Example 467
13-11. Digital Windowed Watchdog Operation Example (25% Window) 467
13-12. RTI Global Control Register (RTIGCTRL) [offset = 00] 470
13-13. RTI Timebase Control Register (RTITBCTRL) [offset = 04h] 471
13-14. RTI Capture Control Register (RTICAPCTRL) [offset = 08h] 472
13-15. RTI Compare Control Register (RTICOMPCTRL) [offset = 0Ch] 473
13-16. RTI Free Running Counter 0 Register (RTIFRC0) [offset = 10h] 474
13-17. RTI Up Counter 0 Register (RTIUC0) [offset = 14h] 474
13-18. RTI Compare Up Counter 0 Register (RTICPUC0) [offset = 18h] 475
13-19. RTI Capture Free Running Counter 0 Register (RTICAFRC0) [offset = 20h] 475
13-20. RTI Capture Up Counter 0 Register (RTICAUC0) [offset = 24h] 476
13-21. RTI Free Running Counter 1 Register (RTIFRC1) [offset = 30h] 476
13-22. RTI Up Counter 1 Register (RTIUC1) [offset = 34h] 477
13-23. RTI Compare Up Counter 1 Register (RTICPUC1) [offset = 38h] 478
13-24. RTI Capture Free Running Counter 1 Register (RTICAFRC1) [offset = 40h] 479
13-25. RTI Capture Up Counter 1 Register (RTICAUC1) [offset = 44h] 479
13-26. RTI Compare 0 Register (RTICOMP0) [offset = 50h] 480
13-27. RTI Update Compare 0 Register (RTIUDCP0) [offset = 54h] 480
13-28. RTI Compare 1 Register (RTICOMP1) [offset = 58h] 481
13-29. RTI Update Compare 1 Register (RTIUDCP1) [offset = 5Ch] 481
13-30. RTI Compare 2 Register (RTICOMP2) [offset = 60h] 482
13-31. RTI Update Compare 2 Register (RTIUDCP2) [offset = 64h] 482
13-32. RTI Compare 3 Register (RTICOMP3) [offset = 68h] 483
13-33. RTI Update Compare 3 Register (RTIUDCP3) [offset = 6Ch] 483
13-34. RTI Timebase Low Compare Register (RTITBLCOMP) [offset = 70h] 484
13-35. RTI Timebase High Compare Register (RTITBHCOMP) [offset = 74h] 484
13-36. RTI Set Interrupt Control Register (RTISETINTENA) [offset = 80h] 485
13-37. RTI Clear Interrupt Control Register (RTICLEARINTENA) [offset = 84h] 487
13-38. RTI Interrupt Flag Register (RTIINTFLAG) [offset = 88h] 489
13-39. Digital Watchdog Control Register (RTIDWDCTRL) [offset = 90h] 490
13-40. Digital Watchdog Preload Register (RTIDWDPRLD) [offset = 94h] 491
13-41. Watchdog Status Register (RTIWDSTATUS) [offset = 98h] 492
13-42. RTI Watchdog Key Register (RTIDWDKEY) [offset = 9Ch] 493
13-43. RTI Watchdog Down Counter Register (RTIDWDCNTR) [offset = A0h] 494
13-44. Digital Windowed Watchdog Reaction Control (RTIWWDRXNCTRL) [offset = A4h] 494
13-45. Digital Windowed Watchdog Window Size Control (RTIWWDSIZECTRL) [offset = A8h] 495
13-46. RTI Compare Interrupt Clear Enable Register (RTIINTCLRENABLE) [offset = ACh] 496

13-47. RTI Compare 0 Clear Register (RTICMP0CLR) [offset = B0h] 497
13-48. RTI Compare 1 Clear Register (RTICMP1CLR) [offset = B4h] 497
13-49. RTI Compare 2 Clear Register (RTICMP2CLR) [offset = B8h] 498
13-50. RTI Compare 3 Clear Register (RTICMP3CLR) [offset = BCh] 498
14-1. CRC Controller Block Diagram For One Channel 501
14-2. LFSR 503
14-3. AUTO Mode Using Hardware Timer Trigger 506
14-4. AUTO Mode With Software CPU Trigger 506
14-5. Semi-CPU Mode With Hardware Timer Trigger 507
14-6. Timeout Example 1 509
14-7. Timeout Example 2 510
14-8. Timeout Example 3 510
14-9. CRC Global Control Register 0 (CRC_CTRL0) [offset = 00h] 517
14-10. CRC Global Control Register 1 (CRC_CTRL1) [offset = 08h] 517
14-11. CRC Global Control Register 2 (CRC_CTRL2) [offset = 10h] 518
14-12. CRC Interrupt Enable Set Register (CRC_INTS) [offset = 18h] 519
14-13. CRC Interrupt Enable Reset Register (CRC_INTR) [offset = 20h] 521
14-14. CRC Interrupt Status Register (CRC_STATUS) [offset = 28h] 523
14-15. CRC Interrupt Offset (CRC_INT_OFFSET_REG) [offset = 30h] 525
14-16. CRC Busy Register (CRC_BUSY) [offset = 38h] 526
14-17. CRC Pattern Counter Preload Register 1 (CRC_PCOUNT_REG1) [offset = 40h] 526
14-18. CRC Sector Counter Preload Register 1 (CRC_SCOUNT_REG1) [offset = 44h] 527
14-19. CRC Current Sector Preload Register 1 (CRC_CURSEC_REG1) [offset = 48h] 527
14-20. CRC Channel 1 Watchdog Timeout Preload Register A (CRC_WDTOPLD1) [offset = 4Ch] 528
14-21. CRC Channel 1 Block Complete Timeout Preload Register B (CRC_BCTOPLD1) [offset = 50h] 528
14-22. Channel 1 PSA Signature Low Register (PSA_SIGREGL1) [offset = 60h] 529
14-23. Channel 1 PSA Signature High Register (PSA_SIGREGH1) [offset = 64h] 529
14-24. Channel 1 CRC Value Low Register (CRC_REGL1) [offset = 68h] 529
14-25. Channel 1 CRC Value High Register (CRC_REGH1) [offset = 6Ch] 530
14-26. Channel 1 PSA Sector Signature Low Register (PSA_SECSIGREGL1) [offset = 70h] 530
14-27. Channel 1 PSA Sector Signature High Register (PSA_SECSIGREGH1) [offset = 74h] 530
14-28. Channel 1 Raw Data Low Register (RAW_DATAREGL1) [offset = 78h] 531
14-29. Channel 1 Raw Data High Register (RAW_DATAREGH1) [offset = 7Ch] 531
14-30. CRC Pattern Counter Preload Register 2 (CRC_PCOUNT_REG2) [offset = 80h] 531
14-31. CRC Sector Counter Preload Register 2 (CRC_SCOUNT_REG2) [offset = 84h] 532
14-32. CRC Current Sector Register 2 (CRC_CURSEC_REG2) [offset = 88h] 532
14-33. CRC Channel 2 Watchdog Timeout Preload Register A (CRC_WDTOPLD2) [offset = 8Ch] 533
14-34. CRC Channel 2 Block Complete Timeout Preload Register B (CRC_BCTOPLD2) [offset = 90h] 533
14-35. Channel 2 PSA Signature Low Register (PSA_SIGREGL2) [offset = A0h] 534
14-36. Channel 2 PSA Signature High Register (PSA_SIGREGH2) [offset = A4h] 534
14-37. Channel 2 CRC Value Low Register (CRC_REGL2) [offset = A8h] 534
14-38. Channel 2 CRC Value High Register (CRC_REGH2) [offset = ACh] 535
14-39. Channel 2 PSA Sector Signature Low Register (PSA_SECSIGREGL2) [offset = B0h] 535
14-40. Channel 2 PSA Sector Signature High Register (PSA_SECSIGREGH2) [offset = B4h] 535
14-41. Channel 2 Raw Data Low Register (RAW_DATAREGL2) [offset = B8h] 536
14-42. Channel 2 Raw Data High Register (RAW_DATAREGH2) [offset = BCh] 536
14-43. Data Bus Selection Register (CRC_TRACE_BUS_SEL) [offset = 140h] 537
15-1. Device Level Interrupt Block Diagram 539
15-2. VIM Interrupt Handling Block Diagram 542

15-3. VIM Channel Mapping 543
15-4. VIM in Default State 544
15-5. VIM in a Programmed State 544
15-6. Interrupt Channel Management 545
15-7. VIM Interrupt Address Memory Map 546
15-8. Parity Bit Mapping 548
15-9. Detail of the IRQ Input 549
15-10. Capture Event Sources 550
15-11. Interrupt Vector Table Parity Flag Register (PARFLG) [offset = ECh] 554
15-12. Interrupt Vector Table Parity Control Register (PARCTL) [offset = F0h] 554
15-13. Address Parity Error Register (ADDERR) [offset = F4h] 555
15-14. Fall-Back Address Parity Error Register (FBPARERR) [offset = F8h] 555
15-15. IRQ Index Offset Vector Register (IRQINDEX) [offset = 00h] 557
15-16. FIQ Index Offset Vector Register (FIQINDEX) [offset = 04h] 557
15-17. FIQ/IRQ Program Control Register 0 (FIRQPR0) [offset = 10h] 558
15-18. FIQ/IRQ Program Control Register 1 (FIRQPR1) [offset = 14h] 558
15-19. FIQ/IRQ Program Control Register 2 (FIRQPR2) [offset = 18h] 558
15-20. FIQ/IRQ Program Control Register 3 (FIRQPR3) [offset = 1Ch] 558
15-21. Pending Interrupt Read Location Register 0 (INTREQ0) [offset = 20h] 560
15-22. Pending Interrupt Read Location Register 1 (INTREQ1) [offset = 24h] 560
15-23. Pending Interrupt Read Location Register 2 (INTREQ2) [offset = 28h] 560
15-24. Pending Interrupt Read Location Register 3 (INTREQ3) [offset = 2Ch] 560
15-25. Interrupt Enable Set Register 0 (REQENASET0) [offset = 30h] 562
15-26. Interrupt Enable Set Register 1 (REQENASET1) [offset = 34h] 562
15-27. Interrupt Enable Set Register 2 (REQENASET2) [offset = 38h] 562
15-28. Interrupt Enable Set Register 3 (REQENASET3) [offset = 3Ch] 562
15-29. Interrupt Enable Clear Register 0 (REQENACLR0) [offset = 40h] 564
15-30. Interrupt Enable Clear Register 1 (REQENACLR1) [offset = 44h] 564
15-31. Interrupt Enable Clear Register 2 (REQENACLR2) [offset = 48h] 564
15-32. Interrupt Enable Clear Register 3 (REQENACLR3) [offset = 4Ch] 564
15-33. Wake-Up Enable Set Register 0 (WAKEENASET0) [offset = 50h] 566
15-34. Wake-Up Enable Set Register 1 (WAKEENASET1) [offset = 54h] 566
15-35. Wake-Up Enable Set Register 2 (WAKEENASET2) [offset = 58h] 566
15-36. Wake-Up Enable Set Register 3 (WAKEENASET3) [offset = 5Ch] 566
15-37. Wake-Up Enable Clear Register 0 (WAKEENACLR0) [offset = 60h] 568
15-38. Wake-Up Enable Clear Register 1 (WAKEENACLR1) [offset = 64h] 568
15-39. Wake-Up Enable Clear Register 2 (WAKEENACLR2) [offset = 68h] 568
15-40. Wake-Up Enable Clear Register 3 (WAKEENACLR3) [offset = 6Ch] 568
15-41. IRQ Interrupt Vector Register (IRQVECREG) [offset = 70h] 570
15-42. IRQ Interrupt Vector Register (FIQVECREG) [offset = 74h] 570
15-43. Capture Event Register (CAPEVT) [offset = 78h] 571
15-44. Interrupt Control Registers (CHANCTRL[0:31]) [offset = 80h-FCh] 572
16-1. DMA Block Diagram 575
16-2. Example of a DMA Transfer Using Frame Trigger Source 576
16-3. Example of a DMA Transfer Using Block Trigger Source 576
16-4. DMA Request Mapping and Control Packet Organization 578
16-5. Control Packet Organization and Memory Map 578
16-6. DMA Transfer Example 1 580
16-7. DMA Indexing Example 1 580

16-8. DMA Indexing Example 2 581
16-9. Fixed Priority Scheme 581
16-10. Example of Priority Queues 582
16-11. Example Channel Assignments 583
16-12. Example of DMA Data Unpacking 584
16-13. Example of DMA Data Packing 585
16-14. DMA Interrupts 589
16-15. Detailed Interrupt Structure (Frame Transfer Complete Path) 589
16-16. Example of Channel Chaining 592
16-17. Example of Protection Mechanism 593
16-18. Global Control Register (GCTRL) [offset = 00] 597
16-19. Channel Pending Register (PEND) [offset = 04h] 598
16-20. DMA Status Register (DMASTAT) [offset = 0Ch] 598
16-21. HW Channel Enable Set and Status Register (HWCHENAS) [offset = 14h] 599
16-22. HW Channel Enable Reset and Status Register (HWCHENAR) [offset = 1Ch] 599
16-23. SW Channel Enable Set and Status Register (SWCHENAS) [offset = 24h] 600
16-24. SW Channel Enable Reset and Status Register (SWCHENAR) [offset = 2Ch] 601
16-25. Channel Priority Set Register (CHPRIOS) [offset = 34h] 601
16-26. Channel Priority Reset Register (CHPRIOR) [offset = 3Ch] 602
16-27. Global Channel Interrupt Enable Set Register (GCHIENAS) [offset = 44h] 602
16-28. Global Channel Interrupt Enable Reset Register (GCHIENAR) [offset = 4Ch] 603
16-29. DMA Request Assignment Register 0 (DREQASI0) [offset = 54h] 604
16-30. DMA Request Assignment Register 1 (DREQASI1) [offset = 58h] 605
16-31. DMA Request Assignment Register 2 (DREQASI2) [offset = 5Ch] 606
16-32. DMA Request Assignment Register 3 (DREQASI3) [offset = 60h] 607
16-33. Port Assignment Register 0 (PAR0) [offset = 94h] 608
16-34. Port Assignment Register 1 (PAR1) [offset = 98h] 609
16-35. FTC Interrupt Mapping Register (FTCMAP) [offset = B4h] 610
16-36. LFS Interrupt Mapping Register (LFSMAP) [offset = BCh] 610
16-37. HBC Interrupt Mapping Register (HBCMAP) [offset = C4h] 611
16-38. BTC Interrupt Mapping Register (BTCMAP) [offset = CCh] 611
16-39. FTC Interrupt Enable Set (FTCINTENAS) [offset = DCh] 612
16-40. FTC Interrupt Enable Reset (FTCINTENAR) [offset = E4h] 612
16-41. LFS Interrupt Enable Set (LFSINTENAS) [offset = ECh] 613
16-42. LFS Interrupt Enable Reset (LFSINTENAR) [offset = F4h] 613
16-43. HBC Interrupt Enable Set (HBCINTENAS) [offset = FCh] 614
16-44. HBC Interrupt Enable Reset (HBCINTENAR) [offset = 104h] 614
16-45. BTC Interrupt Enable Set (BTCINTENAS) [offset = 10Ch] 615
16-46. BTC Interrupt Enable Reset (BTCINTENAR) [offset = 114h] 615
16-47. Global Interrupt Flag Register (GINTFLAG) [offset = 11Ch] 616
16-48. FTC Interrupt Flag Register (FTCFLAG) [offset = 124h] 616
16-49. LFS Interrupt Flag Register (LFSFLAG) [offset = 12Ch] 617
16-50. HBC Interrupt Flag Register (HBCFLAG) [offset = 134h] 617
16-51. BTC Interrupt Flag Register (BTCFLAG) [offset = 13Ch] 618
16-52. FTCA Interrupt Channel Offset Register (FTCAOFFSET) [offset = 14Ch] 619
16-53. LFSA Interrupt Channel Offset Register (LFSAOFFSET) [offset = 150h] 619
16-54. HBCA Interrupt Channel Offset Register (HBCAOFFSET) [offset = 154h] 621
16-55. BTCA Interrupt Channel Offset Register (BTCAOFFSET) [offset = 158h] 621
16-56. FTCB Interrupt Channel Offset Register (FTCBOFFSET) [offset = 160h] 623

16-57. LFSB Interrupt Channel Offset Register (LFSBOFFSET) [offset = 164h] 623
16-58. HBCB Interrupt Channel Offset Register (HBCBOFFSET) [offset = 168h] 625
16-59. BTCB Interrupt Channel Offset Register (BTCBOFFSET) [offset = 16Ch] 625
16-60. Port Control Register (PTCRL) [offset = 178h] 627
16-61. RAM Test Control (RTCTRL) [offset = 17Ch] 628
16-62. Debug Control (DCTRL) [offset = 180h] 629
16-63. Watch Point Register (WPR) [offset = 184h] 630
16-64. Watch Mask Register (WMR) [offset = 188h] 630
16-65. Port B Active Channel Source Address Register (PBACDADDR) [offset = 198h] 631
16-66. Port B Active Channel Destination Address Register (PBACDADDR) [offset = 19Ch] 631
16-67. Port B Active Channel Transfer Count Register (PBACTC) [offset = 1A0h] 632
16-68. Parity Control Register (DMAPCR) [offset = 1A8h] 633
16-69. DMA Parity Error Address Register (DMAPAR) [offset = 1ACh] 634
16-70. DMA Memory Protection Control Register (DMAMPCTRL) [offset = 1B0h] 635
16-71. DMA Memory Protection Status Register (DMAMPST) [offset = 1B4h] 637
16-72. DMA Memory Protection Region 0 Start Address Register (DMAMPR0S) [offset = 1B8h] 638
16-73. DMA Memory Protection Region 0 End Address Register (DMAMPR0E) [offset = 1BCh] 638
16-74. DMA Memory Protection Region 1 Start Address Register (DMAMPR1S) [offset = 1C0h] 639
16-75. DMA Memory Protection Region 1 End Address Register (DMAMPR1E) [offset = 1C4h] 639
16-76. DMA Memory Protection Region 2 Start Address Register (DMAMPR2S) [offset = 1C8h] 640
16-77. DMA Memory Protection Region 2 End Address Register (DMAMPR2E) [offset = 1CCh] 640
16-78. DMA Memory Protection Region 3 Start Address Register (DMAMPR3S) [offset = 1D0h] 641
16-79. DMA Memory Protection Region 3 End Address Register (DMAMPR3E) [offset = 1D4h] 641
16-80. Initial Source Address (ISADDR) [offset = 00] 642
16-81. Initial Destination Address Register (IDADDR) [offset = 04h] 642
16-82. Initial Transfer Count Register (ITCOUNT) [offset = 08h] 643
16-83. Channel Control Register (CHCTRL) [offset = 10h] 643
16-84. Element Index Offset Register (EIOFF) [offset = 14h] 645
16-85. Frame Index Offset Register (FIOFF) [offset = 18h] 645
16-86. Current Source Address Register (CSADDR) [offset = 800h] 646
16-87. Current Destination Address Register (CDADDR) [offset = 804h] 646
16-88. Current Transfer Count Register (CTCOUNT) [offset = 808h] 647
17-1. EMIF Functional Block Diagram 650
17-2. Timing Waveform of SDRAM PRE Command 654
17-3. EMIF to 2M × 16 × 4 bank SDRAM Interface 654
17-4. EMIF to 512K × 16 × 2 bank SDRAM Interface 655
17-5. Timing Waveform for Basic SDRAM Read Operation 662
17-6. Timing Waveform for Basic SDRAM Write Operation 663
17-7. EMIF Asynchronous Interface 665
17-8. EMIF to 8-bit/16-bit Memory Interface 666
17-9. Common Asynchronous Interface 666
17-10. Timing Waveform of an Asynchronous Read Cycle in Normal Mode 670
17-11. Timing Waveform of an Asynchronous Write Cycle in Normal Mode 672
17-12. Timing Waveform of an Asynchronous Read Cycle in Select Strobe Mode 674
17-13. Timing Waveform of an Asynchronous Write Cycle in Select Strobe Mode 676
17-14. Asynchronous Read in Page Mode 677
17-15. Module ID Register (MIDR) [offset = 00] 683
17-16. Asynchronous Wait Cycle Configuration Register (AWCCR) [offset = 04h] 684
17-17. SDRAM Configuration Register (SDCR) [offset = 08h] 685

17-18. SDRAM Refresh Control Register (SDRCR) [offset = 0Ch] 686
17-19. Asynchronous *n* Configuration Register (CE*n*CFG) [offset = 10h - 1Ch] 687
17-20. SDRAM Timing Register (SDTIMR) [offset = 20h] 688
17-21. SDRAM Self Refresh Exit Timing Register (SDSRETR) [offset = 3Ch] 689
17-22. EMIF Interrupt Raw Register (INTRAW) [offset = 40h] 690
17-23. EMIF Interrupt Mask Register (INTMSK) [offset = 44h] 691
17-24. EMIF Interrupt Mask Set Register (INTMSKSET) [offset = 48h] 692
17-25. EMIF Interrupt Mask Clear Register (INTMSKCLR) [offset = 4Ch] 693
17-26. Page Mode Control Register (PMCR) [offset = 68h] 694
17-27. Example Configuration Interface 696
17-28. SDRAM Timing Register (SDTIMR) 697
17-29. SDRAM Self Refresh Exit Timing Register (SDSRETR) 698
17-30. SDRAM Refresh Control Register (SDRCR) 699
17-31. SDRAM Configuration Register (SDCR) 700
17-32. LH28F800BJE-PTTL90 to EMIF Read Timing Waveforms 701
17-33. LH28F800BJE-PTTL90 to EMIF Write Timing Waveforms 702
17-34. Asynchronous *m* Configuration Register (*m* = 1, 2) (CE*n*CFG (*n* = 2, 3)) 703
18-1. System Overlay Block Diagram 706
18-2. Region Definition Example 707
18-3. POM Global Control Register (POMGLBCTRL) [address = FFA0 4000h] 709
18-4. POM Revision ID (POMREV) [address = FFA0 4004h] 710
18-5. POM Clock Gate Control Register [address = FFA0 4008h] 710
18-6. POM Status Register [address = FFA0 400Ch] 711
18-7. POM Program Region Start Register x (POMPROGSTARTx) [address = FFA0 4200h, FFA0 4210h, ..., FFA0 43F0h] 712
18-8. POM Overlay Region Start Register x (POMOVLSTARTx) [address = FFA0 4204h, FFA0 4214h, ..., FFA0 43F4h] 712
18-9. POM Region Size Register x (POMREGSIZEx) [address = FFA0 4208h, FFA0 4218h, ..., FFA0 43F8h] 713
18-10. POM Integration Control Register (POMITCTRL) [address = FFA0 4F00h] 713
18-11. POM Claim Set Register (POMCLAIMSET) [address = FFA0 4FA0h] 714
18-12. POM Claim Clear Register (POMCLAIMCLR) [address = FFA0 4FA4h] 715
18-13. POM Lock Access Register (POMLOCKACCESS) [address = FFA0 4FB0h] 716
18-14. POM Lock Status Register (POMLOCKSTATUS) [address = FFA0 4FB4h] 716
18-15. POM Authentication Status Register (POMAUTHSTATUS) [address = FFA0 4FB8h] 716
18-16. POM Device ID Register (POMDEVID) [address = FFA0 4FC8h] 717
18-17. POM Device Type Register (POMDEVTYPE) [address = FFA0 4FCCh] 717
18-18. POM Peripheral ID 4 Register (POMPERIPHERALID4) [address = FFA0 4FD0h] 718
18-19. POM Peripheral ID 5 Register (POMPERIPHERALID5) [address = FFA0 4FD4h] 718
18-20. POM Peripheral ID 6 Register (POMPERIPHERALID6) [address = FFA0 4FD8h] 719
18-21. POM Peripheral ID 7 Register (POMPERIPHERALID7) [address = FFA0 4FDCh] 719
18-22. POM Peripheral ID 0 Register (POMPERIPHERALID0) [address = FFA0 4FE0h] 720
18-23. POM Peripheral ID 1 Register (POMPERIPHERALID1) [address = FFA0 4FE4] 720
18-24. POM Peripheral ID 2 Register (POMPERIPHERALID2) [address = FFA0 4FE8h] 721
18-25. POM Peripheral ID 3 Register (POMPERIPHERALID3) [address = FFA0 4FECh] 721
18-26. POM Component ID 0 Register (POMCOMPONENTID0) [address = FFA0 4FF0h] 722
18-27. POM Component ID 1 Register (POMCPOMPONENTID1) [address = FFA0 4FF4h] 722
18-28. POM Component ID 2 Register (POMCPOMPONENTID2) [address = FFA0 4FF8h] 723
18-29. POM Component ID 3 Register (POMCPOMPONENTID3) [address = FFA0 4FFCh] 723
19-1. Multiple ePWM Modules 726

19-2. Submodules and Signal Connections for an ePWM Module 727
19-3. Time-Base Submodule Block Diagram 731
19-4. Time-Base Submodule Signals and Registers 732
19-5. Time-Base Frequency and Period 734
19-6. Time-Base Counter Synchronization Scheme 736
19-7. Time-Base Up-Count Mode Waveforms 738
19-8. Time-Base Down-Count Mode Waveforms 739
19-9. Time-Base Up-Down-Count Waveforms, TBCTL[PHSDIR = 0] Count Down On Synchronization Event 740
19-10. Time-Base Up-Down Count Waveforms, TBCTL[PHSDIR = 1] Count Up On Synchronization Event 741
19-11. Counter-Compare Submodule 742
19-12. Detailed View of the Counter-Compare Submodule 742
19-13. Counter-Compare Event Waveforms in Up-Count Mode 745
19-14. Counter-Compare Events in Down-Count Mode 746
19-15. Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 0] Count Down On Synchronization Event 747
19-16. Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 1] Count Up On Synchronization Event 747
19-17. Action-Qualifier Submodule 748
19-18. Action-Qualifier Submodule Inputs and Outputs 749
19-19. Possible Action-Qualifier Actions for EPWMxA and EPWMxB Outputs 750
19-20. Up-Down-Count Mode Symmetrical Waveform 753
19-21. Up, Single Edge Asymmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB—Active High 754
19-22. Up, Single Edge Asymmetric Waveform With Independent Modulation on EPWMxA and EPWMxB—Active Low 756
19-23. Up-Count, Pulse Placement Asymmetric Waveform With Independent Modulation on EPWMxA 757
19-24. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Active Low 759
19-25. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Complementary 760
19-26. Up-Down-Count, Dual Edge Asymmetric Waveform, With Independent Modulation on EPWMxA—Active Low 761
19-27. Dead_Band Submodule 762
19-28. Configuration Options for the Dead-Band Submodule 763
19-29. Dead-Band Waveforms for Typical Cases (0% < Duty < 100%) 765
19-30. PWM-Chopper Submodule 767
19-31. PWM-Chopper Submodule Operational Details 768
19-32. Simple PWM-Chopper Submodule Waveforms Showing Chopping Action Only 768
19-33. PWM-Chopper Submodule Waveforms Showing the First Pulse and Subsequent Sustaining Pulses 769
19-34. PWM-Chopper Submodule Waveforms Showing the Pulse Width (Duty Cycle) Control of Sustaining Pulses 770
19-35. Trip-Zone Submodule 771
19-36. Trip-Zone Submodule Mode Control Logic 775
19-37. Trip-Zone Submodule Interrupt Logic 776
19-38. Event-Trigger Submodule 777
19-39. Event-Trigger Submodule Inter-Connectivity of ADC Start of Conversion 778
19-40. Event-Trigger Submodule Showing Event Inputs and Prescaled Outputs 779
19-41. Event-Trigger Interrupt Generator 780
19-42. Event-Trigger SOCA Pulse Generator 781
19-43. Event-Trigger SOCB Pulse Generator 781
19-44. Digital-Compare Submodule High-Level Block Diagram 782

19-45. DCAEVT1 Event Triggering ... 784
19-46. DCAEVT2 Event Triggering ... 785
19-47. DCBEVT1 Event Triggering ... 785
19-48. DCBEVT2 Event Triggering ... 786
19-49. Event Filtering ... 787
19-50. Blanking Window Timing Diagram ... 787
19-51. Simplified ePWM Module ... 788
19-52. EPWM1 Configured as a Typical Master, EPWM2 Configured as a Slave ... 789
19-53. Control of Four Buck Stages. Here F_{PWM1}≠ F_{PWM2}≠ F_{PWM3}≠ F_{PWM4} ... 790
19-54. Buck Waveforms for (Note: Only three bucks shown here) ... 791
19-55. Control of Four Buck Stages. (Note: F_{PWM2} = N x F_{PWM1}) ... 793
19-56. Buck Waveforms for (Note: F_{PWM2} = $F_{PWM1)}$) ... 794
19-57. Control of Two Half-H Bridge Stages (F_{PWM2} = N x F_{PWM1}) ... 796
19-58. Half-H Bridge Waveforms for (Note: Here F_{PWM2} = F_{PWM1}) ... 797
19-59. Control of Dual 3-Phase Inverter Stages as Is Commonly Used in Motor Control ... 799
19-60. 3-Phase Inverter Waveforms for (Only One Inverter Shown) ... 800
19-61. Configuring Two PWM Modules for Phase Control ... 801
19-62. Timing Waveforms Associated With Phase Control Between 2 Modules ... 802
19-63. TBCTL Register ... 805
19-64. TBSTS Register ... 807
19-65. TBPHS Register ... 808
19-66. TBCTR Register ... 809
19-67. TBPRD Register ... 810
19-68. CMPCTL Register ... 811
19-69. CMPA Register ... 813
19-70. CMPB Register ... 814
19-71. AQCTLA Register ... 815
19-72. AQCTLB Register ... 817
19-73. AQSFRC Register ... 819
19-74. AQCSFRC Register ... 820
19-75. DBCTL Register ... 821
19-76. DBRED Register ... 823
19-77. DBFED Register ... 824
19-78. TZSEL Register ... 825
19-79. TZDCSEL Register ... 827
19-80. TZCTL Register ... 828
19-81. TZEINT Register ... 829
19-82. TZFLG Register ... 830
19-83. TZCLR Register ... 831
19-84. TZFRC Register ... 832
19-85. ETSEL Register ... 833
19-86. ETPS Register ... 835
19-87. ETFLG Register ... 837
19-88. ETCLR Register ... 838
19-89. ETFRC Register ... 839
19-90. PCCTL Register ... 840
19-91. DCTRIPSEL Register ... 841
19-92. DCACTL Register ... 842
19-93. DCBCTL Register ... 843

19-94. DCFCTL Register 844
19-95. DCCAPCTL Register 845
19-96. DCFOFFSET Register 846
19-97. DCFOFFSETCNT Register 847
19-98. DCFWINDOW Register 848
19-99. DCFWINDOWCNT Register 849
19-100. DCCAP Register 850
20-1. Capture and APWM Modes of Operation 853
20-2. Capture Function Diagram 854
20-3. Event Prescale Control 855
20-4. Prescale Function Waveforms 855
20-5. Details of the Continuous/One-shot Block 856
20-6. Details of the Counter and Synchronization Block 857
20-7. Interrupts in eCAP Module 858
20-8. PWM Waveform Details Of APWM Mode Operation 859
20-9. TSCTR Register 861
20-10. CTRPHS Register 862
20-11. CAP1 Register 863
20-12. CAP2 Register 864
20-13. CAP3 Register 865
20-14. CAP4 Register 866
20-15. ECCTL1 Register 867
20-16. ECCTL2 Register 869
20-17. ECEINT Register 871
20-18. ECFLG Register 872
20-19. ECCLR Register 873
20-20. ECFRC Register 874
20-21. Capture Sequence for Absolute Time-stamp and Rising Edge Detect 876
20-22. Capture Sequence for Absolute Time-stamp With Rising and Falling Edge Detect 878
20-23. Capture Sequence for Delta Mode Time-stamp and Rising Edge Detect 880
20-24. Capture Sequence for Delta Mode Time-stamp With Rising and Falling Edge Detect 882
20-25. PWM Waveform Details of APWM Mode Operation 884
21-1. Optical Encoder Disk 886
21-2. QEP Encoder Output Signal for Forward/Reverse Movement 886
21-3. Index Pulse Example 887
21-4. Functional Block Diagram of the eQEP Peripheral 889
21-5. Functional Block Diagram of Decoder Unit 891
21-6. Quadrature Decoder State Machine 893
21-7. Quadrature-clock and Direction Decoding 893
21-8. Position Counter Reset by Index Pulse for 1000 Line Encoder (QPOSMAX = 3999 or 0xF9F) 895
21-9. Position Counter Underflow/Overflow (QPOSMAX = 4) 896
21-10. Software Index Marker for 1000-line Encoder (QEPCTL[IEL] = 1) 898
21-11. Strobe Event Latch (QEPCTL[SEL] = 1) 898
21-12. eQEP Position-compare Unit 899
21-13. eQEP Position-compare Event Generation Points 900
21-14. eQEP Position-compare Sync Output Pulse Stretcher 900
21-15. eQEP Edge Capture Unit 902
21-16. Unit Position Event for Low Speed Measurement (QCAPCTL[UPPS] = 0010) 902
21-17. eQEP Edge Capture Unit - Timing Details 903

21-18. eQEP Watchdog Timer 904
21-19. eQEP Unit Time Base 904
21-20. EQEP Interrupt Generation 905
21-21. QPOSCNT Register 907
21-22. QPOSINIT Register 908
21-23. QPOSMAX Register 909
21-24. QPOSCMP Register 910
21-25. QPOSILAT Register 911
21-26. QPOSSLAT Register 912
21-27. QPOSLAT Register 913
21-28. QUTMR Register 914
21-29. QUPRD Register 915
21-30. QWDTMR Register 916
21-31. QWDPRD Register 917
21-32. QDECCTL Register 918
21-33. QEPCTL Register 919
21-34. QCAPCTL Register 921
21-35. QPOSCTL Register 922
21-36. QEINT Register 923
21-37. QFLG Register 924
21-38. QCLR Register 926
21-39. QFRC Register 928
21-40. QEPSTS Register 929
21-41. QCTMR Register 930
21-42. QCPRD Register 931
21-43. QCTMRLAT Register 932
21-44. QCPRDLAT Register 933
22-1. Channel Assignments of Two ADC Cores 935
22-2. ADC Block Diagram 936
22-3. FIFO Implementation 941
22-4. Format of Conversion Result Read from FIFO, 12-bit ADC 942
22-5. Format of Conversion Result Read from FIFO, 10-bit ADC 942
22-6. ADC Memory Mapping 943
22-7. Format of Conversion Result Directly Read from ADC RAM, 12-bit ADC 943
22-8. Format of Conversion Result Directly Read from ADC RAM, 10-bit ADC 943
22-9. Conversion Results Storage 944
22-10. ADC Groups' Operating Mode Control and Status Registers 946
22-11. Example Look-Up Table Entry 948
22-12. Group1 Enhanced Channel Selection Mode Example 949
22-13. Self-Test and Calibration Logic 957
22-14. Mid-point Value Calculation 960
22-15. Self-Test and Calibration Logic 961
22-16. Timing for Self-Test Mode 962
22-17. Timing for Sample Capacitor Discharge Mode 963
22-18. ADC Memory Map in Parity Test Mode 965
22-19. GPIO Functionality of ADxEVT 965
22-20. ADRSTCR Register 970
22-21. ADOPMODECR Register 971
22-22. ADCLOCKCR Register 973

22-23. ADCALCR Register 974
22-24. ADEVMODECR Register 976
22-25. ADG1MODECR Register 979
22-26. ADG2MODECR Register 982
22-27. ADEVSRC Register 985
22-28. ADG1SRC Register 986
22-29. ADG2SRC Register 987
22-30. ADEVINTENA Register 988
22-31. ADG1INTENA Register 989
22-32. ADG2INTENA Register 990
22-33. ADEVINTFLG Register 991
22-34. ADG1INTFLG Register 992
22-35. ADG2INTFLG Register 993
22-36. ADEVTHRINTCR Register 994
22-37. ADG1THRINTCR Register 995
22-38. ADG2THRINTCR Register 996
22-39. ADEVDMACR Register 997
22-40. ADG1DMACR Register 999
22-41. ADG2DMACR Register 1001
22-42. ADBNDCR Register 1003
22-43. ADBNDEND Register 1004
22-44. ADEVSAMP Register 1005
22-45. ADG1SAMP Register 1006
22-46. ADG2SAMP Register 1007
22-47. ADEVSR Register 1008
22-48. ADG1SR Register 1009
22-49. ADG2SR Register 1010
22-50. ADEVSEL Register 1011
22-51. ADG1SEL Register 1012
22-52. ADG2SEL Register 1013
22-53. ADCALR Register 1014
22-54. ADSMSTATE Register 1015
22-55. ADLASTCONV Register 1016
22-56. ADEVBUFFER_0 to ADEVBUFFER_7 Register (12-bit) 1017
22-57. ADEVBUFFER_0 to ADEVBUFFER_7 Register (10-bit) 1017
22-58. ADG1BUFFER_0 to ADG1BUFFER_7 Register (12-bit) 1019
22-59. ADG1BUFFER_0 to ADG1BUFFER_7 Register (10-bit) 1019
22-60. ADG2BUFFER_0 to ADG2BUFFER_7 Register (12-bit) 1021
22-61. ADG2BUFFER_0 to ADG2BUFFER_7 Register (10-bit) 1021
22-62. ADEVEMUBUFFER Register (12-bit) 1023
22-63. ADEVEMUBUFFER Register (10-bit) 1023
22-64. ADG1EMUBUFFER Register (12-bit) 1025
22-65. ADG1EMUBUFFER Register (10-bit) 1025
22-66. ADG2EMUBUFFER Register (12-bit) 1027
22-67. ADG2EMUBUFFER Register (10-bit) 1027
22-68. ADEVTDIR Register 1029
22-69. ADEVTOUT Register 1030
22-70. ADEVTIN Register 1031
22-71. ADEVTSET Register 1032

22-72. ADEVTCLR Register 1033
22-73. ADEVTPDR Register 1034
22-74. ADEVTPDIS Register 1035
22-75. ADEVTPSEL Register 1036
22-76. ADEVSAMPDISEN Register 1037
22-77. ADG1SAMPDISEN Register 1038
22-78. ADG2SAMPDISEN Register 1039
22-79. ADMAGINT1CR Register (12-bit) 1040
22-80. ADMAGINT1CR Register (10-bit) 1040
22-81. ADMAGINT1MASK Register 1042
22-82. ADMAGINT2CR Register (12-bit) 1043
22-83. ADMAGINT2CR Register (10-bit) 1043
22-84. ADMAGINT2MASK Register 1045
22-85. ADMAGINT3CR Register (12-bit) 1046
22-86. ADMAGINT3CR Register (10-bit) 1046
22-87. ADMAGINT3MASK Register 1048
22-88. ADMAGINTENASET Register 1049
22-89. ADMAGINTENACLR Register 1050
22-90. ADMAGINTFLG Register 1051
22-91. ADMAGINTOFF Register 1052
22-92. ADEVFIFORESETCR Register 1053
22-93. ADG1FIFORESETCR Register 1054
22-94. ADG2FIFORESETCR Register 1055
22-95. ADEVRAMWRADDR Register 1056
22-96. ADG1RAMWRADDR Register 1057
22-97. ADG2RAMWRADDR Register 1058
22-98. ADPARCR Register 1059
22-99. ADPARADDR Register 1060
22-100. ADPWRUPDLYCTRL Register 1061
22-101. ADEVCHNSELMODECTRL Register 1062
22-102. ADG1CHNSELMODECTRL Register 1063
22-103. ADG2CHNSELMODECTRL Register 1064
22-104. ADEVCURRCOUNT Register 1065
22-105. ADEVMAXCOUNT Register 1066
22-106. ADG1CURRCOUNT Register 1067
22-107. ADG1MAXCOUNT Register 1068
22-108. ADG2CURRCOUNT Register 1069
22-109. ADG2MAXCOUNT Register 1070
23-1. N2HET Block Diagram 1073
23-2. Specialized Timer Micromachine 1077
23-3. Program Flow Timings 1078
23-4. Use of the Overflow Interrupt Flag (HETEXC2) 1079
23-5. Multi-Resolution Operation Flow Example 1080
23-6. Debug Control Configuration 1081
23-7. Prescaler Configuration 1085
23-8. I/O Control 1088
23-9. N2HET Loop Resolution Structure for Each Bit 1089
23-10. Loop Resolution Instruction Execution Example 1090
23-11. HR I/O Architecture 1091

23-12. Example of HR Structure Sharing for N2HET Pins 0/1 1092
23-13. XOR-shared HR I/O 1093
23-14. Symmetrical PWM with XOR-sharing Output 1094
23-15. AND-shared HR I/O 1094
23-16. HR0 to HR1 Digital Loopback Logic: LBTYPE[0] = 0 1095
23-17. HR0 to HR1 Analog Loop Back Logic: LBTYPE[0] = 1 1096
23-18. N2HET Input Edge Detection 1097
23-19. ECMP Execution Timings 1098
23-20. High/Low Resolution Modes for ECMP and PWCNT 1099
23-21. PCNT Instruction Timing (With Capture Edge After HR Counter Overflow) 1100
23-22. PCNT Instruction Timing (With Capture Edge Before HR Counter Overflow) 1100
23-23. WCAP Instruction Timing 1101
23-24. I/O Block Diagram Including Pull Control Logic 1102
23-25. N2HET Pin Disable Feature Diagram 1103
23-26. Suppression Filter Counter Operation 1105
23-27. Interrupt Functionality on Instruction Level 1106
23-28. Interrupt Flag/Priority Level Architecture 1107
23-29. Request Line Assignment Example 1108
23-30. Operation of N2HET Count Instructions 1109
23-31. SCNT Count Operation 1109
23-32. ACNT Period Variation Compensations 1110
23-33. N2HET Timings Associated with the Gap Flag (ACNT Deceleration) 1111
23-34. N2HET Timings Associated with the Gap Flag (ACNT Acceleration) 1112
23-35. Angle Generator Principle 1113
23-36. Hardware Angle Generator Block Diagram 1114
23-37. Angle Tick Generation Principle 1115
23-38. New Angle Tick Generation Architecture 1116
23-39. Angle Generation Using Time Based Algorithm 1117
23-40. SCNT Stepping Compensation 1117
23-41. ACNT During Acceleration and Deceleration 1118
23-42. Singularity Check, ACNT Reset and Timing Associated 1119
23-43. Example of HWAG Start Sequence 1120
23-44. Code 1121
23-45. Gap Verification Criteria For a 60-2 Toothed Wheel 1122
23-46. Using the ARST Bit in a Toothed Wheel Without Singularity 1123
23-47. Windowing Filter for Toothed Wheel Input on Falling Active Edge 1124
23-48. Filtering During Singularity Tooth 1125
23-49. HWAG Interrupt Block Diagram 1126
23-50. Hardware Angle Generator/High End Timer Interface 1128
23-51. Angle Count Within the HWAG at Resolution Clock 1128
23-52. Angle Count Within the NHET With Increments 1129
23-53. Compare Without ACMP Instruction 1129
23-54. Example of ACMP Compare Within the NHET 1130
23-55. NHET Interface Block Diagram 1131
23-56. Global Configuration Register (HETGCR) [offset = 00h] 1136
23-57. Prescale Factor Register (HETPFR) 1138
23-58. N2HET Current Address (HETADDR) 1139
23-59. Offset Index Priority Level 1 Register (HETOFF1) 1139
23-60. Offset Index Priority Level 2 Register (HETOFF2) 1140

23-61. Interrupt Enable Set Register (HETINTENAS) 1141
23-62. Interrupt Enable Clear (HETINTENAC) 1141
23-63. Exception Control Register (HETEXC1) 1142
23-64. Exception Control Register 2 (HETEXC2) 1143
23-65. Interrupt Priority Register (HETPRY) 1144
23-66. Interrupt Flag Register (HETFLG) 1144
23-67. AND Share Control Register (HETAND) 1145
23-68. HR Share Control Register (HETHRSH) 1146
23-69. XOR Share Control Register (HETXOR) 1147
23-70. Request Enable Set Register (HETREQENS) 1148
23-71. Request Enable Clear Register (HETREQENC) 1148
23-72. Request Destination Select Register (HETREQDS) [offset = FFF7 B844h] 1149
23-73. N2HET Direction Register (HETDIR) 1150
23-74. N2HET Data Input Register (HETDIN) 1151
23-75. N2HET Data Output Register (HETDOUT) 1151
23-76. N2HET Data Set Register (HETDSET) 1152
23-77. N2HET Data Clear Register (HETDCLR) 1152
23-78. N2HET Open Drain Register (HETPDR) 1153
23-79. N2HET Pull Disable Register (HETPULDIS) 1153
23-80. N2HET Pull Select Register (HETPSL) 1154
23-81. Parity Control Register (HETPCR) 1155
23-82. Parity Address Register (HETPAR) 1156
23-83. Parity Pin Register (HETPPR) 1157
23-84. Suppression Filter Preload Register (HETSFPRLD) 1158
23-85. Suppression Filter Enable Register (HETSFENA) 1158
23-86. Loop Back Pair Select Register (HETLBPSEL) 1159
23-87. Loop Back Pair Direction Register (HETLBPDIR) 1160
23-88. N2HET Pin Disable Register (HETPINDIS) 1161
23-89. HWAG Pin Select Register (HWAPINSEL) 1163
23-90. HWAG Global Control Register 0 (HWAGCR0) 1164
23-91. HWAG Global Control Register 1 (HWAGCR1) 1164
23-92. HWAG Global Control Register 2 (HWAGCR2) 1165
23-93. HWAG Interrupt Enable Set Register (HWAENASET) 1166
23-94. HWAG Interrupt Enable Clear Register (HWAENACLR) 1167
23-95. HWAG Interrupt Level Set Register (HWALVLSET) 1168
23-96. HWAG Interrupt Level Clear Register (HWALVLCLR) 1168
23-97. HWAG Interrupt Flag Register (HWAFLG) 1169
23-98. HWAG Interrupt Offset Register 0 (HWAOFF0) 1170
23-99. HWAG Interrupt Offset Register 1 (HWAOFF1) 1171
23-100. HWAG Angle Value Register (HWAACNT) 1172
23-101. HWAG Previous Tooth Period Value Register (HWAPCNT1) 1173
23-102. HWAG Current Tooth Period Value Register (HWAPCNT) 1173
23-103. HWAG Step Width Register (HWASTWD) 1174
23-104. HWAG Teeth Number Register (HWATHNB) 1175
23-105. HWAG Current Teeth Number Register (HWATHVL) 1175
23-106. HWAG Filter Register (HWAFIL) 1176
23-107. HWAG Filter Register 2 (HWAFIL2) 1176
23-108. HWAG Angle Increment Register (HWAANGI) 1177
23-109. ACMP Program Field (P31:P0) 1183

23-110. ACMP Control Field (C31:C0) 1183
23-111. ACMP Data Field (D31:D0) 1183
23-112. ACNT Program Field (P31:P0) 1185
23-113. ACNT Control Field (C31:C0) 1185
23-114. ACNT Data Field (D31:D0) 1185
23-115. ADCNST Program Field (P31:P0) 1188
23-116. ADCNST Control Field (C31:C0) 1188
23-117. ADCNST Data Field (D31:D0) 1188
23-118. ADCNST Operation If Remote Data Field[31:7] Is Not Zero 1189
23-119. ADCNST Operation if Remote Data Field [31:7] Is Zero 1189
23-120. ADC, ADD, AND, OR, SBB, SUB, XOR Program Field (P31:P0) 1190
23-121. ADC, ADD, AND, OR, SBB, SUB, XOR Control Field (C31:C0) 1190
23-122. ADC, ADD, AND, OR, SBB, SUB, XOR Data Field (D31:D0) 1190
23-123. ADM32 Program Field (P31:P0) 1196
23-124. ADM32 Control Field (C31:C0) 1196
23-125. ADM32 Data Field (D31:D0) 1196
23-126. ADM32 Add and Move Operation for IM&REGTOREG (Case 00) 1198
23-127. ADM32 Add and Move Operation for REM&REGTOREG (Case 01) 1198
23-128. APCNT Program Field (P31:P0) 1199
23-129. APCNT Control Field (C31:C0) 1199
23-130. APCNT Data Field (D31:D0) 1199
23-131. BR Program Field (P31:P0) 1202
23-132. BR Control Field (C31:C0) 1202
23-133. BR Data Field (D31:D0) 1202
23-134. CNT Program Field (P31:P0) 1204
23-135. CNT Control Field (C31:C0) 1204
23-136. CNT Data Field (D31:D0) 1204
23-137. DADM64 Program Field (P31:P0) 1208
23-138. DADM64 Control Field (C31:C0) 1208
23-139. DADM64 Data Field (D31:D0) 1208
23-140. DADM64 Add and Move Operation 1208
23-141. DJZ Program Field (P31:P0) 1210
23-142. DJZ Control Field (C31:C0) 1210
23-143. DJZ Data Field (D31:D0) 1210
23-144. ECMP Program Field (P31:P0) 1212
23-145. ECMP Control Field (C31:C0) 1212
23-146. ECMP Data Field (D31:D0) 1212
23-147. ECNT Program Field (P31:P0) 1215
23-148. ECNT Control Field (C31:C0) 1215
23-149. ECNT Data Field (D31:D0) 1215
23-150. MCMP Program Field (P31:P0) 1217
23-151. MCMP Control Field (C31:C0) 1217
23-152. MCMP Data Field (D31:D0) 1217
23-153. MOV32 Program Field (P31:P0) 1220
23-154. MOV32 Control Field (C31:C0) 1220
23-155. MOV32 Data Field (D31:D0) 1220
23-156. MOV32 Move Operation for IMTOREG (Case 00) 1221
23-157. MOV32 Move Operation for IMTOREG&REM (Case 01) 1222
23-158. MOV32 Move Operation for REGTOREM (Case 10) 1222

23-159. MOV32 Move Operation for REMTOREG (Case 11) 1222
23-160. MOV64 Program Field (P31:P0) 1225
23-161. MOV64 Control Field (C31:C0) 1225
23-162. MOV64 Data Field (D31:D0) 1225
23-163. MOV64 Move Operation 1225
23-164. PCNT Program Field (P31:P0) 1227
23-165. PCNT Control Field (C31:C0) 1227
23-166. PCNT Data Field (D31:D0) 1227
23-167. PWCNT Program Field (P31:P0) 1230
23-168. PWCNT Control Field (C31:C0) 1230
23-169. PWCNT Data Field (D31:D0) 1230
23-170. RADM64 Program Field (P31:P0) 1234
23-171. RADM64 Control Field (C31:C0) 1234
23-172. RADM64 Data Field (D31:D0) 1234
23-173. RADM64 Add and Move Operation 1234
23-174. RCNT Program Field (P31:P0) 1236
23-175. RCNT Control Field (C31:C0) 1236
23-176. RCNT Data Field (D31:D0) 1236
23-177. SCMP Program Field (P31:P0) 1238
23-178. SCMP Control Field (C31:C0) 1238
23-179. SCMP Data Field (D31:D0) 1238
23-180. SCNT Program Field (P31:P0) 1240
23-181. SCNT Control Field (C31:C0) 1240
23-182. SCNT Data Field (D31:D0) 1240
23-183. SHFT Program Field (P31:P0) 1242
23-184. SHFT Control Field (C31:C0) 1242
23-185. SHFT Data Field (D31:D0) 1242
23-186. WCAP Program Field (P31:P0) 1245
23-187. WCAP Control Field (C31:C0) 1245
23-188. WCAP Data Field (D31:D0) 1245
23-189. WCAPE Program Field (P31:P0) 1247
23-190. WCAPE Control Field (C31:C0) 1247
23-191. WCAPE Data Field (D31:D0) 1247
24-1. System Block Diagram 1251
24-2. HTU Block Diagram 1252
24-3. Example of a HTU Transfer 1252
24-4. Single Buffer Timing and Memory Representation 1254
24-5. Timing Example for Circular Buffer Mode 1254
24-6. Dual Buffer Timing 1255
24-7. Timing Example for Auto Switch Buffer Mode 1256
24-8. Timing for Disabling Control Packets 1257
24-9. Timing Example Including Lost Requests 1258
24-10. Timing that Generates No Request Lost Error 1259
24-11. Timing that Generates a Request Lost Error 1259
24-12. Timing Example for Two WCAP Instructions 1260
24-13. Timing of the WCAP, ECNT, PCNT Example 1263
24-14. Global Control Register (HTU GC) [offset = 00] 1267
24-15. Control Packet Enable Register (HTU CPENA) [offset = 04h] 1268
24-16. Control Packet (CP) Busy Register 0 (HTU BUSY0) [offset = 08h] 1269

24-17. Control Packet (CP) Busy Register 1 (HTU BUSY1) [offset = 0Ch] 1270
24-18. Control Packet (CP) Busy Register 2 (HTU BUSY2) [offset = 10h] 1270
24-19. Control Packet (CP) Busy Register 3 (HTU BUSY3) [offset = 14h] 1271
24-20. Active Control Packet and Error Register (HTU ACPE) [offset = 18h] 1271
24-21. Request Lost and Bus Error Control Register (HTU RLBECTRL) [offset = 20h] 1273
24-22. Buffer Full Interrupt Enable Set Register (HTU BFINTS) [offset = 24h] 1274
24-23. Buffer Full Interrupt Enable Clear Register (HTU BFINTC) [offset = 28h] 1274
24-24. Interrupt Mapping Register (HTU INTMAP) [offset = 2Ch] 1275
24-25. Interrupt Offset Register 0 (HTU INTOFF0) [offset = 34h] 1276
24-26. Interrupt Offset Register 1 (HTU INTOFF1) [offset = 38h] 1277
24-27. Buffer Initialization Mode Register (HTU BIM) [offset = 3Ch] 1278
24-28. Request Lost Flag Register (HTU RLOSTFL) [offset = 40h] 1280
24-29. Buffer Full Interrupt Flag Register (HTU BFINTFL) [offset = 44h] 1280
24-30. BER Interrupt Flag Register (HTU BERINTFL) [offset = 48h] 1281
24-31. Memory Protection 1 Start Address Register (HTU MP1S) [offset = 4Ch] 1282
24-32. Memory Protection 1 End Address Register (HTU MP1E) [offset = 50h] 1282
24-33. Debug Control Register (HTU DCTRL) [offset = 54h] 1283
24-34. Watch Point Register (HTU WPR) [offset = 58h] 1284
24-35. Watch Mask Register (HTU WMR) [offset = 5Ch] 1284
24-36. Module Identification Register (HTU ID) [offset = 60h] 1285
24-37. Parity Control Register (HTU PCR) [offset = 64h] 1286
24-38. Parity Address Register (HTU PAR) [offset = 68h] 1287
24-39. Memory Protection Control and Status Register (HTU MPCS) [offset = 70h] 1288
24-40. Memory Protection Start Address Register 0 (HTU MP0S) [offset = 74h] 1291
24-41. Memory Protection End Address Register (HTU MP0E) [offset = 78h] 1291
24-42. Initial Full Address A Register (HTU IFADDRA) 1293
24-43. Initial Full Address B Register (HTU IFADDRB) 1293
24-44. Initial N2HET Address and Control Register (HTU IHADDRCT) 1294
24-45. Initial Transfer Count Register (HTU ITCOUNT) 1295
24-46. Current Full Address A Register (HTU CFADDRA) 1296
24-47. Current Full Address B Register (HTU CFADDRB) 1297
24-48. Current Frame Count Register (HTU CFCOUNT) 1298
25-1. I/O Function Quick Start Flow Chart 1303
25-2. Interrupt Generation Function Quick Start Flow Chart 1304
25-3. GIO Module Diagram 1305
25-4. GIO Block Diagram 1307
25-5. GIO Global Control Register (GIOGCR0) [offset = 00h] 1310
25-6. GIO Interrupt Detect Register (GIOINTDET) [offset = 08h] 1311
25-7. GIO Interrupt Polarity Register (GIOPOL) [offset = 0Ch] 1312
25-8. GIO Interrupt Enable Set Register (GIOENASET) [offset = 10h] 1313
25-9. GIO Interrupt Enable Clear Register (GIOENACLR) [offset = 14h] 1314
25-10. GIO Interrupt Priority Register (GIOLVLSET) [offset = 18h] 1315
25-11. GIO Interrupt Priority Register (GIOLVLCLR) [offset = 1Ch] 1317
25-12. GIO Interrupt Flag Register (GIOFLG) [offset = 20h] 1318
25-13. GIO Offset 1 Register (GIOOFF1) [offset = 24h] 1319
25-14. GIO Offset 2 Register (GIOOFF2) [offset = 28h] 1320
25-15. GIO Emulation 1 Register (GIOEMU1) [offset = 2Ch] 1321
25-16. GIO Emulation 2 Register (GIOEMU2) [offset = 30h] 1322
25-17. GIO Data Direction Registers (GIODIR[A-B]) [offset = 34h, 54h] 1323

25-18. GIO Data Input Registers (GIODIN[A-B]) [offset = 38h, 58h] 1323
25-19. GIO Data Output Registers (GIODOUT[A-B]) [offset = 3Ch, 5Ch] 1324
25-20. GIO Data Set Registers (GIODSET[A-B]) [offset = 40h, 60h] 1324
25-21. GIO Data Clear Registers (GIODCLR[A-B]) [offset = 44h, 64h] 1325
25-22. GIO Open Drain Registers (GIOPDR[A-B]) [offset = 48h, 68h] 1325
25-23. GIO Pull Disable Registers (GIOPULDIS[A-B]) [offset = 4Ch, 6Ch] 1326
25-24. GIO Pull Select Registers (GIOPSL[A-B]) [offset = 50h, 70h] 1326
26-1. Block Diagram 1330
26-2. Bit Timing 1332
26-3. CAN Bit-timing Configuration 1336
26-4. Structure of a Message Object 1338
26-5. Message RAM Representation in Debug/Suspend Mode 1341
26-6. Message RAM Representation in RAM Direct Access Mode 1341
26-7. Data Transfer Between IF1 / IF2 Registers and Message RAM 1343
26-8. Initialization of a Transmit Object 1345
26-9. Initialization of a Single Receive Object for Data Frames 1345
26-10. Initialization of a Single Receive Object for Remote Frames 1346
26-11. CPU Handling of a FIFO Buffer (Interrupt Driven) 1351
26-12. CAN Interrupt Topology 1 1354
26-13. CAN Interrupt Topology 2 1355
26-14. Local Power-Down Mode Flow Diagram 1357
26-15. CAN Core in Silent Mode 1358
26-16. CAN Core in Loop Back Mode 1359
26-17. CAN Core in External Loop Back Mode 1360
26-18. CAN Core in Loop Back Combined with Silent Mode 1361
26-19. CAN Control Register (DCAN CTL) [offset = 00] 1365
26-20. Error and Status Register (DCAN ES) [offset = 04h] 1367
26-21. Error Counter Register (DCAN ERRC) [offset = 08h] 1369
26-22. Bit Timing Register (DCAN BTR) [offset = 0Ch] 1370
26-23. Interrupt Register (DCAN INT) [offset = 10h] 1371
26-24. Test Register (DCAN TEST) [offset = 14h] 1372
26-25. Parity Error Code Register (DCAN PERR) [offset = 1Ch] 1373
26-26. Core Release Register (DCAN REL) [offset = 20h] 1373
26-27. Auto-Bus-On Time Register (DCAN ABOTR) [offset = 80h] 1374
26-28. Transmission Request X Register (DCAN TXRQ X) [offset = 84h] 1374
26-29. Transmission Request 12 Register [offset = 88h] 1375
26-30. Transmission Request 34 Register [offset = 8Ch] 1375
26-31. Transmission Request 56 Register [offset = 90h] 1375
26-32. Transmission Request 78 Register [offset = 94h] 1375
26-33. New Data X Register (DCAN NWDAT X) [offset = 98h] 1376
26-34. New Data 12 Register [offset = 9Ch] 1377
26-35. New Data 34 Register [offset = A0h] 1377
26-36. New Data 56 Register [offset = A4h] 1377
26-37. New Data 78 Register [offset = A8h] 1377
26-38. Interrupt Pending X Register (DCAN INTPND X) [offset = ACh] 1378
26-39. Interrupt Pending 12 Register [offset = B0h] 1379
26-40. Interrupt Pending 34 Register [offset = B4h] 1379
26-41. Interrupt Pending 56 Register [offset = B8h] 1379
26-42. Interrupt Pending 78 Register [offset = BCh] 1379

26-43. Message Valid X Register (DCAN MSGVAL X) [offset = C0h] 1380
26-44. Message Valid 12 Register [offset = C4h] 1381
26-45. Message Valid 34 Register [offset = C8h] 1381
26-46. Message Valid 56 Register [offset = CCh] 1381
26-47. Message Valid 78 Register [offset = D0h] 1381
26-48. Interrupt Multiplexer 12 Register [offset = D8h] 1382
26-49. Interrupt Multiplexer 34 Register [offset = DCh] 1382
26-50. Interrupt Multiplexer 56 Register [offset = E0h] 1382
26-51. Interrupt Multiplexer 78 Register [offset = E4h] 1382
26-52. IF1 Command Registers (DCAN IF1CMD) [offset = 100h] 1383
26-53. IF2 Command Registers (DCAN IF2CMD) [offset = 120h] 1383
26-54. IF1 Mask Register (DCAN IF1MSK) [offset = 104h] 1386
26-55. IF2 Mask Register (DCAN IF2MSK) [offset = 124h] 1386
26-56. IF1 Arbitration Register (DCAN IF1ARB) [offset = 108h] 1387
26-57. IF2 Arbitration Register (DCAN IF2ARB) [offset = 128h] 1387
26-58. IF1 Message Control Register (DCAN IF1MCTL) [offset = 10Ch] 1388
26-59. IF2 Message Control Register (DCAN IF2MCTL) [offset = 12Ch] 1388
26-60. IF1 Data A Register (DCAN IF1DATA) [offset = 110h] 1390
26-61. IF1 Data B Register (DCAN IF1DATB) [offset = 114h] 1390
26-62. IF2 Data A Register (DCAN IF2DATA) [offset = 130h] 1390
26-63. IF2 Data B Register (DCAN IF2DATB) [offset = 134h] 1390
26-64. IF3 Observation Register (DCAN IF3OBS) [offset = 140h] 1391
26-65. IF3 Mask Register (DCAN IF3MSK) [offset = 144h] 1393
26-66. IF3 Arbitration Register (DCAN IF3ARB) [offset = 148h] 1394
26-67. IF3 Message Control Register (DCAN IF3MCTL) [offset = 14Ch] 1395
26-68. IF3 Data A Register (DCAN IF3DATA) [offset = 150h] 1396
26-69. IF3 Data B Register (DCAN IF3DATB) [offset = 154h] 1396
26-70. IF3 Update Enable 12 Register [offset = 160h] 1397
26-71. IF3 Update Enable 34 Register [offset = 164h] 1397
26-72. IF3 Update Enable 56 Register [offset = 168h] 1397
26-73. IF3 Update Enable 78 Register [offset = 16Ch] 1397
26-74. CAN TX IO Control Register (DCAN TIOC) [offset = 1E0h] 1398
26-75. CAN RX IO Control Register (DCAN RIOC) [offset = 1E4h] 1399
27-1. SPI Functional Logic Diagram 1405
27-2. SPI Three-Pin Operation 1406
27-3. Operation with $\overline{\text{SPICS}}$ 1407
27-4. Operation with $\overline{\text{SPIENA}}$ 1408
27-5. SPI Five-Pin Option with $\overline{\text{SPIENA}}$ and $\overline{\text{SPICS}}$ 1409
27-6. Format for Transmitting an 12-Bit Word 1410
27-7. Format for Receiving an 10-Bit Word 1410
27-8. Clock Mode with Polarity = 0 and Phase = 0 1411
27-9. Clock Mode with Polarity = 0 and Phase = 1 1411
27-10. Clock Mode with Polarity = 1 and Phase = 0 1412
27-11. Clock Mode with Polarity = 1 and Phase = 1 1412
27-12. Five Bits per Character (5-Pin Option) 1413
27-13. Typical Diagram when a Buffer in Master is in CSHOLD Mode (SPI-SPI) 1415
27-14. Block Diagram Shift Register, MSB First 1417
27-15. Block Diagram Shift Register, LSB First 1417
27-16. 2-data Line Mode (Phase 0, Polarity 0) 1420

27-17. Two-Pin Parallel Mode Timing Diagram (Phase 0, Polarity 0) 1420
27-18. 4-Data Line Mode (Phase 0, Polarity 0) 1421
27-19. 4 Pins Parallel Mode Timing Diagram (Phase 0, Polarity 0) 1421
27-20. 8-data Line Mode (Phase 0, Polarity 0) 1422
27-21. 8 Pins Parallel Mode Timing Diagram (Phase 0, Polarity 0) 1423
27-22. I/O Paths during I/O Loopback Modes 1425
27-23. TG Interrupt Structure 1428
27-24. SPIFLG Interrupt Structure 1428
27-25. DMA Channel and Request Line (Logical) Structure in Multi-buffer Mode 1429
27-26. SPI Global Control Register 0 (SPIGCR0) [offset = 00] 1433
27-27. SPI Global Control Register 1 (SPIGCR1) [offset = 04h] 1434
27-28. SPI Interrupt Register (SPIINT0) [offset = 08h] 1435
27-29. SPI Interrupt Level Register (SPILVL) [offset = 0Ch] 1437
27-30. SPI Flag Register (SPIFLG) [offset = 10h] 1438
27-31. SPI Pin Control Register 0 (SPIPC0) [offset = 14h] 1441
27-32. SPI Pin Control Register 1 (SPIPC1) [offset = 18h] 1442
27-33. SPI Pin Control Register 2 (SPIPC2) [offset = 1Ch] 1444
27-34. SPI Pin Control Register 3 (SPIPC3) [offset = 20h] 1445
27-35. SPI Pin Control Register 4 (SPIPC4) [offset = 24h] 1446
27-36. SPI Pin Control Register 5 (SPIPC5) [offset = 28h] 1448
27-37. SPI Pin Control Register 6 (SPIPC6) [offset = 2Ch] 1449
27-38. SPI Pin Control Register 7 (SPIPC7) [offset = 30h] 1451
27-39. SPI Pin Control Register 8 (SPIPC8) [offset = 34h] 1452
27-40. SPI Transmit Data Register 0 (SPIDAT0) [offset = 38h] 1453
27-41. SPI Transmit Data Register 1 (SPIDAT1) [offset = 3Ch] 1455
27-42. SPI Receive Buffer Register (SPIBUF) [offset = 40h] 1458
27-43. SPI Emulation Register (SPIEMU) [offset = 44h] 1460
27-44. SPI Delay Register (SPIDELAY) [offset = 48h] 1460
27-45. Example: $t_{C2TDELAY}$= 8 VCLK Cycles 1462
27-46. Example: $t_{T2CDELAY}$= 4 VCLK Cycles 1462
27-47. Transmit-Data-Finished-to-ENA-Inactive-Timeout 1462
27-48. Chip-Select-Active-to-ENA-Signal-Active-Timeout 1462
27-49. SPI Default Chip Select Register (SPIDEF) [offset = 4Ch] 1463
27-50. SPI Data Format Registers (SPIFMT[3:0]) [offset = 5Ch-50h] 1464
27-51. Interrupt Vector 0 (NTVECT0) [offset = 60h] 1466
27-52. Interrupt Vector 1 (INTVECT1) [offset = 64h] 1467
27-53. SPI Pin Control Register 9 (SPIPC9) [offset = 68h] 1469
27-54. Parallel/Modulo Mode Control Register (SPIPMCTRL) [offset = 6Ch] 1470
27-55. Multi-buffer Mode Enable Register (MIBSPIE) [offset = 70h] 1473
27-56. TG Interrupt Enable Set Register (TGITENST) [offset = 74h] 1474
27-57. TG Interrupt Enable Clear Register (TGITENCR) [offset = 78h] 1475
27-58. Transfer Group Interrupt Level Set Register (TGITLVST) [offset = 7Ch] 1476
27-59. Transfer Group Interrupt Level Clear Register (TGITLVCR) [offset = 80h] 1477
27-60. Transfer Group Interrupt Flag Register (TGINTFLG) [offset = 84h] 1478
27-61. Tick Counter Operation 1479
27-62. Tick Count Register (TICKCNT) [offset = 90h] 1479
27-63. Last TG End Pointer (LTGPEND) [offset = 94h] 1480
27-64. MibSPI TG Control Registers (TGxCTRL) [offsets = 98h-D4h] 1481
27-65. DMA Channel Control Register (DMAxCTRL) [offset = D8h-F4h] 1484

27-66. DMAxCOUNT Register (ICOUNT) [offset = F8h-114h] 1486
27-67. DMA Large Count Register (DMACNTLEN) [offset = 118h] 1487
27-68. Multi-buffer RAM Uncorrectable Parity Error Control Register (UERRCTRL) [offset = 120h] 1487
27-69. Multi-buffer RAM Uncorrectable Parity Error Status Register (UERRSTAT) [offset = 124h] 1488
27-70. RXRAM Uncorrectable Parity Error Address Register (UERRADDR1) [offset = 128h] 1489
27-71. TXRAM Uncorrectable Parity Error Address Register (UERRADDR0) [offset = 12Ch] 1490
27-72. RXRAM Overrun Buffer Address Register (RXOVRN_BUF_ADDR) [offset = 130h] 1491
27-73. I/O-Loopback Test Control Register (IOLPBKTSTCR) [offset = 134h] 1492
27-74. SPI Extended Prescale Register 1 (EXTENDED_PRESCALE1 for SPIFMT0 and SPIFMT1) [offset = 138h] 1494
27-75. SPI Extended Prescale Register 2 (EXTENDED_PRESCALE2 for SPIFMT2 and SPIFMT3) [offset = 13Ch] 1496
27-76. Multi-Buffer RAM Configuration 1498
27-77. Multi-Buffer RAM Transmit Data Register (TXRAM) [offset = RAM Base + 0h-1FFh] 1500
27-78. Multi-Buffer RAM Receive Buffer Register (RXRAM) [offset = RAM Base + 200h-3FFh] 1503
27-79. Memory Map for Parity Locations During Normal and Test Mode 1506
27-80. Example of Memory-Mapped Parity Locations During Test Mode 1507
27-81. SPI/MibSPI Pins During Master Mode 3-pin Configuration 1508
27-82. SPI/MibSPI Pins During Master Mode 4-pin with $\overline{\text{SPICS}}$ Configuation 1508
27-83. SPI/MibSPI Pins During Master Mode 4-pin with $\overline{\text{SPIENA}}$ Configuration 1509
27-84. SPI/MibSPI Pins During Master/Slave Mode with 5-pin Configuration 1509
27-85. SPI/MibSPI Pins During Slave Mode 3-pin Configuration 1510
27-86. SPI/MibSPI Pins During Slave Mode 4-pin with $\overline{\text{SPIENA}}$ Configuration 1510
27-87. SPI/MibSPI Pins During Slave Mode in 5-pin Configuration - (Single Slave) 1510
27-88. SPI/MibSPI Pins During Slave Mode in 5-pin Configuration - (Single/Multi Slave) 1510
28-1. Detailed SCI Block Diagram 1516
28-2. SCI/LIN Block Diagram 1517
28-3. Typical SCI Data Frame Formats 1518
28-4. Asynchronous Communication Bit Timing 1519
28-5. Superfractional Divider Example 1521
28-6. Idle-Line Multiprocessor Communication Format 1523
28-7. Address-Bit Multiprocessor Communication Format 1523
28-8. Receive Buffers 1524
28-9. Transmit Buffers 1525
28-10. General Interrupt Scheme 1526
28-11. Interrupt Generation for Given Flags 1527
28-12. LIN Protocol Message Frame Format: Master Header and Slave Response 1534
28-13. Header 3 Fields: Synch Break, Synch, and ID 1534
28-14. Response Format of LIN Message Frame 1535
28-15. Message Header in Terms of T_{bit} 1538
28-16. ID Field 1538
28-17. Measurements for Synchronization 1540
28-18. Synchronization Validation Process and Baud Rate Adjustment 1541
28-19. Optional Embedded Checksum in Response for Extended Frames 1542
28-20. Checksum Compare and Send for Extended Frames 1543
28-21. TXRX Error Detector 1545
28-22. Classic Checksum Generation at Transmitting Node 1546
28-23. LIN 2.0-Compliant Checksum Generation at Transmitting Node 1546
28-24. ID Reception, Filtering and Validation 1548

28-25. LIN Message Frame Showing LIN Interrupt Timing and Sequence 1551
28-26. Wakeup Signal Generation 1555
28-27. SCI Global Control Register 0 (SCIGCR0) [offset = 00] 1558
28-28. SCI Global Control Register 1 (SCIGCR1) [offset = 04h] 1559
28-29. SCI Global Control Register 2 (SCIGCR2) [offset = 08h] 1563
28-30. SCI Set Interrupt Register (SCISETINT) [offset = 0Ch] 1565
28-31. SCI Clear Interrupt Register (SCICLEARINT) [offset = 10h] 1568
28-32. SCI Set Interrupt Level Register (SCISETINTLVL) [offset = 14h] 1572
28-33. SCI Clear Interrupt Level Register (SCICLEARINTLVL) [offset = 18h] 1575
28-34. SCI Flags Register (SCIFLR) [offset = 1Ch] 1578
28-35. SCI Interrupt Vector Offset 0 (SCIINTVECT0) [offset = 20h] 1585
28-36. SCI Interrupt Vector Offset 1 (SCIINTVECT1) [offset = 24h] 1585
28-37. SCI Format Control Register (SCIFORMAT) [offset = 28h] 1586
28-38. Baud Rate Selection Register (BRS) [offset = 2Ch] 1587
28-39. Receiver Emulation Data Buffer (SCIED) [offset = 30h] 1588
28-40. Receiver Data Buffer (SCIRD) [offset = 34h] 1589
28-41. Transmit Data Buffer Register (SCITD) [offset = 38h] 1590
28-42. SCI Pin I/O Control Register 0 (SCIPIO0) [offset = 3Ch] 1590
28-43. SCI Pin I/O Control Register 1 (SCIPIO1) [offset = 40h] 1591
28-44. SCI Pin I/O Control Register 2 (SCIPIO2) [offset = 44h] 1592
28-45. SCI Pin I/O Control Register 3 (SCIPIO3) [offset = 48h] 1593
28-46. SCI Pin I/O Control Register 4 (SCIPIO4) [offset = 4Ch] 1594
28-47. SCI Pin I/O Control Register 5 (SCIPIO5) [offset = 50h] 1595
28-48. SCI Pin I/O Control Register 6 (SCIPIO6) [offset = 54h] 1596
28-49. SCI Pin I/O Control Register 7 (SCIPIO7) [offset = 58h] 1597
28-50. SCI Pin I/O Control Register 8 (SCIPIO8) [offset = 5Ch] 1597
28-51. LIN Compare Register (LINCOMPARE) [offset = 60h] 1598
28-52. LIN Receive Buffer 0 Register (LINRD0) [offset = 64h] 1599
28-53. LIN Receive Buffer 1 Register (RD1) [offset = 68h] 1599
28-54. LIN Mask Register (LINMASK) [offset = 6Ch] 1600
28-55. LIN Identification Register (LINID) [offset = 70h] 1601
28-56. LIN Transmit Buffer 0 Register (LINTD0) [offset = 74h] 1602
28-57. LIN Transmit Buffer 1 Register (LINTD1) [offset = 78h] 1602
28-58. Maximum Baud Rate Selection Register (MBRS) [offset = 7Ch] 1603
28-59. Input/Output Error Enable Register (IODFTCTRL) [offset = 90h] 1604
28-60. GPIO Functionality 1606
29-1. Detailed SCI Block Diagram 1610
29-2. Typical SCI Data Frame Formats 1611
29-3. Asynchronous Communication Bit Timing 1612
29-4. Idle-Line Multiprocessor Communication Format 1614
29-5. Address-Bit Multiprocessor Communication Format 1615
29-6. General Interrupt Scheme 1616
29-7. Interrupt Generation for Given Flags 1617
29-8. SCI Global Control Register 0 (SCIGCR0) [offset = 00] 1624
29-9. SCI Global Control Register 1 (SCIGCR1) [offset = 04h] 1625
29-10. SCI Set Interrupt Register (SCISETINT) [offset = 0Ch] 1628
29-11. SCI Clear Interrupt Register (SCICLEARINT) [offset = 10h] 1630
29-12. SCI Set Interrupt Level Register (SCISETINTLVL) [offset = 14h] 1632
29-13. SCI Clear Interrupt Level Register (SCICLEARINTLVL) [offset = 18h] 1633

29-14. SCI Flags Register (SCIFLR) [offset = 1Ch] 1635
29-15. SCI Interrupt Vector Offset 0 (SCIINTVECT0) [offset = 20h] 1639
29-16. SCI Interrupt Vector Offset 1 (SCIINTVECT1) [offset = 24h] 1639
29-17. SCI Format Control Register (SCIFORMAT) [offset = 28h] 1640
29-18. Baud Rate Selection Register (BRS) [offset = 2Ch] 1641
29-19. Receiver Emulation Data Buffer (SCIED) [offset = 30h] 1642
29-20. Receiver Data Buffer (SCIRD) [offset = 34h] 1642
29-21. Transmit Data Buffer Register (SCITD) [offset = 38h] 1643
29-22. SCI Pin I/O Control Register 0 (SCIPIO0) [offset = 3Ch] 1643
29-23. SCI Pin I/O Control Register 1 (SCIPIO1) [offset = 40h] 1644
29-24. SCI Pin I/O Control Register 2 (SCIPIO2) [offset = 44h] 1645
29-25. SCI Pin I/O Control Register 3 (SCIPIO3) [offset = 48h] 1646
29-26. SCI Pin I/O Control Register 4 (SCIPIO4) [offset = 4Ch] 1647
29-27. SCI Pin I/O Control Register 5 (SCIPIO5) [offset = 50h] 1648
29-28. SCI Pin I/O Control Register 6 (SCIPIO6) [offset = 54h] 1649
29-29. SCI Pin I/O Control Register 7 (SCIPIO7) [offset = 58h] 1650
29-30. SCI Pin I/O Control Register 8 (SCIPIO8) [offset = 5Ch] 1650
29-31. Input/Output Error Enable Register (IODFTCTRL) [offset = 90h] 1651
29-32. GPIO Functionality 1653
30-1. Multiple I2C Modules Connection Diagram 1656
30-2. Simple I2C Block Diagram 1658
30-3. Clocking Diagram for the I2C Module 1659
30-4. Bit Transfer on the I2C Bus 1660
30-5. I2C Module START and STOP Conditions 1661
30-6. I2C Module Data Transfer 1661
30-7. I2C Module 7-Bit Addressing Format 1662
30-8. I2C Module 10-bit Addressing Format 1662
30-9. I2C Module 7-Bit Addressing Format with Repeated START 1662
30-10. I2C Module in Free Data Format 1663
30-11. Arbitration Procedure Between Two Master Transmitters 1666
30-12. Synchronization of Two I2C Clock Generators During Arbitration 1667
30-13. I2C Own Address Manager Register (I2COAR) [offset = 00h] 1672
30-14. I2C Interrupt Mask Register (I2CIMR) [offset = 04h] 1673
30-15. I2C Status Register (I2CSR) [offset = 08h] 1674
30-16. I2C Clock Divider Low Register (I2CCKL) [offset = 0Ch] 1677
30-17. I2C Clock Control High Register (I2CCKH) [offset = 10h] 1677
30-18. I2C Data Count Register (I2CCNT) [offset = 14h] 1678
30-19. I2C Data Receive Register (I2CDRR) [offset = 18h] 1678
30-20. I2C Slave Address Register (I2CSAR) [offset = 1Ch] 1679
30-21. I2C Data Transmit Register (I2CDXR) [offset = 20h] 1679
30-22. I2C Mode Register (I2CMDR) [offset = 24h] 1680
30-23. Typical Timing Diagram of Repeat Mode 1682
30-24. I2C Interrupt Vector Register (I2CIVR) [offset = 28h] 1683
30-25. I2C Extended Mode Register (I2CEMDR) [offset = 2Ch] 1684
30-26. I2C Prescale Register (I2CPSC) [offset = 30h] 1684
30-27. I2C Peripheral ID Register 1 (I2CPID1) [offset = 34h] 1685
30-28. I2C Peripheral ID Register 2 (I2CPID2) [offset = 38h] 1685
30-29. I2C DMA Control Register (I2CDMACR) [offset = 3Ch] 1686
30-30. I2C Pin Function Register (I2CPFNC) [offset = 48h] 1686

30-31. I2C Pin Direction Register (I2CPDIR) [offset = 4Ch] 1687
30-32. I2C Data Input Register (I2CDIN) [offset = 50h] 1687
30-33. I2C Data Output Register (I2CDOUT) [offset 0x54] 1688
30-34. I2C Data Set Register (I2CDSET) [offset = 58h] 1688
30-35. I2C Data Clear Register (I2CDCLR) [offset = 5Ch] 1689
30-36. I2C Pin Open Drain Register (I2CPDR) [offset = 60h] 1689
30-37. I2C Pull Disable Register (I2CPDIS) [offset = 64h] 1690
30-38. I2C Pull Select Register (I2CPSEL) [offset = 68h] 1690
30-39. I2C Pins Slew Rate Select Register (I2CSRS) [offset = 6Ch] 1691
30-40. Difference between Normal Operation and Backward Compatibility Mode 1692
31-1. EMAC and MDIO Block Diagram 1695
31-2. Ethernet Configuration—MII Connections 1697
31-3. Ethernet Configuration—RMII Connections 1699
31-4. Ethernet Frame Format 1701
31-5. Basic Descriptor Format 1702
31-6. Typical Descriptor Linked List 1703
31-7. Transmit Packet Add Flow Chart 1705
31-8. Generate Transmit Packet Flow Chart 1706
31-9. Transmit Queue Interrupt Processing Flow Chart 1707
31-10. Transmit Buffer Descriptor Format 1709
31-11. Receive Buffer Descriptor Format 1712
31-12. EMAC Control Module Block Diagram 1716
31-13. MDIO Module Block Diagram 1718
31-14. EMAC Module Block Diagram 1722
31-15. EMAC Control Module Revision ID Register (REVID) 1744
31-16. EMAC Control Module Software Reset Register (SOFTRESET) 1744
31-17. EMAC Control Module Interrupt Control Register (INTCONTROL) 1745
31-18. EMAC Control Module Receive Threshold Interrupt Enable Register (C0RXTHRESHEN) 1746
31-19. EMAC Control Module Receive Interrupt Enable Register (C0RXEN) 1747
31-20. EMAC Control Module Transmit Interrupt Enable Register (C0TXEN) 1748
31-21. EMAC Control Module Miscellaneous Interrupt Enable Register (C0MISCEN) 1749
31-22. EMAC Control Module Receive Threshold Interrupt Status Register (C0RXTHRESHSTAT) 1750
31-23. EMAC Control Module Receive Interrupt Status Register (C0RXSTAT) 1751
31-24. EMAC Control Module Transmit Interrupt Status Register (C0TXSTAT) 1752
31-25. EMAC Control Module Miscellaneous Interrupt Status Register (C0MISCSTAT) 1753
31-26. EMAC Control Module Receive Interrupts Per Millisecond Register (C0RXIMAX) 1754
31-27. EMAC Control Module Transmit Interrupts Per Millisecond Register (C0TXIMAX) 1755
31-28. MDIO Revision ID Register (REVID) 1756
31-29. MDIO Control Register (CONTROL) 1757
31-30. PHY Acknowledge Status Register (ALIVE) 1758
31-31. PHY Link Status Register (LINK) 1758
31-32. MDIO Link Status Change Interrupt (Unmasked) Register (LINKINTRAW) 1759
31-33. MDIO Link Status Change Interrupt (Masked) Register (LINKINTMASKED) 1760
31-34. MDIO User Command Complete Interrupt (Unmasked) Register (USERINTRAW) 1761
31-35. MDIO User Command Complete Interrupt (Masked) Register (USERINTMASKED) 1762
31-36. MDIO User Command Complete Interrupt Mask Set Register (USERINTMASKSET) 1763
31-37. MDIO User Command Complete Interrupt Mask Clear Register (USERINTMASKCLEAR) 1764
31-38. MDIO User Access Register 0 (USERACCESS0) 1765
31-39. MDIO User PHY Select Register 0 (USERPHYSEL0) 1766

31-40. MDIO User Access Register 1 (USERACCESS1) 1767
31-41. MDIO User PHY Select Register 1 (USERPHYSEL1) 1768
31-42. Transmit Revision ID Register (TXREVID) 1772
31-43. Transmit Control Register (TXCONTROL) 1772
31-44. Transmit Teardown Register (TXTEARDOWN) 1773
31-45. Receive Revision ID Register (RXREVID) 1773
31-46. Receive Control Register (RXCONTROL) 1774
31-47. Receive Teardown Register (RXTEARDOWN) 1774
31-48. Transmit Interrupt Status (Unmasked) Register (TXINTSTATRAW) 1775
31-49. Transmit Interrupt Status (Masked) Register (TXINTSTATMASKED) 1776
31-50. Transmit Interrupt Mask Set Register (TXINTMASKSET) 1777
31-51. Transmit Interrupt Mask Clear Register (TXINTMASKCLEAR) 1778
31-52. MAC Input Vector Register (MACINVECTOR) 1779
31-53. MAC End Of Interrupt Vector Register (MACEOIVECTOR) 1780
31-54. Receive Interrupt Status (Unmasked) Register (RXINTSTATRAW) 1781
31-55. Receive Interrupt Status (Masked) Register (RXINTSTATMASKED) 1782
31-56. Receive Interrupt Mask Set Register (RXINTMASKSET) 1783
31-57. Receive Interrupt Mask Clear Register (RXINTMASKCLEAR) 1784
31-58. MAC Interrupt Status (Unmasked) Register (MACINTSTATRAW) 1785
31-59. MAC Interrupt Status (Masked) Register (MACINTSTATMASKED) 1785
31-60. MAC Interrupt Mask Set Register (MACINTMASKSET) 1786
31-61. MAC Interrupt Mask Clear Register (MACINTMASKCLEAR) 1786
31-62. Receive Multicast/Broadcast/Promiscuous Channel Enable Register (RXMBPENABLE) 1787
31-63. Receive Unicast Enable Set Register (RXUNICASTSET) 1789
31-64. Receive Unicast Clear Register (RXUNICASTCLEAR) 1790
31-65. Receive Maximum Length Register (RXMAXLEN) 1790
31-66. Receive Buffer Offset Register (RXBUFFEROFFSET) 1791
31-67. Receive Filter Low Priority Frame Threshold Register (RXFILTERLOWTHRESH) 1791
31-68. Receive Channel *n* Flow Control Threshold Register (RX*n*FLOWTHRESH) 1792
31-69. Receive Channel *n* Free Buffer Count Register (RX*n*FREEBUFFER) 1792
31-70. MAC Control Register (MACCONTROL) 1793
31-71. MAC Status Register (MACSTATUS) 1795
31-72. Emulation Control Register (EMCONTROL) 1797
31-73. FIFO Control Register (FIFOCONTROL) 1797
31-74. MAC Configuration Register (MACCONFIG) 1798
31-75. Soft Reset Register (SOFTRESET) 1798
31-76. MAC Source Address Low Bytes Register (MACSRCADDRLO) 1799
31-77. MAC Source Address High Bytes Register (MACSRCADDRHI) 1799
31-78. MAC Hash Address Register 1 (MACHASH1) 1800
31-79. MAC Hash Address Register 2 (MACHASH2) 1800
31-80. Back Off Random Number Generator Test Register (BOFFTEST) 1801
31-81. Transmit Pacing Algorithm Test Register (TPACETEST) 1801
31-82. Receive Pause Timer Register (RXPAUSE) 1802
31-83. Transmit Pause Timer Register (TXPAUSE) 1802
31-84. MAC Address Low Bytes Register (MACADDRLO) 1803
31-85. MAC Address High Bytes Register (MACADDRHI) 1804
31-86. MAC Index Register (MACINDEX) 1804
31-87. Transmit Channel *n* DMA Head Descriptor Pointer Register (TX*n*HDP) 1805
31-88. Receive Channel *n* DMA Head Descriptor Pointer Register (RX*n*HDP) 1805

31-89. Transmit Channel *n* Completion Pointer Register (TX*n*CP) 1806
31-90. Receive Channel *n* Completion Pointer Register (RX*n*CP)........ 1806
31-91. Statistics Register 1807
32-1. OHCI Revision Number Register (HCREVISION) [address = FCF78B00h]........ 1821
32-2. HC Operating Mode Register (HCCONTROL) [address = FCF78B04h] 1821
32-3. HC Command and Status Register (HCCOMMANDSTATUS) [address = FCF78B08h] 1823
32-4. HC Interrupt Status Register (HCINTERRUPTSTATUS) [address = FCF78B0Ch] 1824
32-5. HC Interrupt Enable Register (HCINTERRUPTENABLE) [address = FCF78B10h]........ 1825
32-6. HC Interrupt Disable Register (HCINTERRUPTDISABLE) [address = FCF78B14h] 1826
32-7. HC HCCA Address Register (HCHCCA) [address = FCF78B18h] 1827
32-8. HC Current Periodic Register (HCPERIODCURRENTED) [address = FCF78B1Ch]........ 1828
32-9. HC Head Control Register (HCCONTROLHEADED) [address = FCF78B20h]........ 1828
32-10. HC Current Control Register (HCCONTROLCURRENTED) [address = FCF78B24h] 1829
32-11. HC Head Bulk Register (HCBULKHEADED) [address = FCF78B28h] 1829
32-12. HC Current Bulk Register (HCBULKCURRENTED) [address = FCF78B2Ch]........ 1830
32-13. HC Head Done Register (HCDONEHEAD) [address = FCF78B30h] 1830
32-14. HC Frame Interval Register (HCFMINTERVAL) [address = FCF78B34h]........ 1831
32-15. HC Frame Remaining Register (HCFMREMAINING) [address = FCF78B38h] 1831
32-16. HC Frame Number Register (HCFMNUMBER) [address = FCF78B3Ch]........ 1832
32-17. HC Periodic Start Register (HCPERIODICSTART) [address = FCF78B40h] 1832
32-18. HC Low-Speed Threshold Register (HCLSTHRESHOLD) [address = FCF78B44h]........ 1833
32-19. HC Root Hub A Register (HCRHDESCRIPTORA) [address = FCF78B48h]........ 1833
32-20. HC Root Hub B Register (HCRHDESCRIPTORB) [address = FCF78B4Ch] 1835
32-21. HC Root Hub Status Register (HCRHSTATUS) [address = FCF78B50h]........ 1836
32-22. HC Port 0 Status and Control Register (HCRHPORTSTATUS0) [address = FCF78B54h] 1837
32-23. HC Port 1 Status and Control Register (HCRHPORTSTATUS1) [address = FCF78B58h] 1839
32-24. Host UE Address Register (HOSTUEADDR) [address = FCF78BE0h]........ 1841
32-25. Host UE Status Register (HOSTUESTATUS) [address = FCF78BE4h] 1841
32-26. Host Time-out Control Register (HOSTTIMEOUTCTRL) [address = FCF78BE8h] 1842
32-27. Host Revision Register (HOSTREVISION) [address = FCF78BECh]........ 1842
32-28. Revision Register (REV) [address = FCF78A00h]........ 1846
32-29. Endpoint Selection Register (EP_NUM) [address = FCF78A02h] 1847
32-30. Data Register (DATA) [address = FCF78A04h]........ 1848
32-31. Control Register (CTRL) [address = FCF78A06h] 1849
32-32. Status Register (STAT_FLG) [address = FCF78A08h] 1850
32-33. Receive FIFO Status Register (RXFSTAT) [address = FCF78A0Ah] 1853
32-34. System Configuration Register 1 (SYSCON1) [address = FCF78A0Ch] 1854
32-35. System Configuration Register 2 (SYSCON2) [address = FCF78A0Eh] 1856
32-36. Device Status Register (DEVSTAT) [address = FCF78A10h]........ 1857
32-37. Start of Frame Register (SOF) [address = FCF78A12h]........ 1859
32-38. Interrupt Enable Register (IRQ_EN) [address = FCF78A14h] 1860
32-39. DMA Interrupt Enable Register (DMA_IRQ_EN) [address = FCF78A16h] 1861
32-40. Interrupt Source Register (IRQ_SRC) [address = FCF78A18h] 1862
32-41. Non-ISO Endpoint Interrupt Status Register (EPN_STAT) [address = FCF78A1Ah] 1865
32-42. Non-ISO DMA Interrupt Status Register (DMAN_STAT) [address = FCF78A1Ch] 1866
32-43. DMA Receive Channels Configuration Register (RXDMA_CFG) [address = FCF78A20h] 1867
32-44. DMA Transmit Channels Configuration Register (TXDMA_CFG) [address = FCF78A22h]........ 1868
32-45. DMA FIFO Data Register (DATA_DMA) [address = FCF78A24h]........ 1869
32-46. Transmit DMA Control Register n (TXDMAn) [address = FCF78A28h to FCF78A2Ch]........ 1870

32-47. Receive DMA Control Register n (RXDMAn) [address = FCF78A30h to FCF78A34h] 1871
32-48. Endpoint 0 Configuration Register (EP0) [address = FCF78A40h] 1872
32-49. Receive Endpoint n Configuration Register (EPn_RX) [address = FCF78A42h to FCF78A5Eh] 1873
32-50. Transmit Endpoint n Configuration Register (EPn_TX) [address = FCF78A62h to FCF78A7Eh] 1875
32-51. Non-Isochronous, Non-Control OUT Transaction Phases and Interrupts 1878
32-52. Non-Isochronous IN Transaction Phases and Interrupts 1882
32-53. Isochronous OUT Transaction Phases and Interrupts 1885
32-54. Isochronous IN Transaction Phases and Interrupts 1886
32-55. Stages and Transaction Phases of Autodecoded Control Transfers 1888
32-56. Stages and Transaction Phases of Non-Autodecoded Control Transfers 1889
32-57. Example of RAM Organization 1896
32-58. Device Configuration Routine 1897
32-59. Endpoint Configuration Routine 1898
32-60. Prepare for USB RX Transfers Routine 1899
32-61. Prepare for TX Transfer on Endpoint n Routine 1900
32-62. General USB Interrupt ISR Source Parsing Flowchart 1902
32-63. Setup Interrupt Handler 1903
32-64. Parse Command Routine (Setup Stage Control Transfer Request) 1904
32-65. Endpoint 0 RX Interrupt Handler 1905
32-66. Prepare for Control Write Status Stage Routine 1906
32-67. Endpoint 0 TX Interrupt Handler 1907
32-68. Prepare for Control Read Status Stage Routine 1908
32-69. USB Device Controller Device State Transitions 1910
32-70. Typical Operation for USB Device State Changed Interrupt Handler 1911
32-71. Attached/Unattached Handler 1912
32-72. Configuration Changed Handler 1913
32-73. Address Changed Handler 1914
32-74. USB Device Reset Handler Flowchart 1915
32-75. Typical Operation for USB Suspend/Resume General USB Interrupt Handler 1916
32-76. Non-ISO Endpoint-Specific (Except EP 0) ISR Flowchart 1917
32-77. Non-Isochronous Non-Control Endpoint Receive Interrupt Handler 1918
32-78. Read Non-Isochronous RX FIFO Data Flowchart 1919
32-79. Non-Isochronous Non-Control Endpoint Transmit Interrupt Handler 1920
32-80. Write Non-Isochronous TX FIFO Data Flowchart 1921
32-81. SOF Interrupt Handler Flowchart 1922
32-82. Read Isochronous RX FIFO Data Flowchart 1923
32-83. Write Isochronous TX FIFO Data Flowchart 1924
32-84. Non-ISO RX DMA Transaction Example (RX_TC=2) 1927
32-85. Non-ISO RX DMA Start Routine 1927
32-86. Non-ISO RX DMA EOT Interrupt Handler 1928
32-87. Non-ISO RX DMA Transactions Count Interrupt Handler 1929
32-88. ISO RX DMA Transaction 1929
32-89. ISO RX DMA Start Routine 1930
32-90. Non-ISO TX DMA Start Routine 1931
32-91. Non-ISO TX DMA Done Interrupt Handler 1932
32-92. ISO TX DMA Start Routine 1934
32-93. Power Management Signal Values 1936
32-94. Power Management Flowchart 1937
32-95. Pin Group 0 USB Host Using 6-Wire USB Transceiver 1940

32-96. USB Device Connections on USB Pin Group 0 Using 6-Wire Transceiver 1941
33-1. eFuse Self Test Flow Chart 1946
33-2. EFC Boundary Control Register (EFCBOUND) [offset = 1Ch] 1947
33-3. EFC Pins Register (EFCPINS) [offset = 2Ch] 1949
33-4. EFC Error Status Register (EFCERRSTAT) [offset = 3Ch] 1950
33-5. EFC Self Test Cycles Register (EFCSTCY) [offset = 48h] 1950
33-6. EFC Self Test Cycles Register (EFCSTSIG) [offset = 4Ch] 1951

List of Tables

2-1. Definition of Terms 102
2-2. Bus Master / Slave Access Privileges 104
2-3. Module Registers / Memories Memory-Map 106
2-4. Flash Memory Banks and Sectors 110
2-5. PBIST Memory Grouping 112
2-6. PBIST Algorithm Mapping 113
2-7. Memory Initialization Select Mapping 116
2-8. Causes of Resets 117
2-9. Clock Sources 120
2-10. Clock Domains 121
2-11. Typical Low-Power Modes 124
2-12. Clock Test Mode Options 126
2-13. DCC1 Counter 0 Clock Inputs 128
2-14. DCC1 Counter 1 Clock / Signal Inputs 128
2-15. DCC2 Counter 0 Clock Inputs 129
2-16. DCC2 Counter 1 Clock / Signal Inputs 129
2-17. Primary System Control Registers 130
2-18. SYS Pin Control Register 1 (SYSPC1) Field Descriptions 132
2-19. SYS Pin Control Register 2 (SYSPC2) Field Descriptions 132
2-20. SYS Pin Control Register 3 (SYSPC3) Field Descriptions 133
2-21. SYS Pin Control Register 4 (SYSPC4) Field Descriptions 133
2-22. SYS Pin Control Register 5 (SYSPC5) Field Descriptions 134
2-23. SYS Pin Control Register 6 (SYSPC6) Field Descriptions 134
2-24. SYS Pin Control Register 7 (SYSPC7) Field Descriptions 135
2-25. SYS Pin Control Register 8 (SYSPC8) Field Descriptions 135
2-26. SYS Pin Control Register 9 (SYSPC9) Field Descriptions 136
2-27. Clock Source Disable Register (CSDIS) Field Descriptions 137
2-28. Clock Sources Table 137
2-29. Clock Source Disable Set Register (CSDISSET) Field Descriptions 138
2-30. Clock Source Disable Clear Register (CSDISCLR) Field Descriptions 139
2-31. Clock Domain Disable Register (CDDIS) Field Descriptions 140
2-32. Clock Domain Disable Set Register (CDDISSET) Field Descriptions 142
2-33. Clock Domain Disable Clear Register (CDDISCLR) Field Descriptions 144
2-34. GCLK, HCLK, VCLK, and VCLK2 Source Register (GHVSRC) Field Descriptions 146
2-35. Peripheral Asynchronous Clock Source Register (VCLKASRC) Field Descriptions 147
2-36. RTI Clock Source Register (RCLKSRC) Field Descriptions 148
2-37. Clock Source Valid Register (CSVSTAT) Field Descriptions 149
2-38. Memory Self-Test Global Control Register (MSTGCR) Field Descriptions 150
2-39. Memory Hardware Initialization Global Control Register (MINITGCR) Field Descriptions 151
2-40. MBIST Controller/Memory Initialization Enable Register (MSINENA) Field Descriptions 152
2-41. Memory Self-Test Fail Status Register (MSTFAIL) Field Descriptions 153
2-42. MSTC Global Status Register (MSTCGSTAT) Field Descriptions 154
2-43. Memory Hardware Initialization Status Register (MINISTAT) Field Descriptions 155
2-44. PLL Control Register 1 (PLLCTL1) Field Descriptions 156
2-45. PLL Control Register 2 (PLLCTL2) Field Descriptions 157
2-46. SYS Pin Control Register 10 (SYSPC10) Field Descriptions 158
2-47. Die Identification Register, Lower Word (DIEIDL) Field Descriptions 159

2-48. Die Identification Register, Upper Word (DIEIDH) Field Descriptions 159
2-49. LPO/Clock Monitor Control Register (LPOMONCTL) Field Descriptions..... 160
2-50. Clock Test Register (CLKTEST) Field Descriptions..... 163
2-51. DFT Control Register (DFTCTRLREG) Field Descriptions..... 165
2-52. DFT Control Register 2 (DFTCTRLREG2) Field Descriptions 166
2-53. General Purpose Register (GPREG1) Field Descriptions 167
2-54. Imprecise Fault Status Register (IMPFASTS) Field Descriptions 169
2-55. Imprecise Fault Write Address Register (IMPFTADD) Field Descriptions 170
2-56. System Software Interrupt Request 1 Register (SSIR1) Field Descriptions 171
2-57. System Software Interrupt Request 2 Register (SSIR2) Field Descriptions 171
2-58. System Software Interrupt Request 3 Register (SSIR3) Field Descriptions 172
2-59. System Software Interrupt Request 4 Register (SSIR4) Field Descriptions 172
2-60. RAM Control Register (RAMGCR) Field Descriptions 173
2-61. Bus Matrix Module Control Register 1 (BMMCR) Field Descriptions 174
2-62. CPU Reset Control Register (CPURSTGCR) Field Descriptions 175
2-63. Clock Control Register (CLKCNTRL) Field Descriptions 176
2-64. ECP Control Register (ECPCNTL) Field Descriptions..... 177
2-65. DEV Parity Control Register 1 (DEVCR1) Field Descriptions 178
2-66. System Exception Control Register (SYSECR) Field Descriptions 178
2-67. System Exception Status Register (SYSESR) Field Descriptions 179
2-68. System Test Abort Status Register (SYSTASR) Field Descriptions 180
2-69. Global Status Register (GLBSTAT) Field Descriptions 181
2-70. Device Identification Register (DEVID) Field Descriptions 182
2-71. Software Interrupt Vector Register (SSIVEC) Field Descriptions 183
2-72. System Software Interrupt Flag Register (SSIF) Field Descriptions 184
2-73. Secondary System Control Registers 185
2-74. PLL Control Register 3 (PLLCTL3) Field Descriptions 186
2-75. CPU Logic BIST Clock Prescaler (STCLKDIV) Field Descriptions..... 187
2-76. Clock 2 Control Register (CLK2CNTRL) Field Descriptions 187
2-77. Peripheral Asynchronous Clock Configuration 1 Register (VCLKACON1) Field Descriptions 188
2-78. Clock Slip Register (CLKSLIP) Field Descriptions 190
2-79. EFUSE Controller Control Register (EFC_CTLREG) Field Descriptions..... 191
2-80. Die Identification Register, Lower Word (DIEIDL_REG0) Field Descriptions..... 191
2-81. Die Identification Register, Upper Word (DIEIDH_REG1) Field Descriptions 192
2-82. Die Identification Register, Lower Word (DIEIDL_REG2) Field Descriptions..... 192
2-83. Die Identification Register, Upper Word (DIEIDH_REG3) Field Descriptions 193
2-84. Peripheral Central Resource Control Registers 194
2-85. Peripheral Memory Protection Set Register 0 (PMPROTSET0) Field Descriptions 195
2-86. Peripheral Memory Protection Set Register 1 (PMPROTSET1) Field Descriptions 195
2-87. Peripheral Memory Protection Clear Register 0 (PMPROTCLR0) Field Descriptions..... 196
2-88. Peripheral Memory Protection Clear Register 1 (PMPROTCLR1) Field Descriptions..... 196
2-89. Peripheral Protection Set Register 0 (PPROTSET0) Field Descriptions 197
2-90. Peripheral Protection Set Register 1 (PPROTSET1) Field Descriptions 198
2-91. Peripheral Protection Set Register 2 (PPROTSET2) Field Descriptions 198
2-92. Peripheral Protection Set Register 3 (PPROTSET3) Field Descriptions 199
2-93. Peripheral Protection Clear Register 0 (PPROTCLR0) Field Descriptions 199
2-94. Peripheral Protection Clear Register 1 (PPROTCLR1) Field Descriptions 200
2-95. Peripheral Protection Clear Register 2 (PPROTCLR2) Field Descriptions 200
2-96. Peripheral Protection Clear Register 3 (PPROTCLR3) Field Descriptions 201

2-97. Peripheral Memory Power-Down Set Register 0 (PCSPWRDWNSET0) Field Descriptions 201
2-98. Peripheral Memory Power-Down Set Register 1 (PCSPWRDWNSET1) Field Descriptions 202
2-99. Peripheral Memory Power-Down Clear Register 0 (PCSPWRDWNCLR0) Field Descriptions................ 202
2-100. Peripheral Memory Power-Down Set Register 1 (PCSPWRDWNCLR1) Field Descriptions 203
2-101. Peripheral Power-Down Set Register 0 (PSPWRDWNSET0) Field Descriptions............................... 204
2-102. Peripheral Power-Down Set Register 1 (PSPWRDWNSET1) Field Descriptions............................... 205
2-103. Peripheral Power-Down Set Register 2 (PSPWRDWNSET2) Field Descriptions............................... 205
2-104. Peripheral Power-Down Set Register 3 (PSPWRDWNSET3) Field Descriptions............................... 206
2-105. Peripheral Power-Down Clear Register 0 (PSPWRDWNCLR0) Field Descriptions 206
2-106. Peripheral Power-Down Clear Register 1 (PSPWRDWNCLR1) Field Descriptions 207
2-107. Peripheral Power-Down Clear Register 2 (PSPWRDWNCLR2) Field Descriptions 207
2-108. Peripheral Power-Down Clear Register 3 (PSPWRDWNCLR3) Field Descriptions 208
3-1. PMM Registers 216
3-2. LOGICPDPWRCTRL0 Register Field Descriptions 217
3-3. MEMPDPWRCTRL0 Register Field Descriptions 219
3-4. PDCLKDIS Register Field Descriptions.. 220
3-5. PDCLKDISSET Register Field Descriptions 221
3-6. PDCLKDISCLR Register Field Descriptions 222
3-7. LOGICPDPWRSTAT0 Register Field Descriptions 223
3-8. LOGICPDPWRSTAT1 Register Field Descriptions 224
3-9. LOGICPDPWRSTAT2 Register Field Descriptions 225
3-10. LOGICPDPWRSTAT3 Register Field Descriptions 226
3-11. MEMPDPWRSTAT0 Register Field Descriptions 227
3-12. MEMPDPWRSTAT1 Register Field Descriptions 229
3-13. GLOBALCTRL1 Register Field Descriptions.. 231
3-14. GLOBALSTAT Register Field Descriptions 232
3-15. PRCKEYREG Register Field Descriptions 233
3-16. LPDDCSTAT1 Register Field Descriptions.. 234
3-17. LPDDCSTAT2 Register Field Descriptions.. 235
3-18. MPDDCSTAT1 Register Field Descriptions.. 236
3-19. MPDDCSTAT2 Register Field Descriptions.. 237
3-20. ISODIAGSTAT Register Field Descriptions.. 238
4-1. Input Multiplexing on 144QFP Parts 241
4-2. ADC1 Trigger Event Selection 243
4-3. ADC2 Trigger Event Selection 243
4-4. Controls for ePWMx Inputs... 249
4-5. Controls for eQEPx_ERROR Connection to ePWMx nTZ4 Inputs... 249
4-6. Controls for eCAPx Inputs 250
4-7. Controls for eQEPx Inputs... 250
4-8. IOMM Registers... 252
4-9. REVISION_REG Register Field Descriptions 253
4-10. ENDIAN_REG Register Field Descriptions.. 254
4-11. KICK_REG0 Register Field Descriptions 255
4-12. KICK_REG1 Register Field Descriptions 256
4-13. ERR_RAW_STATUS_REG Register Field Descriptions 257
4-14. ERR_ENABLED_STATUS_REG Register Field Descriptions.. 258
4-15. ERR_ENABLE_REG Register Field Descriptions.. 259
4-16. ERR_ENABLE_CLR_REG Register Field Descriptions 260
4-17. FAULT_ADDRESS_REG Register Field Descriptions... 261

4-18. FAULT_STATUS_REG Register Field Descriptions 262
4-19. FAULT_CLEAR_REG Register Field Descriptions 263
4-20. PINMMR_0 to PINMMR_47 Register Field Descriptions 264
4-21. Output Multiplexing and Control 265
5-1. ECC Encoding for LE Devices 271
5-2. Syndrome Table, Decode to Bit in Error 272
5-3. Alternate Syndrome Table 273
5-4. TI OTP Bank 0 Sector Information Field Descriptions 275
5-5. TI OTP Sector Information Address 275
5-6. TI OTP Bank 0 Package and Memory Size Information Field Descriptions 276
5-7. TI OTP Bank 0 LPO Trim and Max HCLK Information Field Descriptions 276
5-8. DIAG_MODE Encoding 278
5-9. Bus 1 Diagnostic Mode Summary 283
5-10. Bus 2 and ECC Diagnostic Mode Summary 283
5-11. Port Signals Diagnostic Mode Summary 284
5-12. Flash Control Registers 285
5-13. Flash Option Control Register (FRDCNTL) Field Descriptions 286
5-14. Flash Error Detection and Correction Control Register 1 (FEDACCTRL1) Field Descriptions 287
5-15. Flash Error Correction Control and Correction Register 2 (FEDACCTRL2) Field Descriptions 289
5-16. Flash Correctable Error Count Register (FCOR_ERR_CNT) Field Descriptions 289
5-17. Flash Correctable Error Address Register (FCOR_ERR_ADD) Field Descriptions 290
5-18. Flash Correctable Error Position Register (FCOR_ERR_POS) Field Descriptions 291
5-19. Flash Error Detection and Correction Status Register (FEDACSTATUS) Field Descriptions 292
5-20. Flash Uncorrectable Error Address Register (FUNC_ERR_ADD) Field Descriptions 295
5-21. Flash Error Detection and Correction Sector Disable Register (FEDACSDIS) Field Descriptions 296
5-22. Primary Address Tag Register (FPRIM_ADD_TAG) Field Descriptions 297
5-23. Duplicate Address Tag Register (FDUP_ADD_TAG) Field Descriptions 297
5-24. Flash Bank Protection Register (FBPROT) Field Descriptions 298
5-25. Flash Bank Sector Enable Register (FBSE) Field Descriptions 298
5-26. Flash Bank Busy Register (FBBUSY) Field Descriptions 299
5-27. Flash Bank Access Control Register (FBAC) Field Descriptions 300
5-28. Flash Bank Fallback Power Register (FBFALLBACK) Field Descriptions 301
5-29. Flash Pump Access Control Register 1 (FPAC1) Field Descriptions 302
5-30. Flash Pump Access Control Register 1 (FPAC1) Field Descriptions 303
5-31. Flash Pump Access Control Register 2 (FPAC2) Field Descriptions 304
5-32. Flash Module Access Control Register (FMAC) Field Descriptions 304
5-33. Flash Module Status Register (FMSTAT) Field Descriptions 305
5-34. EEPROM Emulation Data MSW Register (FEMU_DMSW) Field Descriptions 307
5-35. EEPROM Emulation Data LSW Register (FEMU_DLSW) Field Descriptions 307
5-36. EEPROM Emulation ECC Register (FEMU_ECC) Field Descriptions 308
5-37. EEPROM Emulation Address Register (FEMU_ADDR) Field Descriptions 309
5-38. Diagnostic Control Register (FDIAGCTRL) Field Descriptions 310
5-39. Uncorrected Raw Data High Register (FRAW_DATAH) Field Descriptions 312
5-40. Uncorrected Raw Data Low Register (FRAW_DATAL) Field Descriptions 312
5-41. Uncorrected Raw ECC Register (FRAW_ECC) Field Descriptions 313
5-42. Parity Override Register (FPAR_OVR) Field Descriptions 314
5-43. Flash Error Detection and Correction Sector Disable Register (FEDACSDIS2) Field Descriptions 315
5-44. FSM Register Write Enable (FSM_WR_ENA) Field Descriptions 316
5-45. FSM Sector Register (FSM_SECTOR) Field Descriptions 316

5-46. EPROM Emulation Configuration Register (EEPROM_CONFIG) Field Descriptions 317
5-47. EEPROM Emulation Error Detection and Correction Control Register 1 (EE_CTRL1) Field Descriptions ... 318
5-48. EEPROM Emulation Error Correction Control Register 2 (EE_CTRL2) Field Descriptions 320
5-49. EEPROM Emulation Correctable Error Count Register (EE_COR_ERR_CNT) Field Descriptions 320
5-50. EEPROM Emulation Correctable Error Address Register (EE_COR_ERR_ADD) Field Descriptions 321
5-51. EEPROM Emulation Correctable Error Position Register (EE_COR_ERR_POS) Field Descriptions......... 322
5-52. EEPROM Emulation Error Status Register (EE_STATUS) Field Descriptions 323
5-53. EEPROM Emulation Uncorrectable Error Address Register (EE_UNC_ERR_ADD) Field Descriptions...... 324
5-54. Flash Bank Configuration Register (FCFG_BANK) Field Descriptions 325
6-1. TCRAM_MODULE_CONTROL_AND_STATUS Registers 332
6-2. TCRAM_MODULE_CONTROL_AND_STATUS Access Type Codes 332
6-3. RAMCTRL Register Field Descriptions 333
6-4. RAMTHRESHOLD Register Field Descriptions 335
6-5. RAMOCCUR Register Field Descriptions 336
6-6. RAMINTCTRL Register Field Descriptions 337
6-7. RAMERRSTATUS Register Field Descriptions 338
6-8. RAMSERRADDR Register Field Descriptions 340
6-9. RAMUERRADDR Register Field Descriptions 341
6-10. RAMTEST Register Field Descriptions 342
6-11. RAMADDRDECVECT Register Field Descriptions 344
6-12. RAMPERRADDR Register Field Descriptions 345
6-13. INIT_DOMAIN Register Field Descriptions 346
7-1. PBIST Registers 354
7-2. RAM Configuration Register (RAMT) Field Descriptions 355
7-3. Datalogger Register (DLR) Field Descriptions 356
7-4. PBIST Activate/ROM Clock Enable Register (PACT) Field Descriptions 357
7-5. PBIST ID Register Field Descriptions 358
7-6. Override Register (OVER) Field Descriptions.. 359
7-7. Fail Status Fail Register 0 (FSRF0) Field Descriptions... 360
7-8. Fail Status Count 0 Register (FSRC0) Field Descriptions.. 361
7-9. Fail Status Count Register 1 (FSRC1) Field Descriptions.. 361
7-10. Fail Status Address 0 Register (FSRA0) Field Descriptions 362
7-11. Fail Status Address 1 Register (FSRA1) Field Descriptions 362
7-12. Fail Status Data Register 0 (FSRDL0) Field Descriptions.. 363
7-13. Fail Status Data Register 1 (FSRDL1) Field Descriptions.. 363
7-14. ROM Mask Register (ROM) Field Descriptions.. 364
7-15. Algorithm Mask Register (ALGO) Field Descriptions 364
7-16. RAM Info Mask Lower Register (RINFOL) Field Descriptions 365
7-17. RAM Info Mask Upper Register (RINFOU) Field Descriptions 366
8-1. STC Test Coverage and Duration 374
8-2. Typical STC Execution Times.. 374
8-3. STC Control Registers 375
8-4. STC Global Control Register 0 (STCGCR0) Field Descriptions.. 376
8-5. STC Global Control Register 1 (STCGCR1) Field Descriptions.. 376
8-6. Self-Test Run Timeout Counter Preload Register (STCTPR) 377
8-7. STC Current ROM Address Register (STC_CADDR) Field Descriptions 377
8-8. STC Current Interval Count Register (STCCICR) Field Descriptions... 378
8-9. Self-Test Global Status Register (STCGSTAT) Field Descriptions 379
8-10. Self-Test Fail Status Register (STCFSTAT) Field Descriptions 380

8-11. CPU1 Current MISR Register (CPU1_CURMISR[3:0]) Field Descriptions 381
8-12. CPU2 Current MISR Register (CPU2_CURMISR[3:0]) Field Descriptions 382
8-13. Signature Compare Self-Check Regsiter (STCSCSCR) Field Descriptions 383
9-1. Compare Match Test Sequence 388
9-2. Compare Mismatch Test Sequence 389
9-3. CCM-R4F Control Registers 390
9-4. CCM-R4F Status Register (CCMSR) Field Descriptions 391
9-5. CCM-R4F Key Register (CCMKEYR) Field Descriptions 392
10-1. Valid Frequency Ranges for PLL 401
10-2. PLL Value Encoding 402
10-3. Summary of PLL Timings 406
10-4. PLL Module Registers 411
10-5. LPOCLKDET Module Registers 411
10-6. SSW PLL BIST Control Register 1 (SSWPLL1) Field Descriptions 412
10-7. SSW PLL BIST Control Register 2 (SSWPLL2) Field Descriptions 413
10-8. SSW PLL BIST Control Register 3 (SSWPLL3) Field Descriptions 414
11-1. DCC Control Registers 426
11-2. DCC Global Control Register (DCCGCTRL) Field Descriptions 427
11-3. DCC Revision Id Register (DCCREV) Field Descriptions 428
11-4. DCC Counter0 Seed Register (DCCCNT0SEED) Field Descriptions 428
11-5. DCC Valid0 Seed Register (DCCVALID0SEED) Field Descriptions 429
11-6. DCC Counter1 Seed Register (DCCCNT0SEED) Field Descriptions 429
11-7. DCC Status Register (DCCSTAT) Field Descriptions 430
11-8. DCC Counter0 Value Register (DCCCNT0) Field Descriptions 431
11-9. DCC Valid0 Value Register (DCCVALID0) Field Descriptions 432
11-10. DCC Counter1 Value Register (DCCCNT1) Field Descriptions 432
11-11. DCC Counter1 Clock Source Selection Register (DCCCNT1CLKSRC) Field Descriptions 433
11-12. DCC Counter0 Clock Source Selection Register (DCCCNT0CLKSRC) Field Descriptions 434
12-1. ESM Interrupt and $\overline{\text{ERROR}}$ Pin Behavior 437
12-2. ESM Module Registers 443
12-3. ESM Enable $\overline{\text{ERROR}}$ Pin Action/Response Register 1 (ESMEEPAPR1) Field Descriptions 444
12-4. ESM Disable $\overline{\text{ERROR}}$ Pin Action/Response Register 1 (ESMDEPAPR1) Field Descriptions 444
12-5. ESM Interrupt Enable Set Register 1 (ESMIESR1) Field Descriptions 445
12-6. ESM Interrupt Enable Clear Register 1 (ESMIECR1) Field Descriptions 445
12-7. ESM Interrupt Level Set Register 1 (ESMILSR1) Field Descriptions 446
12-8. ESM Interrupt Level Clear Register 1 (ESMILCR1) Field Descriptions 446
12-9. ESM Status Register 1 (ESMSR1) Field Descriptions 447
12-10. ESM Status Register 2 (ESMSR2) Field Descriptions 447
12-11. ESM Status Register 3 (ESMSR3) Field Descriptions 448
12-12. ESM $\overline{\text{ERROR}}$ Pin Status Register (ESMEPSR) Field Descriptions 448
12-13. ESM Interrupt Offset High Register (ESMIOFFHR) Field Descriptions 449
12-14. ESM Interrupt Offset Low Register (ESMIOFFLR) Field Descriptions 450
12-15. ESM Low-Time Counter Register (ESMLTCR) Field Descriptions 451
12-16. ESM Low-Time Counter Preload Register (ESMLTCPR) Field Descriptions 451
12-17. ESM Error Key Register (ESMEKR) Field Descriptions 452
12-18. ESM Status Shadow Register 2 (ESMSSR2) Field Descriptions 452
12-19. ESM Influence $\overline{\text{ERROR}}$ Pin Set Register 4 (ESMIEPSR4) Field Descriptions 453
12-20. ESM Influence $\overline{\text{ERROR}}$ Pin Clear Register 4 (ESMIEPCR4) Field Descriptions 453
12-21. ESM Interrupt Enable Set Register 4 (ESMIESR4) Field Descriptions 454

12-22. ESM Interrupt Enable Clear Register 4 (ESMIECR4) Field Descriptions 454
12-23. ESM Interrupt Level Set Register 4 (ESMILSR4) Field Descriptions 455
12-24. ESM Interrupt Level Clear Register 4 (ESMILCR4) Field Descriptions 455
12-25. ESM Status Register 4 (ESMSR4) Field Descriptions 456
13-1. RTI Registers 469
13-2. RTI Global Control Register (RTIGCTRL) Field Descriptions 470
13-3. RTI Timebase Control Register (RTITBCTRL) Field Descriptions 471
13-4. RTI Capture Control Register (RTICAPCTRL) Field Descriptions 472
13-5. RTI Compare Control Register (RTICOMPCTRL) Field Descriptions 473
13-6. RTI Free Running Counter 0 Register (RTIFRC0) Field Descriptions 474
13-7. RTI Up Counter 0 Register (RTIUC0) Field Descriptions 474
13-8. RTI Compare Up Counter 0 Register (RTICPUC0) Field Descriptions 475
13-9. RTI Capture Free Running Counter 0 Register (RTICAFRC0) Field Descriptions 475
13-10. RTI Capture Up Counter 0 Register (RTICAUC0) Field Descriptions 476
13-11. RTI Free Running Counter 1 Register (RTIFRC1) Field Descriptions 476
13-12. RTI Up Counter 1 Register (RTIUC1) Field Descriptions 477
13-13. RTI Compare Up Counter 1 Register (RTICPUC1) Field Descriptions 478
13-14. RTI Capture Free Running Counter 1 Register (RTICAFRC1) Field Descriptions 479
13-15. RTI Capture Up Counter 1 Register (RTICAUC1) Field Descriptions 479
13-16. RTI Compare 0 Register (RTICOMP0) Field Descriptions 480
13-17. RTI Update Compare 0 Register (RTIUDCP0) Field Descriptions 480
13-18. RTI Compare 1 Register (RTICOMP1) Field Descriptions 481
13-19. RTI Update Compare 1 Register (RTIUDCP1) Field Descriptions 481
13-20. RTI Compare 2 Register (RTICOMP2) Field Descriptions 482
13-21. RTI Update Compare 2 Register (RTIUDCP2) Field Descriptions 482
13-22. RTI Compare 3 Register (RTICOMP3) Field Descriptions 483
13-23. RTI Update Compare 3 Register (RTIUDCP3) Field Descriptions 483
13-24. RTI Timebase Low Compare Register (RTITBLCOMP) Field Descriptions 484
13-25. RTI Timebase High Compare Register (RTITBHCOMP) Field Descriptions 484
13-26. RTI Set Interrupt Control Register (RTISETINTENA) Field Descriptions 485
13-27. RTI Clear Interrupt Control Register (RTICLEARINTENA) Field Descriptions 487
13-28. RTI Interrupt Flag Register (RTIINTFLAG) Field Descriptions 489
13-29. Digital Watchdog Control Register (RTIDWDCTRL) Field Descriptions 490
13-30. Digital Watchdog Preload Register (RTIDWDPRLD) Field Descriptions 491
13-31. Watchdog Status Register (RTIWDSTATUS) Field Descriptions 492
13-32. RTI Watchdog Key Register (RTIDWDKEY) Field Descriptions 493
13-33. Example of a WDKEY Sequence 493
13-34. RTI Watchdog Down Counter Register (RTIDWDCNTR) Field Descriptions 494
13-35. Digital Windowed Watchdog Reaction Control (RTIWWDRXNCTRL) Field Descriptions 494
13-36. Digital Windowed Watchdog Window Size Control (RTIWWDSIZECTRL) Field Descriptions 495
13-37. RTI Compare Interrupt Clear Enable Register (RTIINTCLRENABLE) Field Descriptions 496
13-38. RTI Compare 0 Clear Register (RTICMP0CLR) Field Descriptions 497
13-39. RTI Compare 1 Clear Register (RTICMP1CLR) Field Descriptions 497
13-40. RTI Compare 2 Clear Register (RTICMP2CLR) Field Descriptions 498
13-41. RTI Compare 3 Clear Register (RTICMP3CLR) Field Descriptions 498
14-1. CRC Modes in Which DMA Request and Counter Logic are Active or Inactive 507
14-2. Modes in Which Interrupt Condition Can Occur 508
14-3. Interrupt Offset Mapping 511
14-4. CRC Control Registers 516

14-5. CRC Global Control Register 0 (CRC_CTRL0) Field Descriptions 517
14-6. CRC Global Control Register 1 (CRC_CTRL1) Field Descriptions 517
14-7. CRC Global Control Register 2 (CRC_CTRL2) Field Descriptions 518
14-8. CRC Interrupt Enable Set Register (CRC_INTS) Field Descriptions 519
14-9. CRC Interrupt Enable Reset Register (CRC_INTR) Field Descriptions 521
14-10. CRC Interrupt Status Register (CRC_STATUS) Field Descriptions 523
14-11. CRC Interrupt Offset (CRC_INT_OFFSET_REG) Field Descriptions 525
14-12. CRC Busy Register (CRC_BUSY) Field Descriptions 526
14-13. CRC Pattern Counter Preload Register 1 (CRC_PCOUNT_REG1) Field Descriptions..... 526
14-14. CRC Sector Counter Preload Register 1 (CRC_SCOUNT_REG1) Field Descriptions..... 527
14-15. CRC Current Sector Register 1 (CRC_CURSEC_REG1) Field Descriptions 527
14-16. CRC Channel 1 Watchdog Timeout Preload Register A (CRC_WDTOPLD1) Field Descriptions 528
14-17. CRC Channel 1 Block Complete Timeout Preload Register B (CRC_BCTOPLD1) Field Descriptions 528
14-18. Channel 1 PSA Signature Low Register (PSA_SIGREGL1) Field Descriptions 529
14-19. Channel 1 PSA Signature High Register (PSA_SIGREGH1) Field Descriptions 529
14-20. Channel 1 CRC Value Low Register (CRC_REGL1) Field Descriptions 529
14-21. Channel 1 CRC Value High Register (CRC_REGH1) Field Descriptions 530
14-22. Channel 1 PSA Sector Signature Low Register (PSA_SECSIGREGL1) Field Descriptions 530
14-23. Channel 1 PSA Sector Signature High Register (PSA_SECSIGREGH1) Field Descriptions 530
14-24. Channel 1 Raw Data Low Register (RAW_DATAREGL1) Field Descriptions 531
14-25. Channel 1 Raw Data High Register (RAW_DATAREGH1) Field Descriptions 531
14-26. CRC Pattern Counter Preload Register 2 (CRC_PCOUNT_REG2) Field Descriptions..... 531
14-27. CRC Sector Counter Preload Register 2 (CRC_SCOUNT_REG2) Field Descriptions..... 532
14-28. CRC Current Sector Register 2 (CRC_CURSEC_REG2) Field Descriptions 532
14-29. CRC Channel 2 Watchdog Timeout Preload Register A (CRC_WDTOPLD2) Field Descriptions 533
14-30. CRC Channel 2 Block Complete Timeout Preload Register B (CRC_BCTOPLD2) Field Descriptions 533
14-31. Channel 2 PSA Signature Low Register (PSA_SIGREGL2) Field Descriptions 534
14-32. Channel 2 PSA Signature High Register (PSA_SIGREGH2) Field Descriptions 534
14-33. Channel 2 CRC Value Low Register (CRC_REGL2) Field Descriptions 534
14-34. Channel 2 CRC Value High Register (CRC_REGH2) Field Descriptions 535
14-35. Channel 2 PSA Sector Signature Low Register (PSA_SECSIGREGL2) Field Descriptions 535
14-36. Channel 2 PSA Sector Signature High Register (PSA_SECSIGREGH2) Field Descriptions 535
14-37. Channel 2 Raw Data Low Register (RAW_DATAREGL2) Field Descriptions 536
14-38. Channel 2 Raw Data High Register (RAW_DATAREGH2) Field Descriptions 536
14-39. Data Bus Selection Register Field Descriptions 537
15-1. VIM Control Registers 553
15-2. Interrupt Vector Table Parity Flag Register (PARFLG) Field Descriptions 554
15-3. Interrupt Vector Table Parity Control Register (PARCTL) Field Descriptions 554
15-4. Address Parity Error Register (PARERR) Field Descriptions 555
15-5. Fall Back Address Parity Error Register (FBPARERR) Field Descriptions 555
15-6. Interrupt Dispatch 556
15-7. IRQ Index Offset Vector Register (IRQINDEX) Field Descriptions 557
15-8. FIQ Index Offset Vector Register (FIQINDEX) Field Descriptions 557
15-9. FIQ/IRQ Program Control Registers (FIRQPR) Field Descriptions 559
15-10. Pending Interrupt Read Location Registers (INTREQ) Field Descriptions 561
15-11. Interrupt Enable Set Registers (REQENASET) Field Descriptions 563
15-12. Interrupt Enable Clear Registers (REQENACLR) Field Descriptions 565
15-13. Wake-Up Enable Set Registers (WAKEENASET) Field Descriptions 567
15-14. Wake-Up Enable Clear Registers (WAKEENACLR) Field Descriptions 569

15-15. IRQ Interrupt Vector Register (IRQVECREG) Field Descriptions 570
15-16. FIQ Interrupt Vector Register (FIQVECREG) Field Descriptions 570
15-17. Capture Event Register (CAPEVT) Field Descriptions 571
15-18. Interrupt Control Registers Organization 572
15-19. Interrupt Control Registers (CHANCTL) Field Descriptions 572
16-1. Arbitration According to Priority Queues and Priority Schemes 582
16-2. DMA Request Line Connection 586
16-3. Maximum Number of DMA Transactions per Channel in Non-Bypass Mode 591
16-4. Maximum Number of DMA Transactions per Channel in Bypass Mode 591
16-5. Control Packet RAM 594
16-6. Control Packet RAM 594
16-7. Parity RAM 594
16-8. DMA Control Registers 595
16-9. Control Packet Memory Map 596
16-10. Global Control Register (GCTRL) Field Descriptions 597
16-11. Channel Pending Register (PEND) Field Descriptions 598
16-12. DMA Status Register (DMASTAT) Field Descriptions 598
16-13. HW Channel Enable Set and Status Register (HWCHENAS) Field Descriptions 599
16-14. HW Channel Enable Reset and Status Register (HWCHENAR) Field Descriptions 599
16-15. SW Channel Enable Set and Status Register (SWCHENAS) Field Descriptions 600
16-16. SW Channel Enable Reset and Status Register (SWCHENAR) Field Descriptions 601
16-17. Channel Priority Set Register (CHPRIOS) Field Descriptions 601
16-18. Channel Priority Reset Register (CHPRIOR) Field Descriptions 602
16-19. Global Channel Interrupt Enable Set Register (GCHIENAS) Field Descriptions 602
16-20. Global Channel Interrupt Enable Reset Register (GCHIENAR) Field Descriptions 603
16-21. DMA Request Assignment Register 0 (DREQASI0) Field Descriptions 604
16-22. DMA Request Assignment Register 1 (DREQASI1) Field Descriptions 605
16-23. DMA Request Assignment Register 2 (DREQASI2) Field Descriptions 606
16-24. DMA Request Assignment Register 3 (DREQASI3) Field Descriptions 607
16-25. Port Assignment Register 0 (PAR0) Field Descriptions 608
16-26. Port Assignment Register 1 (PAR1) Field Descriptions 609
16-27. FTC Interrupt Mapping Register (FTCMAP) Field Descriptions 610
16-28. LFS Interrupt Mapping Register (LFSMAP) Field Descriptions 610
16-29. HBC Interrupt Mapping Register (HBCMAP) Field Descriptions 611
16-30. BTC Interrupt Mapping Register (BTCMAP) Field Descriptions 611
16-31. FTC Interrupt Enable Set (FTCINTENAS) Field Descriptions 612
16-32. FTC Interrupt Enable Reset (FTCINTENAR) Field Descriptions 612
16-33. LFS Interrupt Enable Set (LFSINTENAS) Field Descriptions 613
16-34. LFS Interrupt Enable Reset (LFSINTENAR) Field Descriptions 613
16-35. HBC Interrupt Enable Set (HBCINTENAS) Field Descriptions 614
16-36. HBC Interrupt Enable Reset (HBCINTENAR) Field Descriptions 614
16-37. BTC Interrupt Enable Reset (BTCINTENAS) Field Descriptions 615
16-38. BTC Interrupt Enable Reset (BTCINTENAR) Field Descriptions 615
16-39. Global Interrupt Flag Register (GINTFLAG) Field Descriptions 616
16-40. FTC Interrupt Flag Register (FTCFLAG) Field Descriptions 616
16-41. LFS Interrupt Flag Register (LFSFLAG) Field Descriptions 617
16-42. HBC Interrupt Flag (HBCFLAG) Field Descriptions 617
16-43. BTC Interrupt Flag Register (BTCFLAG) Field Descriptions 618
16-44. FTCA Interrupt Channel Offset Register (FTCAOFFSET) Field Descriptions 619

16-45. LFSA Interrupt Channel Offset Register (LFSAOFFSET) Field Descriptions 619
16-46. HBCA Interrupt Channel Offset Register (HBCAOFFSET) Field Descriptions 621
16-47. BTCA Interrupt Channel Offset Register (BTCAOFFSET) Field Descriptions 621
16-48. FTCB Interrupt Channel Offset Register (FTCBOFFSET) Field Descriptions 623
16-49. LFSB Interrupt Channel Offset Register (LFSBOFFSET) Field Descriptions 623
16-50. HBCB Interrupt Channel Offset Register (HBCBOFFSET) Field Descriptions 625
16-51. BTCB Interrupt Channel Offset Register (BTCBOFFSET) Field Descriptions 625
16-52. Port Control Register (PTCRL) Field Descriptions 627
16-53. RAM Test Control (RTCTRL) Field Descriptions 628
16-54. Debug Control (DCTRL) Field Descriptions 629
16-55. Watch Point Register (WPR) Field Descriptions 630
16-56. Watch Mask Register (WMR) Field Descriptions 630
16-57. Port B Active Channel Source Address Register (PBACDADDR) Field Descriptions 631
16-58. Port B Active Channel Destination Address Register (PBACDADDR) Field Descriptions 631
16-59. Port B Active Channel Transfer Count Register (PBACTC) Field Descriptions 632
16-60. Parity Control Register (DMAPCR) Field Descriptions 633
16-61. DMA Parity Error Address Register (DMAPAR) Field Descriptions 634
16-62. DMA Memory Protection Control Register (DMAMPCTRL) Field Descriptions 635
16-63. DMA Memory Protection Status Register (DMAMPST) Field Descriptions 637
16-64. DMA Memory Protection Region 0 Start Address Register (DMAMPR0S) Field Descriptions 638
16-65. DMA Memory Protection Region 0 End Address Register (DMAMPR0E) Field Descriptions 638
16-66. DMA Memory Protection Region 1 Start Address Register (DMAMPR1S) Field Descriptions 639
16-67. DMA Memory Protection Region 1 End Address Register (DMAMPR1E) Field Descriptions 639
16-68. DMA Memory Protection Region 2 Start Address Register (DMAMPR2S) Field Descriptions 640
16-69. DMA Memory Protection Region 2 End Address Register (DMAMPR2E) Field Descriptions 640
16-70. DMA Memory Protection Region 3 Start Address Register (DMAMPR3S) Field Descriptions 641
16-71. DMA Memory Protection Region 3 End Address Register (DMAMPR3E) Field Descriptions 641
16-72. Initial Source Address (ISADDR) Field Descriptions 642
16-73. Initial Destination Address Register (IDADDR) Field Descriptions 642
16-74. Initial Transfer Count Register (ITCOUNT) Field Descriptions 643
16-75. Channel Control Register (CHCTRL) Field Descriptions 644
16-76. Element Index Offset Register (EIOFF) Field Descriptions 645
16-77. Frame Index Offset Register (FIOFF) Field Descriptions 645
16-78. Current Source Address Register (CSADDR) Field Descriptions 646
16-79. Current Destination Address Register (CDADDR) Field Descriptions 646
16-80. Current Transfer Count Register (CTCOUNT) Field Descriptions 647
17-1. EMIF Pins Used to Access Both SDRAM and Asynchronous Memories 651
17-2. EMIF Pins Specific to SDRAM 652
17-3. EMIF Pins Specific to Asynchronous Memory 652
17-4. EMIF SDRAM Commands 653
17-5. Truth Table for SDRAM Commands 653
17-6. 16-bit EMIF Address Pin Connections 655
17-7. Description of the SDRAM Configuration Register (SDCR) 656
17-8. Description of the SDRAM Refresh Control Register (SDRCR) 656
17-9. Description of the SDRAM Timing Register (SDTIMR) 657
17-10. Description of the SDRAM Self Refresh Exit Timing Register (SDSRETR) 657
17-11. SDRAM LOAD MODE REGISTER Command 658
17-12. Refresh Urgency Levels 659
17-13. Mapping from Logical Address to EMIF Pins for 16-bit SDRAM 664

17-14. Normal Mode vs. Select Strobe Mode 665
17-15. Description of the Asynchronous *m* Configuration Register (CE*n*CFG) 667
17-16. Description of the Asynchronous Wait Cycle Configuration Register (AWCC) 668
17-17. Description of the EMIF Interrupt Mask Set Register (INTMSKSET) 668
17-18. Description of the EMIF Interrupt Mast Clear Register (INTMSKCLR) 668
17-19. Asynchronous Read Operation in Normal Mode 669
17-20. Asynchronous Write Operation in Normal Mode 671
17-21. Asynchronous Read Operation in Select Strobe Mode 673
17-22. Asynchronous Write Operation in Select Strobe Mode 675
17-23. Interrupt Monitor and Control Bit Fields 679
17-24. External Memory Interface (EMIF) Registers 683
17-25. Module ID Register (MIDR) Field Descriptions 683
17-26. Asynchronous Wait Cycle Configuration Register (AWCCR) Field Descriptions 684
17-27. SDRAM Configuration Register (SDCR) Field Descriptions 685
17-28. SDRAM Refresh Control Register (SDRCR) Field Descriptions 686
17-29. Asynchronous *n* Configuration Register (CE*n*CFG) Field Descriptions 687
17-30. SDRAM Timing Register (SDTIMR) Field Descriptions 688
17-31. SDRAM Self Refresh Exit Timing Register (SDSRETR) Field Descriptions 689
17-32. EMIF Interrupt Raw Register (INTRAW) Field Descriptions 690
17-33. EMIF Interrupt Mask Register (INTMSK) Field Descriptions 691
17-34. EMIF Interrupt Mask Set Register (INTMSKSET) Field Descriptions 692
17-35. EMIF Interrupt Mask Clear Register (INTMSKCLR) Field Descriptions 693
17-36. Page Mode Control Register (PMCR) Field Descriptions 694
17-37. SR Field Value For the EMIF to K4S641632H-TC(L)70 Interface 695
17-38. SDTIMR Field Calculations for the EMIF to K4S641632H-TC(L)70 Interface 697
17-39. RR Calculation for the EMIF to K4S641632H-TC(L)70 Interface 698
17-40. RR Calculation for the EMIF to K4S641632H-TC(L)70 Interface 699
17-41. SDCR Field Values For the EMIF to K4S641632H-TC(L)70 Interface 700
17-42. AC Characteristics for a Read Access 701
17-43. AC Characteristics for a Write Access 701
18-1. POM Registers 708
18-2. POM Global Control Register (POMGLBCTRL) Field Descriptions 709
18-3. POM Revision ID (POMREV) Field Descriptions 710
18-4. POM Clock Gate Control Register (POMCLKCTRL) Field Descriptions 710
18-5. POM Status Register (POMFLG) Field Descriptions 711
18-6. POM Program Region Start Address Register x (POMPROGSTARTx) Field Descriptions 712
18-7. POM Overlay Region Start Address Register x (POMOVLSTARTx) Field Descriptions 712
18-8. POM Region Size Register x (POMREGSIZEx) Field Descriptions 713
18-9. POM Integration Control Register (POMITCTRL) Field Descriptions 713
18-10. POM Claim Set Register (POMCLAIMSET) Field Descriptions 714
18-11. POM Claim Clear Register (POMCLAIMCLR) Field Descriptions 715
18-12. POM Lock Access Register (POMLOCKACCESS) Field Descriptions 716
18-13. POM Lock Status Register (POMLOCKSTATUS) Field Descriptions 716
18-14. POM Authentication Status Register (POMAUTHSTATUS) Field Descriptions 716
18-15. POM Device ID Register (POMDEVID) Field Descriptions 717
18-16. POM Device Type Register (POMDEVTYPE) Field Descriptions 717
18-17. POM Peripheral ID 4 Register (POMPERIPHERALID4) Field Descriptions 718
18-18. POM Peripheral ID 5 Register (POMPERIPHERALID5) Field Descriptions 718
18-19. POM Peripheral ID 6 Register (POMPERIPHERALID6) Field Descriptions 719

18-20. POM Peripheral ID 7 Register (POMPERIPHERALID7) Field Descriptions 719
18-21. POM Peripheral ID 0 Register (POMPERIPHERALID0) Field Descriptions 720
18-22. POM Peripheral ID 1 Register (POMPERIPHERALID1) Field Descriptions 720
18-23. POM Peripheral ID 2 Register (POMPERIPHERALID2) Field Descriptions 721
18-24. POM Peripheral ID 3 Register (POMPERIPHERALID3) Field Descriptions 721
18-25. POM Component ID 0 Register (POMCOMPONENTID0) Field Descriptions 722
18-26. POM Component ID 1 Register (POMCOMPONENTID1) Field Descriptions 722
18-27. POM Component ID 2 Register (POMCOMPONENTID2) Field Descriptions 723
18-28. POM Component ID 3 Register (POMCOMPONENTID3) Field Descriptions 723
19-1. ePWM Module Control and Status Register Set Grouped by Submodule 728
19-2. Submodule Configuration Parameters 729
19-3. Time-Base Submodule Registers...... 732
19-4. Key Time-Base Signals 733
19-5. Counter-Compare Submodule Registers 743
19-6. Counter-Compare Submodule Key Signals 743
19-7. Action-Qualifier Submodule Registers 748
19-8. Action-Qualifier Submodule Possible Input Events 749
19-9. Action-Qualifier Event Priority for Up-Down-Count Mode 751
19-10. Action-Qualifier Event Priority for Up-Count Mode 751
19-11. Action-Qualifier Event Priority for Down-Count Mode...... 751
19-12. Behavior if CMPA/CMPB is Greater than the Period 752
19-13. Dead-Band Generator Submodule Registers 762
19-14. Classical Dead-Band Operating Modes 764
19-15. Dead-Band Delay Values in μS as a Function of DBFED and DBRED 766
19-16. PWM-Chopper Submodule Registers...... 767
19-17. Possible Pulse Width Values for VCLK4 = 100 MHz 769
19-18. Trip-Zone Submodule Registers...... 772
19-19. Possible Actions On a Trip Event...... 773
19-20. Event-Trigger Submodule Registers 779
19-21. Digital Compare Submodule Registers 783
19-22. EPWM_MODULE_CONTROL_AND_STATUS_REGISTER Registers 803
19-23. EPWM_MODULE_CONTROL_AND_STATUS_REGISTER Access Type Codes...... 804
19-24. TBCTL Register Field Descriptions...... 805
19-25. TBSTS Register Field Descriptions 807
19-26. TBPHS Register Field Descriptions 808
19-27. TBCTR Register Field Descriptions 809
19-28. TBPRD Register Field Descriptions...... 810
19-29. CMPCTL Register Field Descriptions 811
19-30. CMPA Register Field Descriptions 813
19-31. CMPB Register Field Descriptions 814
19-32. AQCTLA Register Field Descriptions 815
19-33. AQCTLB Register Field Descriptions 817
19-34. AQSFRC Register Field Descriptions...... 819
19-35. AQCSFRC Register Field Descriptions...... 820
19-36. DBCTL Register Field Descriptions 821
19-37. DBRED Register Field Descriptions...... 823
19-38. DBFED Register Field Descriptions...... 824
19-39. TZSEL Register Field Descriptions...... 825
19-40. TZDCSEL Register Field Descriptions...... 827

19-41. TZCTL Register Field Descriptions 828
19-42. TZEINT Register Field Descriptions 829
19-43. TZFLG Register Field Descriptions 830
19-44. TZCLR Register Field Descriptions 831
19-45. TZFRC Register Field Descriptions 832
19-46. ETSEL Register Field Descriptions 833
19-47. ETPS Register Field Descriptions 835
19-48. ETFLG Register Field Descriptions 837
19-49. ETCLR Register Field Descriptions 838
19-50. ETFRC Register Field Descriptions 839
19-51. PCCTL Register Field Descriptions 840
19-52. DCTRIPSEL Register Field Descriptions 841
19-53. DCACTL Register Field Descriptions 842
19-54. DCBCTL Register Field Descriptions 843
19-55. DCFCTL Register Field Descriptions 844
19-56. DCCAPCTL Register Field Descriptions 845
19-57. DCFOFFSET Register Field Descriptions 846
19-58. DCFOFFSETCNT Register Field Descriptions 847
19-59. DCFWINDOW Register Field Descriptions 848
19-60. DCFWINDOWCNT Register Field Descriptions 849
19-61. DCCAP Register Field Descriptions 850
20-1. ECAP Registers 860
20-2. ECAP Access Type Codes 860
20-3. TSCTR Register Field Descriptions 861
20-4. CTRPHS Register Field Descriptions 862
20-5. CAP1 Register Field Descriptions 863
20-6. CAP2 Register Field Descriptions 864
20-7. CAP3 Register Field Descriptions 865
20-8. CAP4 Register Field Descriptions 866
20-9. ECCTL1 Register Field Descriptions 867
20-10. ECCTL2 Register Field Descriptions 869
20-11. ECEINT Register Field Descriptions 871
20-12. ECFLG Register Field Descriptions 872
20-13. ECCLR Register Field Descriptions 873
20-14. ECFRC Register Field Descriptions 874
21-1. EQEP Memory Map 890
21-2. Quadrature Decoder Truth Table 892
21-3. EQEP Registers 906
21-4. EQEP Access Type Codes 906
21-5. QPOSCNT Register Field Descriptions 907
21-6. QPOSINIT Register Field Descriptions 908
21-7. QPOSMAX Register Field Descriptions 909
21-8. QPOSCMP Register Field Descriptions 910
21-9. QPOSILAT Register Field Descriptions 911
21-10. QPOSSLAT Register Field Descriptions 912
21-11. QPOSLAT Register Field Descriptions 913
21-12. QUTMR Register Field Descriptions 914
21-13. QUPRD Register Field Descriptions 915
21-14. QWDTMR Register Field Descriptions 916

21-15. QWDPRD Register Field Descriptions 917
21-16. QDECCTL Register Field Descriptions 918
21-17. QEPCTL Register Field Descriptions 919
21-18. QCAPCTL Register Field Descriptions 921
21-19. QPOSCTL Register Field Descriptions 922
21-20. QEINT Register Field Descriptions 923
21-21. QFLG Register Field Descriptions 924
21-22. QCLR Register Field Descriptions 926
21-23. QFRC Register Field Descriptions 928
21-24. QEPSTS Register Field Descriptions 929
21-25. QCTMR Register Field Descriptions 930
21-26. QCPRD Register Field Descriptions 931
21-27. QCTMRLAT Register Field Descriptions 932
21-28. QCPRDLAT Register Field Descriptions 933
22-1. ADC Look-Up Table Field Descriptions 948
22-2. Calibration Reference Voltages 958
22-3. Self-Test Reference Voltages 961
22-4. Determination of ADC Input Channel Condition 962
22-5. Output Buffer and Pull Control Behavior for ADxEVT as GPIO Pins 966
22-6. ADC_CONTROL Registers 967
22-7. ADC_CONTROL Access Type Codes 969
22-8. ADRSTCR Register Field Descriptions 970
22-9. ADOPMODECR Register Field Descriptions 971
22-10. ADCLOCKCR Register Field Descriptions 973
22-11. ADCALCR Register Field Descriptions 974
22-12. ADEVMODECR Register Field Descriptions 976
22-13. ADG1MODECR Register Field Descriptions 979
22-14. ADG2MODECR Register Field Descriptions 982
22-15. ADEVSRC Register Field Descriptions 985
22-16. ADG1SRC Register Field Descriptions 986
22-17. ADG2SRC Register Field Descriptions 987
22-18. ADEVINTENA Register Field Descriptions 988
22-19. ADG1INTENA Register Field Descriptions 989
22-20. ADG2INTENA Register Field Descriptions 990
22-21. ADEVINTFLG Register Field Descriptions 991
22-22. ADG1INTFLG Register Field Descriptions 992
22-23. ADG2INTFLG Register Field Descriptions 993
22-24. ADEVTHRINTCR Register Field Descriptions 994
22-25. ADG1THRINTCR Register Field Descriptions 995
22-26. ADG2THRINTCR Register Field Descriptions 996
22-27. ADEVDMACR Register Field Descriptions 997
22-28. ADG1DMACR Register Field Descriptions 999
22-29. ADG2DMACR Register Field Descriptions 1001
22-30. ADBNDCR Register Field Descriptions 1003
22-31. ADBNDEND Register Field Descriptions 1004
22-32. ADEVSAMP Register Field Descriptions 1005
22-33. ADG1SAMP Register Field Descriptions 1006
22-34. ADG2SAMP Register Field Descriptions 1007
22-35. ADEVSR Register Field Descriptions 1008

22-36. ADG1SR Register Field Descriptions 1009
22-37. ADG2SR Register Field Descriptions 1010
22-38. ADEVSEL Register Field Descriptions 1011
22-39. ADG1SEL Register Field Descriptions 1012
22-40. ADG2SEL Register Field Descriptions 1013
22-41. ADCALR Register Field Descriptions 1014
22-42. ADSMSTATE Register Field Descriptions 1015
22-43. ADLASTCONV Register Field Descriptions 1016
22-44. ADEVBUFFER_0 to ADEVBUFFER_7 Register Field Descriptions 1018
22-45. ADG1BUFFER_0 to ADG1BUFFER_7 Register Field Descriptions 1020
22-46. ADG2BUFFER_0 to ADG2BUFFER_7 Register Field Descriptions 1022
22-47. ADEVEMUBUFFER Register Field Descriptions 1024
22-48. ADG1EMUBUFFER Register Field Descriptions 1026
22-49. ADG2EMUBUFFER Register Field Descriptions 1028
22-50. ADEVTDIR Register Field Descriptions 1029
22-51. ADEVTOUT Register Field Descriptions 1030
22-52. ADEVTIN Register Field Descriptions 1031
22-53. ADEVTSET Register Field Descriptions 1032
22-54. ADEVTCLR Register Field Descriptions 1033
22-55. ADEVTPDR Register Field Descriptions 1034
22-56. ADEVTPDIS Register Field Descriptions 1035
22-57. ADEVTPSEL Register Field Descriptions 1036
22-58. ADEVSAMPDISEN Register Field Descriptions 1037
22-59. ADG1SAMPDISEN Register Field Descriptions 1038
22-60. ADG2SAMPDISEN Register Field Descriptions 1039
22-61. ADMAGINT1CR Register Field Descriptions 1041
22-62. ADMAGINT1MASK Register Field Descriptions 1042
22-63. ADMAGINT2CR Register Field Descriptions 1044
22-64. ADMAGINT2MASK Register Field Descriptions 1045
22-65. ADMAGINT3CR Register Field Descriptions 1047
22-66. ADMAGINT3MASK Register Field Descriptions 1048
22-67. ADMAGINTENASET Register Field Descriptions 1049
22-68. ADMAGINTENACLR Register Field Descriptions 1050
22-69. ADMAGINTFLG Register Field Descriptions 1051
22-70. ADMAGINTOFF Register Field Descriptions 1052
22-71. ADEVFIFORESETCR Register Field Descriptions 1053
22-72. ADG1FIFORESETCR Register Field Descriptions 1054
22-73. ADG2FIFORESETCR Register Field Descriptions 1055
22-74. ADEVRAMWRADDR Register Field Descriptions 1056
22-75. ADG1RAMWRADDR Register Field Descriptions 1057
22-76. ADG2RAMWRADDR Register Field Descriptions 1058
22-77. ADPARCR Register Field Descriptions 1059
22-78. ADPARADDR Register Field Descriptions 1060
22-79. ADPWRUPDLYCTRL Register Field Descriptions 1061
22-80. ADEVCHNSELMODECTRL Register Field Descriptions 1062
22-81. ADG1CHNSELMODECTRL Register Field Descriptions 1063
22-82. ADG2CHNSELMODECTRL Register Field Descriptions 1064
22-83. ADEVCURRCOUNT Register Field Descriptions 1065
22-84. ADEVMAXCOUNT Register Field Descriptions 1066

22-85. ADG1CURRCOUNT Register Field Descriptions 1067
22-86. ADG1MAXCOUNT Register Field Descriptions 1068
22-87. ADG2CURRCOUNT Register Field Descriptions 1069
22-88. ADG2MAXCOUNT Register Field Descriptions 1070
23-1. N2HET RAM Base Addresses 1081
23-2. N2HET RAM Bank Structure 1082
23-3. Pin Safe State Upon Parity Error Detection 1083
23-4. N2HET Parity Bit Mapping 1084
23-5. Prescale Factor Register Encoding 1085
23-6. Interpretation of the 7-Bit HR Data Field 1086
23-7. Edge Detection Input Timing for Loop Resolution Instructions 1097
23-8. Edge Detection Input Timing for High Resolution Instructions 1097
23-9. Input Buffer, Output Buffer, and Pull Control Behavior 1103
23-10. N2HET Pin Disable Feature 1104
23-11. Pulse Length Examples for Suppression Filter 1105
23-12. Interrupt Sources and Corresponding Offset Values in Registers HETOFFx 1105
23-13. HWAG Interrupt Sources and Offset Values 1126
23-14. HWAG Interrupt Descriptions 1127
23-15. N2HET Registers 1135
23-16. Global Configuration Register (HETGCR) Field Descriptions 1136
23-17. Prescale Factor Register (HETPFR) Field Descriptions 1138
23-18. N2HET Current Address (HETADDR) Field Descriptions 1139
23-19. Offset Index Priority Level 1 Register (HETOFF1) Field Descriptions 1139
23-20. Interrupt Offset Encoding Format 1140
23-21. Offset Index Priority Level 2 Register (HETOFF2) Field Descriptions 1140
23-22. Interrupt Enable Set Register (HETINTENAS) Field Descriptions 1141
23-23. NHET Interrupt Enable Clear (HETINTENAC) Field Descriptions 1141
23-24. Exception Control Register 1 (HETEXC1) Field Descriptions 1142
23-25. Exception Control Register 2 (HETEXC2) Field Descriptions 1143
23-26. Interrupt Priority Register (HETPRY) Field Descriptions 1144
23-27. Interrupt Flag Register (HETFLG) Field Descriptions 1144
23-28. AND Share Control Register (HETAND) Field Descriptions 1145
23-29. HR Share Control Register (HETHRSH) Field Descriptions 1146
23-30. XOR Share Control Register (HETXOR) Field Descriptions 1147
23-31. Request Enable Set Register (HETREQENS) Field Descriptions 1148
23-32. Request Enable Clear Register (HETREQENC) Field Descriptions 1148
23-33. Request Destination Select Register (HETREQDS) Field Descriptions 1149
23-34. N2HET Direction Register (HETDIR) Field Descriptions 1150
23-35. N2HET Data Input Register (HETDIN) Field Descriptions 1151
23-36. N2HET Data Output Register (HETDOUT) Field Descriptions 1151
23-37. N2HET Data Set Register (HETDSET) Field Descriptions 1152
23-38. N2HET Data Clear Register (HETDCLR) Field Descriptions 1152
23-39. N2HET Open Drain Register (HETPDR) Field Descriptions 1153
23-40. N2HET Pull Disable Register (HETPULDIS) Field Descriptions 1153
23-41. N2HET Pull Select Register (HETPSL) Field Descriptions 1154
23-42. Parity Control Register (HETPCR) Field Descriptions 1155
23-43. Parity Address Register (HETPAR) Field Descriptions 1156
23-44. Parity Pin Register (HETPPR) Field Descriptions 1157
23-45. Known State on Parity Error 1157

23-46. Suppression Filter Preload Register (HETSFPRLD) Field Descriptions 1158
23-47. Suppression Filter Enable Register (HETSFENA) Field Descriptions 1158
23-48. Loop Back Pair Select Register (HETLBPSEL) Field Descriptions 1159
23-49. Loop Back Pair Direction Register (HETLBPDIR) Field Descriptions 1160
23-50. NHET Pin Disable Register (HETPINDIS) Field Descriptions 1161
23-51. HWAG Registers 1162
23-52. HWAG Pin Select Register (HWAPINSEL) Field Descriptions 1163
23-53. HWAG Global Control Register 0 (HWAGCR0) Field Descriptions 1164
23-54. HWAG Global Control Register 1 (HWAGCR1) Field Descriptions 1164
23-55. HWAG Global Control Register 2 (HWAGCR2) Field Descriptions 1165
23-56. HWAG Interrupt Enable Set Register (HWAENASET) Field Descriptions 1166
23-57. HWAG Interrupts 1166
23-58. HWAG Interrupt Enable Clear Register (HWAENACLR) Field Descriptions 1167
23-59. HWAG Interrupt Level Set Register (HWALVLSET) Field Descriptions 1168
23-60. HWAG Interrupt Level Clear Register (HWALVLCLR) Field Descriptions 1168
23-61. HWAG Interrupt Flag Register (HWAFLG) Field Descriptions 1169
23-62. HWAG Interrupt Offset Register 0 (HWAOFF0) Field Descriptions 1170
23-63. HWAG Interrupt Offset Register 1 (HWAOFF1) Field Descriptions 1171
23-64. HWAG Angle Value Register (HWAACNT) Field Descriptions 1172
23-65. HWAG Previous Tooth Period Value Register (HWAPCNT1) Field Descriptions 1173
23-66. HWAG Current Tooth Period Value Register (HWAPCNT) Field Descriptions 1173
23-67. HWAG Step Width Register (HWASTWD) Field Descriptions 1174
23-68. HWAG Teeth Number Register (HWATHNB) Field Descriptions 1175
23-69. HWAG Current Teeth Number Register (HWATHVL) Field Descriptions 1175
23-70. HWAG Filter Register (HWAFIL) Field Descriptions 1176
23-71. HWAG Filter Register 2 (HWAFIL2) Field Descriptions 1176
23-72. HWAG Angle Increment Register (HWAANGI) Field Descriptions 1177
23-73. Instruction Summary 1178
23-74. FLAGS Generated by Instruction 1179
23-75. Interrupt Capable Instructions 1179
23-76. Arithmetic / Bitwise Logic Sub-Opcodes 1191
23-77. Source Operand Choices 1191
23-78. Destination Operand Choices 1191
23-79. Shift Encoding 1192
23-80. Execution Time for ADC, ADD, AND, OR, SBB, SUB, XOR Instructions 1192
23-81. Move Types for ADM32 1197
23-82. Edge Select Encoding for APCNT 1200
23-83. Branch Condition Encoding for BR 1203
23-84. DADM64 Control Field Description 1208
23-85. Event Encoding Format for ECNT 1216
23-86. Magnitude Compare Order for MCMP 1218
23-87. Move Type Encoding Selection 1221
23-88. MOV64 Control Field Descriptions 1225
23-89. Comparison Type Encoding Format 1226
23-90. Counter Type Encoding Format 1228
23-91. Comparison Type Encoding Format 1235
23-92. RADM64 Control Field Descriptions 1235
23-93. Step Width Encoding for SCNT 1241
23-94. SHIFT MODE Encoding Format 1243

23-95. SHIFT Condition Encoding 1243
23-96. Event Encoding Format for WCAP 1246
23-97. Event Encoding Format for WCAPE 1248
24-1. CPENA / TMBx Priority Rules 1257
24-2. Triggered Control Packets 1260
24-3. DCP RAM 1262
24-4. DCP Parity RAM 1262
24-5. Field Addresses of the WCAP, ECNT, PCNT Example 1263
24-6. 32-Bit-Transfer of Data Fields 1264
24-7. Destination Buffer Values 1264
24-8. 64-Bit-Transfer of Control Field and Data Fields 1265
24-9. Destination Buffer Values 1265
24-10. HTU Control Registers 1266
24-11. Global Control Register (HTU GC) Field Descriptions 1267
24-12. Control Packet Enable Register (HTU CPENA) Field Descriptions 1268
24-13. CPENA Write Results 1268
24-14. CPENA Read Results 1268
24-15. Control Packet (CP) Busy Register 0 (HTU BUSY0) Field Descriptions 1269
24-16. Control Packet (CP) Busy Register 1 (HTU BUSY1) Field Descriptions 1270
24-17. Control Packet (CP) Busy Register 2 (HTU BUSY2) Field Descriptions 1270
24-18. Control Packet (CP) Busy Register 3 (HTU BUSY3) Field Descriptions 1271
24-19. Active Control Packet and Error Register (HTU ACPE) Field Descriptions 1271
24-20. Request Lost and Bus Error Control Register (HTU RLBECTRL) Field Descriptions 1273
24-21. Buffer Full Interrupt Enable Set Register (HTU BFINTS) Field Descriptions 1274
24-22. Buffer Full Interrupt Enable Clear Register (HTU BFINTC) Field Descriptions 1274
24-23. Interrupt Mapping Register (HTU INTMAP) Field Descriptions 1275
24-24. Interrupt Offset Register 0 (HTU INTOFF0) Field Descriptions 1276
24-25. Interrupt Offset Register 1 (HTU INTOFF1) Field Descriptions 1277
24-26. Buffer Initialization Mode Register (HTU BIM) Field Descriptions 1278
24-27. Buffer Initialization 1278
24-28. Request Lost Flag Register (HTU RLOSTFL) Field Descriptions 1280
24-29. Buffer Full Interrupt Flag Register (HTU BFINTFL) Field Descriptions 1280
24-30. BER Interrupt Flag Register (HTU BERINTFL) Field Descriptions 1281
24-31. Memory Protection 1 Start Address Register (HTU MP1S) Field Descriptions 1282
24-32. Memory Protection 1 End Address Register (HTU MP1E) Field Descriptions 1282
24-33. Debug Control Register (HTU DCTRL) Field Descriptions 1283
24-34. Watch Point Register (HTU WPR) Field Descriptions 1284
24-35. Watch Mask Register (HTU WMR) Field Descriptions 1284
24-36. Module Identification Register (HTU ID) Field Descriptions 1285
24-37. Parity Control Register (HTU PCR) Field Descriptions 1286
24-38. Parity Address Register (HTU PAR) Field Descriptions 1287
24-39. Memory Protection Control and Status Register (HTU MPCS) Field Descriptions 1288
24-40. Memory Protection 0 Start Address Register (HTU MP0S) Field Descriptions 1291
24-41. Memory Protection End Address Register (HTU MP0E) Field Descriptions 1291
24-42. Double Control Packet Memory Map 1292
24-43. Initial Full Address A Register (HTU IFADDRA) Field Descriptions 1293
24-44. Initial Full Address B Register (HTU IFADDRB) Field Descriptions 1293
24-45. Initial N2HET Address and Control Register (HTU IHADDRCT) Field Descriptions 1294
24-46. Initial Transfer Count Register (HTU ITCOUNT) Field Descriptions 1295

24-47. Current Full Address A Register (HTU CFADDRA) Field Descriptions 1296
24-48. Current Full Address B Register (HTU CFADDRB) Field Descriptions 1297
24-49. Current Frame Count Register (HTU CFCOUNT) Field Descriptions 1298
24-50. Application Examples for Setting the Transfer Modes of CP A and B of a DCP 1299
25-1. GIO Control Registers 1309
25-2. GIO Global Control Register (GIOGCR0) Field Descriptions 1310
25-3. GIO Interrupt Detect Register (GIOINTDET) Field Descriptions 1311
25-4. GIO Interrupt Polarity Register (GIOPOL) Field Descriptions 1312
25-5. GIO Interrupt Enable Set Register (GIOENASET) Field Descriptions 1313
25-6. GIO Interrupt Enable Clear Register (GIOENACLR) Field Descriptions 1314
25-7. GIO Interrupt Priority Register (GIOLVLSET) Field Descriptions 1315
25-8. GIO Interrupt Priority Register (GIOLVLCLR) Field Descriptions 1317
25-9. GIO Interrupt Flag Register (GIOFLG) Field Descriptions 1318
25-10. GIO Offset 1 Register (GIOOFF1) Field Descriptions 1319
25-11. GIO Offset 2 Register (GIOOFF2) Field Descriptions 1320
25-12. GIO Emulation 1 Register (GIOEMU1) Field Descriptions 1321
25-13. GIO Emulation 2 Register (GIOEMU2) Field Descriptions 1322
25-14. GIO Data Direction Registers (GIODIR[A-B]) Field Descriptions 1323
25-15. GIO Data Input Registers (GIODIN[A-B]) Field Descriptions 1323
25-16. GIO Data Output Registers (GIODOUT[A-B]) Field Descriptions 1324
25-17. GIO Data Set Registers (GIODSET[A-B]) Field Descriptions 1324
25-18. GIO Data Clear Registers (GIODCLR[A-B]) Field Descriptions 1325
25-19. GIO Open Drain Registers (GIOPDR[A-B]) Field Descriptions 1325
25-20. GIO Pull Disable Registers (GIOPULDIS[A-B]) Field Descriptions 1326
25-21. GIO Pull Select Registers (GIOPSL[A-B]) Field Descriptions 1326
25-22. Output Buffer and Pull Control Behavior for GIO Pins 1327
26-1. Parameters of the CAN Bit Time 1332
26-2. Message Object Field Descriptions 1338
26-3. Message RAM Addressing in Debug/Suspend and RDA Mode 1340
26-4. Message Interface Register Sets 1 and 2 1342
26-5. Message Interface Register 3 1344
26-6. DCAN Control Registers 1363
26-7. CAN Control Register Field Descriptions 1365
26-8. Error and Status Register Field Descriptions 1367
26-9. Error Counter Register Field Descriptions 1369
26-10. Bit Timing Register Field Descriptions 1370
26-11. Interrupt Register Field Descriptions 1371
26-12. Test Register Field Descriptions 1372
26-13. Parity Error Code Register Field Descriptions 1373
26-14. Core Release Register (DCAN REL) Field Descriptions 1373
26-15. Auto-Bus-On Time Register Field Descriptions 1374
26-16. Transmission Request Registers Field Descriptions 1375
26-17. New Data Registers Field Descriptions 1377
26-18. Interrupt Pending Registers Field Descriptions 1379
26-19. Message Valid Registers Field Descriptions 1381
26-20. Interrupt Multiplexer Registers Field Descriptions 1382
26-21. IF1/IF2 Command Register Field Descriptions 1384
26-22. IF1/IF2 Mask Register Field Descriptions 1386
26-23. IF1/IF2 Arbitration Register Field Descriptions 1387

26-24. IF1/IF2 Message Control Register Field Descriptions 1389
26-25. IF3 Observation Register Field Descriptions 1391
26-26. IF3 Mask Register Field Descriptions 1393
26-27. IF3 Arbitration Register Field Descriptions 1394
26-28. IF3 Message Control Register Field Descriptions 1395
26-29. IF3 Update Control Register Field Descriptions 1397
26-30. CAN TX IO Control Register Field Descriptions 1398
26-31. CAN RX IO Control Register Field Descriptions 1399
27-1. Pin Configurations 1404
27-2. Clocking Modes 1411
27-3. Pin Mapping for SIMO Pin with MSB First 1418
27-4. Pin Mapping for SOMI Pin with MSB First 1418
27-5. Pin Mapping for SIMO Pin with LSB First 1419
27-6. Pin Mapping for SOMI Pin with LSB First 1419
27-7. SPI Registers 1432
27-8. SPI Global Control Register 0 (SPIGCR0) Field Descriptions 1433
27-9. SPI Global Control Register 1 (SPIGCR1) Field Descriptions 1434
27-10. SPI Interrupt Register (SPIINT0) Field Descriptions 1435
27-11. SPI Interrupt Level Register (SPILVL) Field Descriptions 1437
27-12. SPI Flag Register (SPIFLG) Field Descriptions 1438
27-13. SPI Pin Control (SPIPC0) Field Descriptions 1441
27-14. SPI Pin Control Register (SPIPC1) Field Descriptions 1442
27-15. SPI Pin Control Register 2 (SPIPC2) Field Descriptions 1444
27-16. SPI Pin Control Register 3 (SPIPC3) Field Descriptions 1445
27-17. SPI Pin Control Register 4 (SPIPC4) Field Descriptions 1446
27-18. SPI Pin Control Register 5 (SPIPC5) Field Descriptions 1448
27-19. SPI Pin Control Register 6 (SPIPC6) Field Descriptions 1450
27-20. SPI Pin Control Register 7 (SPIPC7) Field Descriptions 1451
27-21. SPI Pin Control Register 8 (SPIPC8) Field Descriptions 1452
27-22. SPI Transmit Data Register 0 (SPIDAT0) Field Descriptions 1454
27-23. SPI Transmit Data Register 1 (SPIDAT1) Field Descriptions 1455
27-24. Chip Select Number Active 1457
27-25. SPI Receive Buffer Register (SPIBUF) Field Descriptions 1458
27-26. SPI Emulation Register (SPIEMU) Field Descriptions 1460
27-27. SPI Delay Register (SPIDELAY) Field Descriptions 1460
27-28. SPI Default Chip Select Register (SPIDEF) Field Descriptions 1463
27-29. SPI Data Format Registers (SPIFMT) Field Descriptions 1464
27-30. Transfer Group Interrupt Vector 0 (INTVECT0) 1466
27-31. Transfer Group Interrupt Vector 1 (INTVECT1) 1467
27-32. SPI Pin Control Register 9 (SPIPC9) Field Descriptions 1469
27-33. SPI Parallel/Modulo Mode Control Register (SPIPMCTRL) Field Descriptions 1470
27-34. Multi-buffer Mode Enable Register (MIBSPIE) Field Descriptions 1473
27-35. TG Interrupt Enable Set Register (TGITENST) Field Descriptions 1474
27-36. TG Interrupt Enable Clear Register (TGITENCR) Field Descriptions 1475
27-37. Transfer Group Interrupt Level Set Register (TGITLVST) Field Descriptions 1476
27-38. Transfer Group Interrupt Level Clear Register (TGITLVCR) Field Descriptions 1477
27-39. Transfer Group Interrupt Flag Register (TGINTFLG) Field Descriptions 1478
27-40. Tick Count Register (TICKCNT) Field Descriptions 1479
27-41. Last TG End Pointer (LTGPEND) Field Descriptions 1480

27-42. TG Control Registers (TGxCTRL) Field Descriptions 1481
27-43. DMA Channel Control Register (DMAxCTRL) Field Descriptions 1484
27-44. MibSPI DMAxCOUNT Register (ICOUNT) Field Descriptions 1486
27-45. MibSPI DMA Large Count Register (DMACNTLEN) Field Descriptions 1487
27-46. Multi-buffer RAM Uncorrectable Parity Error Control Register (UERRCTRL) Field Descriptions 1487
27-47. Multi-buffer RAM Uncorrectable Parity Error Status Register (UERRSTAT) Field Descriptions 1488
27-48. RXRAM Uncorrectable Parity Error Address Register (UERRADDR1) Field Descriptions 1489
27-49. TXRAM Uncorrectable Parity Error Address Register (UERRADDR0) Field Descriptions 1490
27-50. RXRAM Overrun Buffer Address Register (RXOVRN_BUF_ADDR) Field Descriptions 1491
27-51. I/O-Loopback Test Control Register (IOLPBKTSTCR) Field Descriptions 1492
27-52. SPI Extended Prescale Register 1 (EXTENDED_PRESCALE1) Field Descriptions 1494
27-53. SPI Extended Prescale Register 2 (EXTENDED_PRESCALE2) Field Descriptions 1496
27-54. Multi-Buffer RAM Register Summary 1499
27-55. Multi-Buffer RAM Transmit Data Register (TXRAM) Field Descriptions 1500
27-56. Chip Select Number Active 1502
27-57. Multi-Buffer Receive Buffer Register (RXRAM) Field Descriptions 1503
28-1. Superfractional Bit Modulation for SCI Mode (Normal Configuration) 1520
28-2. Superfractional Bit Modulation for SCI Mode (Maximum Configuration) 1521
28-3. SCI Mode (Minimum Configuration) 1521
28-4. SCI/LIN Interrupts 1528
28-5. Response Length Info Using IDBYTE Field Bits [5:4] for LIN Standards Earlier than 1.3 1535
28-6. Response Length with SCIFORMAT[18:16] Programming 1535
28-7. Superfractional Bit Modulation for LIN Master Mode and Slave Mode 1537
28-8. Timeout Values in T_{bit} Units 1544
28-9. SCI/LIN Control Registers 1557
28-10. SCI Global Control Register 0 (SCIGCR0) Field Descriptions 1558
28-11. SCI Global Control Register 1 (SCIGCR1) Field Descriptions 1559
28-12. SCI Receiver Status Flags 1562
28-13. SCI Transmitter Status Flags 1562
28-14. SCI Global Control Register 2 (SCIGCR2) Field Descriptions 1563
28-15. SCI Set Interrupt Register (SCISETINT) Field Descriptions 1565
28-16. SCI Clear Interrupt Register (SCICLEARINT) Field Descriptions 1568
28-17. SCI Set Interrupt Level Register (SCISETINTLVL) Field Descriptions 1572
28-18. SCI Clear Interrupt Level Register (SCICLEARINTLVL) Field Descriptions 1575
28-19. SCI Flags Register (SCIFLR) Field Descriptions 1578
28-20. SCI Interrupt Vector Offset 0 (SCIINTVECT0) Field Descriptions 1585
28-21. SCI Interrupt Vector Offset 1 (SCIINTVECT1) Field Descriptions 1585
28-22. SCI Format Control Register (SCIFORMAT) Field Descriptions 1586
28-23. Baud Rate Selection Register (BRS) Field Descriptions 1587
28-24. Comparative Baud Values for Different P Values, Asynchronous Mode 1588
28-25. Receiver Emulation Data Buffer (SCIED) Field Descriptions 1588
28-26. Receiver Data Buffer (SCIRD) Field Descriptions 1589
28-27. Transmit Data Buffer Register (SCITD) Field Descriptions 1590
28-28. SCI Pin I/O Control Register 0 (SCIPIO0) Field Descriptions 1590
28-29. SCI Pin I/O Control Register 1 (SCIPIO1) Field Descriptions 1591
28-30. LINTX Pin Control 1591
28-31. LINRX Pin Control 1591
28-32. SCI Pin I/O Control Register 2 (SCIPIO2) Field Descriptions 1592
28-33. SCI Pin I/O Control Register 3 (SCIPIO3) Field Descriptions 1593

28-34. SCI Pin I/O Control Register 4 (SCIPIO4) Field Descriptions 1594
28-35. SCI Pin I/O Control Register 5 (SCIPIO5) Field Descriptions 1595
28-36. SCI Pin I/O Control Register 6 (SCIPIO6) Field Descriptions 1596
28-37. SCI Pin I/O Control Register 7 (SCIPIO7) Field Descriptions 1597
28-38. SCI Pin I/O Control Register 8 (SCIPIO8) Field Descriptions 1597
28-39. LIN Compare Register (LINCOMPARE) Field Descriptions 1598
28-40. LIN Receive Buffer 0 Register (LINRD0) Field Descriptions 1599
28-41. LIN Receive Buffer 1 Register (RD1) Field Descriptions 1599
28-42. LIN Mask Register (LINMASK) Field Descriptions 1600
28-43. LIN Identification Register (LINID) Field Descriptions 1601
28-44. LIN Transmit Buffer 0 Register (LINTD0) Field Descriptions 1602
28-45. LIN Transmit Buffer 1 Register (LINTD1) Field Descriptions 1602
28-46. Maximum Baud Rate Selection Register (MBRS) Field Descriptions 1603
28-47. Input/Output Error Enable Register (IODFTCTRL) Field Descriptions 1604
28-48. Input Buffer, Output Buffer, and Pull Control Behavior as GPIO Pins 1607
29-1. SCI Interrupts 1618
29-2. DMA and Interrupt Requests in Multiprocessor Modes 1619
29-3. SCI Control Registers Summary 1623
29-4. SCI Global Control Register 0 (SCIGCR0) Fied Descriptions 1624
29-5. SCI Global Control Register 1 (SCIGCR1) Field Descriptions 1625
29-6. SCI Set Interrupt Register (SCISETINT) Field Descriptions 1628
29-7. SCI Clear Interrupt Register (SCICLEARINT) Field Descriptions 1630
29-8. SCI Set Interrupt Level Register (SCISETINTLVL) Field Descriptions 1632
29-9. SCI Clear Interrupt Level Register (SCICLEARINTLVL) Field Descriptions 1633
29-10. SCI Flags Register (SCIFLR) Field Descriptions 1635
29-11. SCI Receiver Status Flags 1638
29-12. SCI Transmitter Status Flags 1638
29-13. SCI Interrupt Vector Offset 0 (SCIINTVECT0) Field Descriptions 1639
29-14. SCI Interrupt Vector Offset 1 (SCIINTVECT1) Field Descriptions 1639
29-15. SCI Format Control Register (SCIFORMAT) Field Descriptions 1640
29-16. Baud Rate Selection Register (BRS) Field Descriptions 1641
29-17. Comparative Baud Values (Asynchronous Mode) 1641
29-18. Receiver Emulation Data Buffer (SCIED) Field Descriptions 1642
29-19. Receiver Data Buffer (SCIRD) Field Descriptions 1642
29-20. Transmit Data Buffer Register (SCITD) Field Descriptions 1643
29-21. SCI Pin I/O Control Register 0 (SCIPIO0) Field Descriptions 1643
29-22. SCI Pin I/O Control Register 1 (SCIPIO1) Field Descriptions 1644
29-23. SCITX Pin Control 1644
29-24. SCIRX Pin Control 1644
29-25. SCI Pin I/O Control Register 2 (SCIPIO2) Field Descriptions 1645
29-26. SCI Pin I/O Control Register 3 (SCIPIO3) Field Descriptions 1646
29-27. SCI Pin I/O Control Register 4 (SCIPIO4) Field Descriptions 1647
29-28. SCI Pin I/O Control Register 5 (SCIPIO5) Field Descriptions 1648
29-29. SCI Pin I/O Control Register 6 (SCIPIO6) Field Descriptions 1649
29-30. SCI Pin I/O Control Register 7 (SCIPIO7) Field Descriptions 1650
29-31. SCI Pin I/O Control Register 8 (SCIPIO8) Field Descriptions 1650
29-32. Input/Output Error Enable Register (IODFTCTRL) Field Descriptions 1651
29-33. Input Buffer, Output Buffer, and Pull Control Behavior as GPIO Pins 1654
30-1. Ways to Generate a NACK Bit 1663

30-2. Interrupt Requests Generated by I2C Module 1668
30-3. I2C Control Registers 1671
30-4. I2C Own Address Manager Register (I2COAR) Field Descriptions 1672
30-5. Correct Mode for OA Bits 1672
30-6. I2C Interrupt Mask Register (I2CIMR) Field Descriptions 1673
30-7. I2C Status Register (I2CSTR) Field Descriptions 1674
30-8. I2C Clock Divider Low Register (I2CCKL) Field Descriptions 1677
30-9. I2C Clock Control High Register (I2CCKH) Field Descriptions 1677
30-10. I2C Data Count Register (I2CCNT) Field Descriptions 1678
30-11. I2C Data Receive Register (I2CDRR) Field Descriptions 1678
30-12. I2C Slave Address Register (I2CSAR) Field Descriptions 1679
30-13. Correct Mode for SA Bits 1679
30-14. I2C Data Transmit Register (I2CDXR) Field Descriptions 1679
30-15. I2C Mode Register (I2CMDR) Field Descriptions 1680
30-16. I2C Module Condition, Bus Activity, and Mode 1682
30-17. I2C Module Operating Modes 1682
30-18. Number of Bits Sent on Bus 1682
30-19. I2C Interrupt Vector Register (I2CIVR) Field Descriptions 1683
30-20. Interrupt Codes for INTCODE Bits 1683
30-21. I2C Extended Mode Register (I2CEMDR) Field Descriptions 1684
30-22. I2C Prescale Register (I2CPSC) Field Descriptions 1684
30-23. I2C Peripheral ID Register 1 (I2CPID1) Field Descriptions 1685
30-24. I2C Peripheral ID Register 2 (I2CPID2) Field Descriptions 1685
30-25. I2C DMA Control Register (I2CDMACR) Field Descriptions 1686
30-26. I2C Pin Function Register (I2CPFNC) Field Descriptions 1686
30-27. I2C Pin Direction Register (I2CPDIR) Field Descriptions 1687
30-28. I2C Data Input Register (I2CDIN) Field Descriptions 1687
30-29. I2C Data Output Register (I2CDOUT) Field Descriptions 1688
30-30. I2C Data Set Register (I2CDSET) Field Description 1688
30-31. I2C Data Clear Register (I2CDSET) Field Descriptions 1689
30-32. I2C Pin Open Drain Register (I2CPDR) Field Descriptions 1689
30-33. I2C Pull Disable Register (I2CPDIS) Field Descriptions 1690
30-34. I2C Pull Select Register (I2CPSEL) Field Descriptions 1690
30-35. Input Buffer, Output Buffer, and Pull Control Behavior as GPIO Pins 1691
30-36. I2C Pins Slew Rate Select Register (I2CSRS) Field Descriptions 1691
31-1. EMAC and MDIO Signals for MII Interface 1698
31-2. EMAC and MDIO Signals for RMII Interface 1699
31-3. MDIO Multiplexing Control 1700
31-4. MII / RMII Multiplexing Control 1700
31-5. Ethernet Frame Description 1701
31-6. Basic Descriptor Description 1703
31-7. Receive Frame Treatment Summary 1731
31-8. Middle of Frame Overrun Treatment 1732
31-9. Emulation Control 1742
31-10. EMAC Control Module Registers 1743
31-11. EMAC Control Module Revision ID Register (REVID) Field Descriptions 1744
31-12. EMAC Control Module Software Reset Register (SOFTRESET) 1744
31-13. EMAC Control Module Interrupt Control Register (INTCONTROL) 1745
31-14. EMAC Control Module Receive Threshold Interrupt Enable Register (C0RXTHRESHEN) 1746

31-15. EMAC Control Module Receive Interrupt Enable Register (C0RXEN) 1747
31-16. EMAC Control Module Transmit Interrupt Enable Register (C0TXEN) 1748
31-17. EMAC Control Module Miscellaneous Interrupt Enable Register (C0MISCEN) 1749
31-18. EMAC Control Module Receive Threshold Interrupt Status Register (C0RXTHRESHSTAT) 1750
31-19. EMAC Control Module Receive Interrupt Status Register (C0RXSTAT) 1751
31-20. EMAC Control Module Transmit Interrupt Status Register (C0TXSTAT) 1752
31-21. EMAC Control Module Miscellaneous Interrupt Status Register (C0MISCSTAT) 1753
31-22. EMAC Control Module Receive Interrupts Per Millisecond Register (C0RXIMAX) 1754
31-23. EMAC Control Module Transmit Interrupts Per Millisecond Register (C0TXIMAX) 1755
31-24. Management Data Input/Output (MDIO) Registers 1756
31-25. MDIO Revision ID Register (REVID) Field Descriptions 1756
31-26. MDIO Control Register (CONTROL) Field Descriptions 1757
31-27. PHY Acknowledge Status Register (ALIVE) Field Descriptions 1758
31-28. PHY Link Status Register (LINK) Field Descriptions 1758
31-29. MDIO Link Status Change Interrupt (Unmasked) Register (LINKINTRAW) Field Descriptions 1759
31-30. MDIO Link Status Change Interrupt (Masked) Register (LINKINTMASKED) Field Descriptions 1760
31-31. MDIO User Command Complete Interrupt (Unmasked) Register (USERINTRAW) Field Descriptions 1761
31-32. MDIO User Command Complete Interrupt (Masked) Register (USERINTMASKED) Field Descriptions 1762
31-33. MDIO User Command Complete Interrupt Mask Set Register (USERINTMASKSET) Field Descriptions 1763
31-34. MDIO User Command Complete Interrupt Mask Clear Register (USERINTMASKCLEAR) Field Descriptions 1764
31-35. MDIO User Access Register 0 (USERACCESS0) Field Descriptions 1765
31-36. MDIO User PHY Select Register 0 (USERPHYSEL0) Field Descriptions 1766
31-37. MDIO User Access Register 1 (USERACCESS1) Field Descriptions 1767
31-38. MDIO User PHY Select Register 1 (USERPHYSEL1) Field Descriptions 1768
31-39. Ethernet Media Access Controller (EMAC) Registers 1769
31-40. Transmit Revision ID Register (TXREVID) Field Descriptions 1772
31-41. Transmit Control Register (TXCONTROL) Field Descriptions 1772
31-42. Transmit Teardown Register (TXTEARDOWN) Field Descriptions 1773
31-43. Receive Revision ID Register (RXREVID) Field Descriptions 1773
31-44. Receive Control Register (RXCONTROL) Field Descriptions 1774
31-45. Receive Teardown Register (RXTEARDOWN) Field Descriptions 1774
31-46. Transmit Interrupt Status (Unmasked) Register (TXINTSTATRAW) Field Descriptions 1775
31-47. Transmit Interrupt Status (Masked) Register (TXINTSTATMASKED) Field Descriptions 1776
31-48. Transmit Interrupt Mask Set Register (TXINTMASKSET) Field Descriptions 1777
31-49. Transmit Interrupt Mask Clear Register (TXINTMASKCLEAR) Field Descriptions 1778
31-50. MAC Input Vector Register (MACINVECTOR) Field Descriptions 1779
31-51. MAC End Of Interrupt Vector Register (MACEOIVECTOR) Field Descriptions 1780
31-52. Receive Interrupt Status (Unmasked) Register (RXINTSTATRAW) Field Descriptions 1781
31-53. Receive Interrupt Status (Masked) Register (RXINTSTATMASKED) Field Descriptions 1782
31-54. Receive Interrupt Mask Set Register (RXINTMASKSET) Field Descriptions 1783
31-55. Receive Interrupt Mask Clear Register (RXINTMASKCLEAR) Field Descriptions 1784
31-56. MAC Interrupt Status (Unmasked) Register (MACINTSTATRAW) Field Descriptions 1785
31-57. MAC Interrupt Status (Masked) Register (MACINTSTATMASKED) Field Descriptions 1785
31-58. MAC Interrupt Mask Set Register (MACINTMASKSET) Field Descriptions 1786
31-59. MAC Interrupt Mask Clear Register (MACINTMASKCLEAR) Field Descriptions 1786
31-60. Receive Multicast/Broadcast/Promiscuous Channel Enable Register (RXMBPENABLE) Field Descriptions 1787
31-61. Receive Unicast Enable Set Register (RXUNICASTSET) Field Descriptions 1789

31-62. Receive Unicast Clear Register (RXUNICASTCLEAR) Field Descriptions 1790
31-63. Receive Maximum Length Register (RXMAXLEN) Field Descriptions 1790
31-64. Receive Buffer Offset Register (RXBUFFEROFFSET) Field Descriptions 1791
31-65. Receive Filter Low Priority Frame Threshold Register (RXFILTERLOWTHRESH) Field Descriptions 1791
31-66. Receive Channel *n* Flow Control Threshold Register (RX*n*FLOWTHRESH) Field Descriptions 1792
31-67. Receive Channel *n* Free Buffer Count Register (RX*n*FREEBUFFER) Field Descriptions 1792
31-68. MAC Control Register (MACCONTROL) Field Descriptions 1793
31-69. MAC Status Register (MACSTATUS) Field Descriptions 1795
31-70. Emulation Control Register (EMCONTROL) Field Descriptions 1797
31-71. FIFO Control Register (FIFOCONTROL) Field Descriptions 1797
31-72. MAC Configuration Register (MACCONFIG) Field Descriptions 1798
31-73. Soft Reset Register (SOFTRESET) Field Descriptions 1798
31-74. MAC Source Address Low Bytes Register (MACSRCADDRLO) Field Descriptions 1799
31-75. MAC Source Address High Bytes Register (MACSRCADDRHI) Field Descriptions 1799
31-76. MAC Hash Address Register 1 (MACHASH1) Field Descriptions 1800
31-77. MAC Hash Address Register 2 (MACHASH2) Field Descriptions 1800
31-78. Back Off Test Register (BOFFTEST) Field Descriptions 1801
31-79. Transmit Pacing Algorithm Test Register (TPACETEST) Field Descriptions 1801
31-80. Receive Pause Timer Register (RXPAUSE) Field Descriptions 1802
31-81. Transmit Pause Timer Register (TXPAUSE) Field Descriptions 1802
31-82. MAC Address Low Bytes Register (MACADDRLO) Field Descriptions 1803
31-83. MAC Address High Bytes Register (MACADDRHI) Field Descriptions 1804
31-84. MAC Index Register (MACINDEX) Field Descriptions 1804
31-85. Transmit Channel *n* DMA Head Descriptor Pointer Register (TX*n*HDP) Field Descriptions 1805
31-86. Receive Channel *n* DMA Head Descriptor Pointer Register (RX*n*HDP) Field Descriptions 1805
31-87. Transmit Channel *n* Completion Pointer Register (TX*n*CP) Field Descriptions 1806
31-88. Receive Channel *n* Completion Pointer Register (RX*n*CP) Field Descriptions 1806
32-1. USB Host / Device Interface Signal Multiplexing and Control 1818
32-2. USB Host Controller Registers 1820
32-3. OHCI Revision Number Register (HCREVISION) Bit Field Descriptions 1821
32-4. HC Operating Mode Register (HCCONTROL) Bit Field Descriptions 1821
32-5. HC Command and Status Register (HCCOMMANDSTATUS) Bit Field Descriptions 1823
32-6. HC Interrupt Status Register (HCINTERRUPTSTATUS) Bit Field Descriptions 1824
32-7. HC Interrupt Enable Register (HCINTERRUPTENABLE) Bit Field Descriptions 1825
32-8. HC Interrupt Disable Register (HCINTERRUPTDISABLE) Bit Field Descriptions 1826
32-9. HC HCCA Address Register (HCHCCA) Bit Field Descriptions 1827
32-10. HC Current Periodic Register (HCPERIODCURRENTED) Bit Field Descriptions 1828
32-11. HC Head Control Register (HCCONTROLHEADED) Bit Field Descriptions 1828
32-12. HC Current Control Register (HCCONTROLCURRENTED) Bit Field Descriptions 1829
32-13. HC Head Bulk Register (HCBULKHEADED) Bit Field Descriptions 1829
32-14. HC Current Bulk Register (HCBULKCURRENTED) Bit Field Descriptions 1830
32-15. HC Head Done Register (HCDONEHEAD) Bit Field Descriptions 1830
32-16. HC Frame Interval Register (HCFMINTERVAL) Bit Field Descriptions 1831
32-17. HC Frame Remaining Register (HCFMREMAINING) Bit Field Descriptions 1831
32-18. HC Frame Number Register (HCFMNUMBER) Bit Field Descriptions 1832
32-19. HC Periodic Start Register (HCPERIODICSTART) Bit Field Descriptions 1832
32-20. HC Low-Speed Threshold Register (HCLSTHRESHOLD) Bit Field Descriptions 1833
32-21. HC Root Hub A Register (HCRHDESCRIPTORA) Field Descriptions 1833
32-22. HC Root Hub B Register (HCRHDESCRIPTORB) Bit Field Descriptions 1835

32-23. HC Root Hub Status Register (HCRHSTATUS) Bit Field Descriptions 1836
32-24. HC Port 0 Status and Control Register (HCRHPORTSTATUS0) Field Descriptions 1837
32-25. HC Port 1 Status and Control Register (HCRHPORTSTATUS1) Field Descriptions 1839
32-26. Host UE Address Register (HOSTUEADDR) Field Descriptions 1841
32-27. Host UE Status Register (HOSTUESTATUS) Bit Field Descriptions.......... 1841
32-28. Host Time-out Control Register (HOSTTIMEOUTCTRL) Bit Field Descriptions.......... 1842
32-29. Host Revision Register (HOSTREVISION) Field Descriptions.......... 1842
32-30. USB Device Controller Registers 1845
32-31. Revision Register (REV) Field Descriptions 1846
32-32. Endpoint Selection Register (EP_NUM) Field Descriptions 1847
32-33. Data Register (DATA) Field Descriptions 1848
32-34. Control Register (CTRL) Field Descriptions 1849
32-35. Status Register (STAT_FLG) Field Descriptions.......... 1850
32-36. Receive FIFO Status Register (RXFSTAT) Field Descriptions.......... 1853
32-37. System Configuration Register 1 (SYSCON1) Field Descriptions 1854
32-38. Little-Endian and Big-Endian Formats.......... 1855
32-39. System Configuration Register 2 (SYSCON2) Field Descriptions 1856
32-40. Device Status Register (DEVSTAT) Field Descriptions.......... 1857
32-41. Start of Frame Register (SOF) Field Descriptions 1859
32-42. Interrupt Enable Register (IRQ_EN) Field Descriptions.......... 1860
32-43. DMA Interrupt Enable Register (DMA_IRQ_EN) Field Descriptions 1861
32-44. Interrupt Source Register (IRQ_SRC) Field Descriptions.......... 1862
32-45. Non-ISO Endpoint Interrupt Status Register (EPN_STAT) Field Descriptions.......... 1865
32-46. Non-ISO DMA Interrupt Status Register (DMAN_STAT) Field Descriptions 1866
32-47. DMA Receive Channels Configuration Register (RXDMA_CFG) Field Descriptions 1867
32-48. DMA Transmit Channels Configuration Register (TXDMA_CFG) Field Descriptions.......... 1868
32-49. DMA FIFO Data Register (DATA_DMA) Field Descriptions.......... 1869
32-50. Transmit DMA Control Register n (TXDMAn) Field Descriptions.......... 1870
32-51. Receive DMA Control Register n (RXDMAn) Field Descriptions 1871
32-52. Endpoint 0 Configuration Register (EP0) Field Descriptions 1872
32-53. Receive Endpoint n Configuration Register (EPn_RX) Field Descriptions 1873
32-54. Transmit Endpoint n Configuration Register (EPn_TX) Field Descriptions.......... 1875
32-55. Autodecoded Versus Non-Autodecoded Control Requests 1894
32-56. USB Device Controller Interrupt Type by Endpoint Type 1925
32-57. Signaling Between USB Controller and Unidirectional USB Transceiver Using DAT/SE0 Signaling 1939
33-1. ESM Signals Set by eFuse Controller 1944
33-2. eFuse Controller Registers.......... 1947
33-3. EFC Boundary Register (EFCBOUND) Field Descriptions 1947
33-4. EFC Pins Register (EFCPINS) Field Descriptions.......... 1949
33-5. EFC Error Status Register (EFCERRSTAT) Field Descriptions 1950
33-6. EFC Self Test Cycles Register (EFCSTCY) Field Descriptions 1950
33-7. EFC Self Test Cycles Register (EFCSTSIG) Field Descriptions.......... 1951

Read This First

About This Manual

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

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

Notational Conventions

This document uses the following conventions.

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

Glossary

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

Related Documentation From Texas Instruments

For product information, visit the Texas Instruments website at http://www.ti.com.

SPNA106— *Initialization of Hercules™ ARM® Cortex®-R4F Microcontrollers Application Report.* Provides a brief overview and initialization procedure of the TMS570LS31x series and the RM4x series of microcontrollers in the Hercules family.

SPNS182— *RM46Lx30 16- and 32-BIT RISC Flash Microcontroller Data Manual.*

SPNS183— *RM46Lx40 16- and 32-BIT RISC Flash Microcontroller Data Manual.*

SPNS184— *RM46Lx50 16- and 32-BIT RISC Flash Microcontroller Data Manual.*

SPNS185— *RM46L852 16- and 32-BIT RISC Flash Microcontroller Data Manual.*

SPNU551— *Safety Manual for RM46x Hercules™ ARM® Safety Critical Microcontrollers User's Guide.* A safety manual for the Texas Instruments Hercules safety critical microcontroller product family. The product family utilizes a common safety architecture that is implemented in multiple application focused products.

SPNU566— *RM46x Hercules™ Development Kit (HDK) User's Guide.* Describes the board level operations of the RM46 Hercules Development Kit (HDK). The HDK is based on the Texas Instruments RM46L852 Microcontroller. The RM46 HDK is a table top card that allows engineers and software developers to evaluate certain characteristics of the RM46L852 microcontroller to determine if the microcontroller meets the designer's application requirements as well as begin early application development. Evaluators can create software to execute on board or expand the system in a variety of ways.

Community Resources

The following links connect to TI community resources. Linked contents are provided "AS IS" by the respective contributors. They do not constitute TI specifications and do not necessarily reflect TI's views; see TI's Terms of Use.

TI E2E™ Online Community— ***TI's Engineer-to-Engineer (E2E) Community.*** Created to foster collaboration among engineers. At e2e.ti.com, you can ask questions, share knowledge, explore ideas and help solve problems with fellow engineers.

TI Embedded Processors Wiki— ***Texas Instruments Embedded Processors Wiki.*** Established to help developers get started with Embedded Processors from Texas Instruments and to foster innovation and growth of general knowledge about the hardware and software surrounding these devices.

Trademarks

Hercules, E2E are trademarks of Texas Instruments.
CoreSight is a trademark of ARM Limited.
ARM, Cortex are registered trademarks of ARM Limited.

Chapter 1

SPNU514C–March 2018

Introduction

Topic **Page**

1.1 Designed for Safety Applications 96
1.2 Family Description 96
1.3 Endianism Considerations 99

1.1 Designed for Safety Applications

The RM46x device architecture has been designed from the ground up to simplify development of functionally safe systems. The basic architectural concept is known as a safe island approach. Power, clock, reset, and basic processing function are protected to a high level of diagnostic coverage in hardware. Some of the key features of the safe island region are:

- A dual core lockstep processing solution built around ARM® Cortex®-R4F CPU that detects failures at the core boundary on a cycle by cycle basis. Special measures in processor layout, clock distribution, power distribution, reset distribution, and temporal diversity are all implemented to mitigate common cause failures of the logical CPU and its checker. For complete details on the ARM® Cortex®-R4F CPU, refer to the ARM® Cortex®-R4F Technical Reference Manual.
- Hardware BIST controllers which provide an extremely high level of diagnostic coverage for the lockstep CPUs and SRAMs in the system, while executing faster and consuming less memory than equivalent software-based self-test solutions
- ECC on the SRAM and flash memories tightly coupled to the R4F. The ECC controllers are located inside the CPU. This approach has two key advantages:
  - The interconnect between CPU and the memory is also covered by the diagnostic
  - The ECC logic itself is checked on a cycle by cycle basis
- Onboard voltage and reset monitoring logic
- Onboard oscillator and PLL failure detection logic including a backup RC oscillator which can be utilized upon failure

The RM46x device architecture also includes many features to simplify diagnostics of remaining logic such as:

- Continuous parity diagnostics on all peripheral memories
- Analog and digital loopback to test for shorts on I/O
- HW self-test and diagnostics on the ADC module to check integrity of both analog inputs and the ADC core conversion function
- A DMA driven hardware engine for the background calculation of CRC signatures during data transfers
- A centralized error reporting function including a status output pin to enable external monitoring of the device status

1.2 Family Description

The RM46x integrates the ARM® Cortex®-R4F Floating Point CPU that offers an efficient 1.6 DMIPS/MHz and has configurations that can run up to 220MHz providing up to 352 DMIPS. The device supports the little-endian [LE] format.

The RM46x has up to 1.25MB integrated Flash and up to 192KB data RAM configurations with single bit error correction and double bit error detection. The flash memory on this device is a nonvolatile, electrically erasable and programmable memory implemented with a 64-bit-wide data bus interface. The flash operates on a 3.3V supply input (same level as I/O supply) for all read, program and erase operations. When in pipeline mode, the flash operates with a system clock frequency of up to 220MHz. The SRAM supports single-cycle read/write accesses in byte, halfword, and word modes.

The RM46x device features peripherals for real-time control-based applications, including two Next Generation High End Timer (N2HET) timing coprocessors with up to 40 total IO terminals and a 12-bit A to D converter supporting up to 24 inputs.

The N2HET is an advanced intelligent timer that provides sophisticated timing functions for real-time applications. The timer is software-controlled, using a reduced instruction set, with a specialized timer micromachine and an attached I/O port. The N2HET can be used for pulse width modulated outputs, capture or compare inputs, or general-purpose I/O. It is especially well suited for applications requiring multiple sensor information and drive actuators with complex and accurate time pulses. A High End Timer Transfer Unit (HET-TU) can perform DMA type transactions to transfer N2HET data to or from main memory. A Memory Protection Unit (MPU) is built into the HET-TU.

The device has two 12-bit-resolution MibADCs with 24 total channels and 64 words of parity protected buffer RAM each. The MibADC channels can be converted individually or can be grouped by software for sequential conversion sequences. Sixteen channels are shared between the two MibADCs. There are three separate groupings. Each sequence can be converted once when triggered or configured for continuous conversion mode.

The device has multiple communication interfaces: three MibSPIs, up to two SPIs, one LIN, one SCI, three DCANs, one I^{2}C, one Ethernet, and USB Host and Device modules. The SPI provides a convenient method of serial interaction for high-speed communications between similar shift-register type devices. The LIN supports the Local Interconnect standard 2.0 and can be used as a UART in full-duplex mode using the standard Non-Return-to-Zero (NRZ) format. The DCAN supports the CAN 2.0B protocol standard and uses a serial, multimaster communication protocol that efficiently supports distributed real-time control with robust communication rates of up to 1 megabit per second (Mbps). The DCAN is ideal for applications operating in noisy and harsh environments (for example, automotive and industrial fields) that require reliable serial communication or multiplexed wiring. The Ethernet module supports MII and MDIO interfaces.

The I2C module is a multi-master communication module providing an interface between the microcontroller and an I2C compatible device via the I2C serial bus. The I2C supports both 100 Kbps and 400 Kbps speeds.

The frequency-modulated phase-locked loop (FMPLL) clock module is used to multiply the external frequency reference to a higher frequency for internal use. The FMPLL provides one of the seven possible clock source inputs to the global clock module (GCM). The GCM module manages the mapping between the available clock sources and the device clock domains.

The device also has an external clock prescaler (ECP) module that when enabled, outputs a continuous external clock on the ECLK pin/ball. The ECLK frequency is a user-programmable ratio of the peripheral interface clock (VCLK) frequency. This low frequency output can be monitored externally as an indicator of the device operating frequency.

The Direct Memory Access Controller (DMA) has 16 channels, 32 control packets and parity protection on its memory. A Memory Protection Unit (MPU) is built into the DMA to protect memory against erroneous transfers.

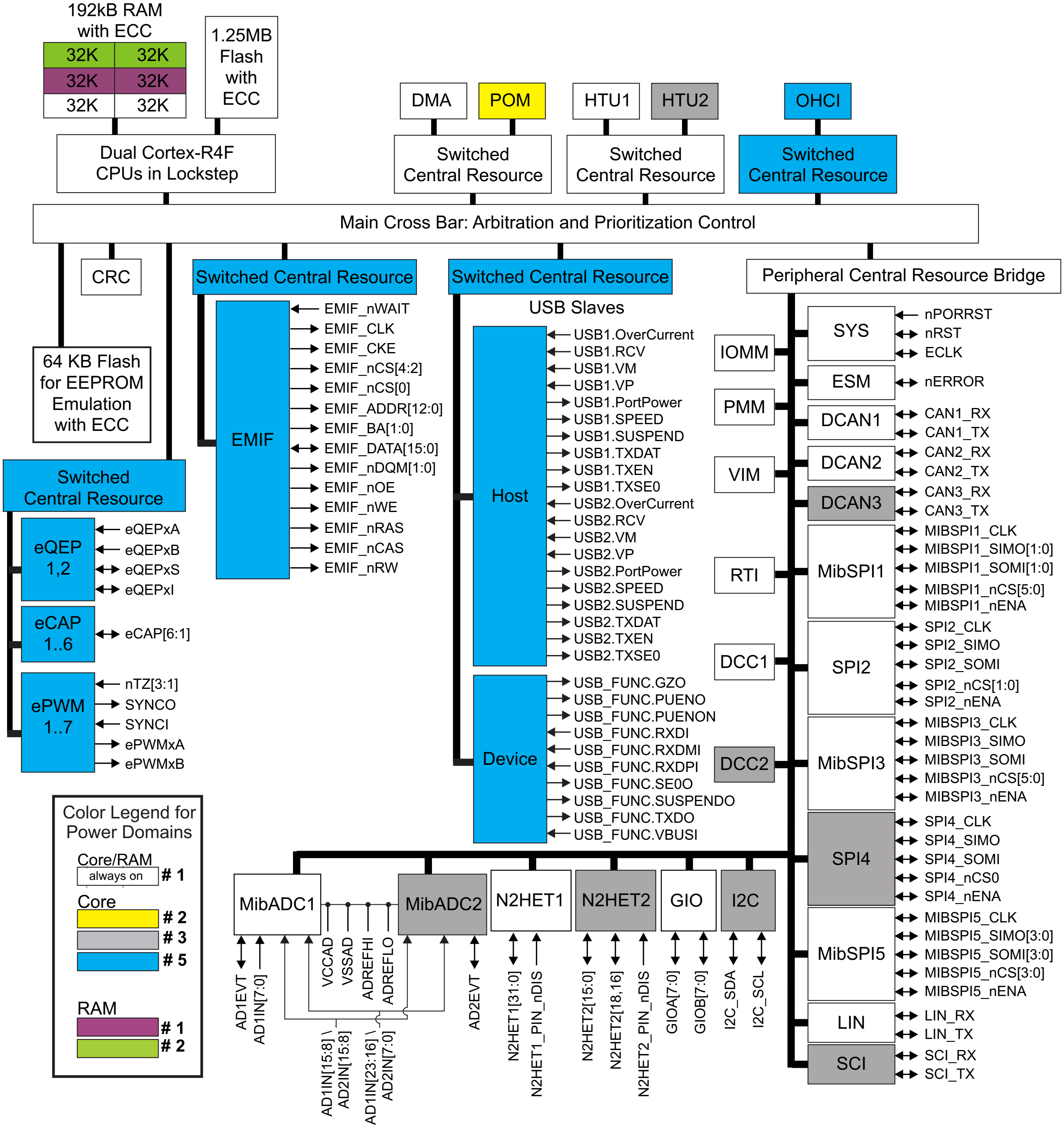
The Error Signaling Module (ESM) monitors all device errors and determines whether an interrupt or external Error pin/ball is triggered when a fault is detected. The nERROR can be monitored externally as an indicator of a fault condition in the microcontroller.

The External Memory Interface (EMIF) provides a memory extension to asynchronous and synchronous memories or other slave devices.

The device supports all the built-in ARM® Cortex®-R4F CoreSight™ debug features. In addition, a Parameter Overlay Module (POM) can re-route Flash accesses to internal memory or to external memory via the EMIF, thus avoiding the re-programming steps necessary for parameter updates in Flash.

With integrated safety features and a wide choice of communication and control peripherals, the RM46x is an ideal solution for high performance real time control applications with safety critical requirements.

Figure 1-1. Block Diagram



1.3 Endianism Considerations

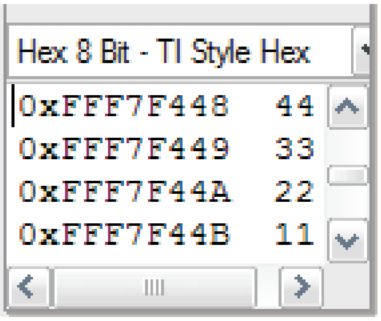
1.3.1 RM46x: Little Endian (LE)

The RM46x family is based on the ARM® Cortex®-R4F core. ARM has designed this core to be used in big-endian and little-endian systems. For the TI RM46x family, the endianness has been configured to little-endian. Little-endian systems store the least-significant byte of a multi-byte data field in the lowest memory address. Also, the address of the multi-byte data field is the lowest address. Following is an example of the physical addresses of individual bytes.

Figure 1-2. Example: SPIDELAY – 0xFFF7F448

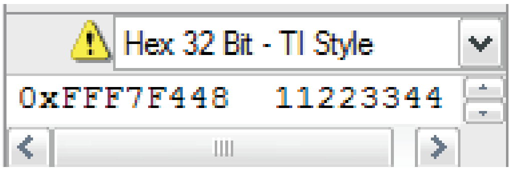
31	24	23	16
C2TDELAY[7:0]		T2CDELAY[7:0]	
Byte 3 - 0xFFF7F44B		Byte 2 - 0xFFF7F44A	
15	8	7	0
T2EDELAY[7:0]		C2EDELAY[7:0]	
Byte 1 - 0xFFF7F449		Byte 0 - 0xFFF7F448	

32-bit accesses to this register should use the lowest address, that is, 0xFFF7F448. Writing 0x11223344 to address 0xFFF7F448 shows the following when viewing the memory in 8-bit and 32-bit modes.



Hex 8 Bit - TI Style Hex

```
0xFFF7F448   44
0xFFF7F449   33
0xFFF7F44A   22
0xFFF7F44B   11
```



Hex 32 Bit - TI Style

```
0xFFF7F448   11223344
```

As such, the headers provided as part of HALCoGen do take the endianness into account and provide header structures that are agnostic to endianness. This is achieved by using C directives for the compiler that make use of the compile options configured for the project by the user (__little_endian__ used in Code Composer Studio codegen tools). This directive may need to be adapted for other compilers.

```
#ifdef __little_endian__
    char  C2EDELAY      :  8U;  /**lt; 0xF448: CS to ENA        */
    char  T2EDELAY      :  8U;  /**lt; 0xF449: Transmit to ENA  */
    char  T2CDELAY      :  8U;  /**lt; 0xF44A: Transmit to CS   */
    char  C2TDELAY      :  8U;  /**lt; 0xF44B: CS to Transmit   */
#else
    char  C2TDELAY      :  8U;  /**lt; 0xF448: CS to Transmit   */
    char  T2CDELAY      :  8U;  /**lt; 0xF449: Transmit to CS   */
    char  T2EDELAY      :  8U;  /**lt; 0xF44A: Transmit to ENA  */
    char  C2EDELAY      :  8U;  /**lt; 0xF44B: CS to ENA        */
```

Architecture

This chapter consists of five sections. The first section describes specific aspects of the device architecture. The second section describes the clocking structure of the microcontrollers. The third section gives an overview of the device memory organization. The fourth section details exceptions on the device, and the last section describes the system and peripheral control registers of the microcontroller.

Topic **Page**

2.1 Introduction 101
2.2 Memory Organization 105
2.3 Exceptions 117
2.4 Clocks 120
2.5 System and Peripheral Control Registers 130

2.1 Introduction

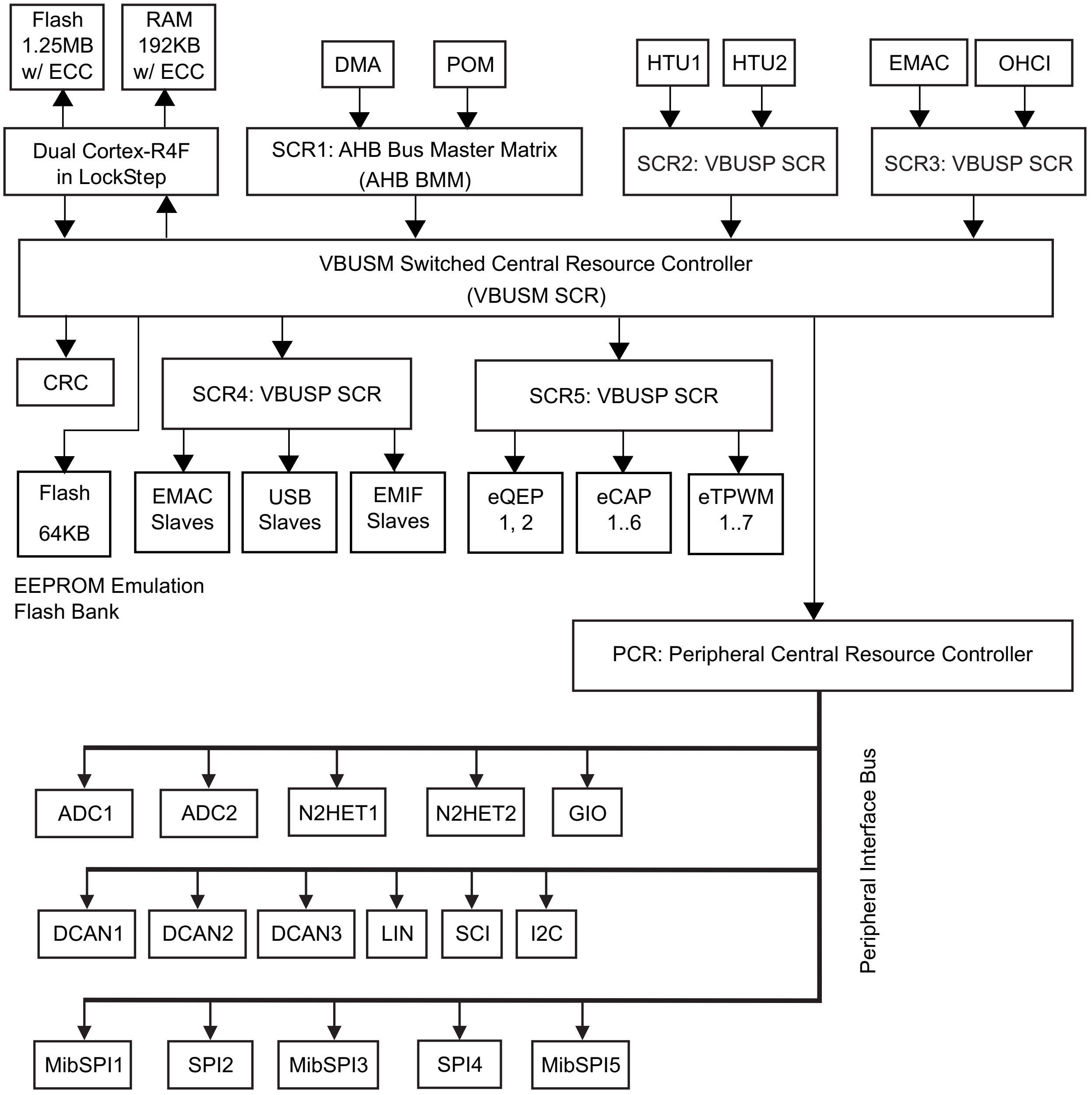
The RM46x family of microcontrollers is based on the Texas Instruments TMS570 Architecture. This chapter describes specific aspects of the architecture as applicable to the RM46x family of microcontrollers.

2.1.1 Architecture Block Diagram

The RM46x microcontrollers are based on the TMS570 Platform architecture, which defines the interconnect between the bus masters and the bus slaves.

Figure 2-1 shows a high-level architectural block diagram for the superset microcontroller.

Figure 2-1. Architectural Block Diagram



2.1.2 Definitions of Terms

Table 2-1 provides a definition of terms used in the architectural block diagram.

Table 2-1. Definition of Terms

Acronym / Term	Full Form	Description
ADCx	Analog-to-Digital Converter	The ADC uses the Successive Approximation Register architecture. It features a selectable 10-bit or 12-bit resolution. The ADC module also includes a RAM to hold the conversion results. A digital logic wrapper manages accesses to the control and status registers. There are two ADC modules on this device.
CRC	Cyclic Redundancy Checker	The CRC module provides two channels to perform background signature verification on any memory region. It also supports maximum-length Parallel Signature Analysis (PSA) based on a 64-bit primitive polynomial. The CRC module is a bus slave in this device.
DAP	Debug Access Port	The DAP allows a tool such as a debugger to read from or write to any region in the device memory-map. The DAP is a bus master in this device.
DCANx	Controller Area Network controller	The DCAN supports the CAN 2.0B protocol standard and uses a serial, multi-master communication protocol that efficiently supports distributed real-time control with robust communication rates of up to 1 megabit per second (Mbps). The DCAN is ideal for applications operating in noisy and harsh environments (for example, automotive and industrial fields) that require reliable serial communication or multiplexed wiring.
DMA	Direct Memory Access	The DMA module is used for transferring 8-, 16-, 32- or 64-bit data across the entire device memory-map. The DMA module is one of the bus masters on the device. That is, it can initiate a read or a write transaction.
eCAP	Enhanced Capture Module	The enhanced Capture (eCAP) module is essential in systems where accurate timing of external events is important.
eFuse	Electronically Programmable Fuse controller	Electrically programmable fuses (eFuses) are used to configure the device after deassertion of PORRST. The eFuse values are read and loaded into internal registers as part of the power-on-reset sequence. The eFuse values are protected with Single-Bit Error Correction Double-Bit Error Detection (SECDED) codes. These fuses are programmed during the initial factory test of the device. The eFuse controller is designed so that the state of the eFuses cannot be changed once the device is packaged.
ePWM	Enhanced Pulse Width Modulator	The enhanced pulse width modulator (ePWM) peripheral is a key element in controlling many of the power electronic systems found in both commercial and industrial equipments. These systems include digital motor control, switch mode power supply control, uninterruptible power supplies (UPS), and other forms of power conversion. The ePWM peripheral performs a digital to analog (DAC) function, where the duty cycle is equivalent to a DAC analog value; it is sometimes referred to as a Power DAC.
eQEP	Enhanced Quadrature Encoder Pulse Module	The enhanced quadrature encoder pulse (eQEP) module is used for direct interface with a linear or rotary incremental encoder to get position, direction, and speed information from a rotating machine for use in a high-performance motion and position-control system.
ECC	Error Correction Code	This is a code that is used by the Single-Bit Error Correction Double-Bit Error Detection (SECDED) logic inside the two Cortex-R4F processors (CPUs). There are 8 bits of ECC for every 64 bits of data accessed from the CPU tightly-coupled memories (flash and RAM).
EMAC	Ethernet Media Access Controller	The EMAC has a dedicated DMA-type component that is used to transfer data to / from the EMAC descriptor memory from / to another memory in the device memory-map. This DMA-type component of the EMAC is a bus master in this device.
EMAC slaves	Ethernet Media Access Controller slave ports	There are four EMAC slaves: 1. EMAC Control Module: this provides an interface between the EMAC and MDIO modules and the bus masters. It also includes 8KB of RAM to hold EMAC packet buffer descriptors. 2. EMAC: The EMAC module interfaces to the other devices on the Ethernet Network using the Media Independent Interface (MII) or Reduced Media Independent Interface (RMII). 3. Management Data Input / Output (MDIO): The MDIO module is used to manage the physical layer (PHY) device connected to the EMAC module. 4. Communications Port Programming Interface (CPPI): This is the 8KB of RAM used to hold the EMAC packet buffer descriptors.

Table 2-1. Definition of Terms (continued)

Acronym / Term	Full Form	Description
EEPROM Emulation Flash Bank	Emulated Electrically Erasable Programmable Read-Only Memory	This is a flash bank that is dedicated for use as an emulated EEPROM. This device supports 64KB of flash for emulated EEPROM.
EMIF slaves	External Memory Interface slave ports	There are five EMIF slaves:- External SDRAM memory: EMIF chip select 0- External asynchronous memories: EMIF chip selects 2, 3 and 4- EMIF module control and status registers
GIO	General-purpose Input/Output	The GIO module allows up to 16 terminals to be used as general-purpose Input or Output. Each of these are also capable of generating an interrupt to the CPU.
HTUx	High-end timer Transfer Unit	The HTU is a dedicated transfer unit for the New Enhanced High-End Timer module. The HTU has a native interface to the N2HET RAM, and is used to transfer data to and from the N2HET RAM from and to another region in the device memory-map. There is one HTU per N2HET module, so that there are 2 HTU modules on the device. The HTUx are bus masters in this device.
I2C	Inter-Integrated Circuit controller	The I2C module is a multi-master communication module providing an interface between the device and an I2C-compatible device via the I2C serial bus. The I2C supports both 100 Kbps and 400 Kbps speeds.
LIN	Local Interconnect Network controller	The LIN module supports the Local Interconnect standard revision 2.1 and can be used as a UART in full-duplex mode using the standard Non-Return-to-Zero (NRZ) format.
Lockstep	–	This is the mode of operation of the dual ARM Cortex-R4F CPUs. The outputs of the two CPUs are compared on each CPU clock cycle. Any miscompare is flagged as an error of the highest severity level.
MibSPIx	Multi-Buffered Serial Peripheral Interface	The MibSPIx modules also support the standard SPI communication protocol. The transfers are all grouped into transfer chunks called “transfer groups”. These transfer groups are made up of one ore more buffers in the MibSPIx RAM. The RAM is used to hold the control information and data to be transmitted, as well as the status information and data that is received. There are three MibSPI modules in this device.
N2HETx	New Enhanced High-End Timer	The N2HET is an advanced intelligent timer that provides sophisticated timing functions for real-time applications. The timer is software-controlled, using a reduced instruction set, with a specialized timer micromachine and an attached I/O port. The N2HET can be used for pulse width modulated outputs, capture or compare inputs, or general-purpose I/O.
PCR	Peripheral Central Resource controller	The PCR manages the accesses to the peripheral registers and peripheral memories. It provides a global reset for all the peripherals. It also supports the capability to selectively enable or disable the clock for each peripheral individually. The PCR also manages the accesses to the system module registers required to configure the device’s clocks, interrupts, and so on. The system module registers also include status flags for indicating exception conditions – resets, aborts, errors, interrupts.
POM	Parameter Overlay Module	The parameter overlay module redirects accesses to a programmable region in flash memory (read-only) to a RAM memory, either on-chip or via the external memory interface (EMIF). This allows a user to evaluate the impact of changing values of constants stored in the flash memory without actually having to erase and reprogram the flash. The POM is also a bus master in this device.
SCI	Serial Communication Interface	The SCI module supports the standard UART in full-duplex mode using the standard Non-Return-to-Zero (NRZ) format.
SCR1: AHB BMM	AMBA High-performance Bus Matrix Module	The DMA, POM and DAP all act as masters on the AMBA High-performance Bus (AHB). The BMM arbitrates between concurrent accesses by these masters using a fixed priority scheme. The modules in descending order of priority are POM —> DMA —> DAP.
SCR2: VBUSP SCR	VBUSP Switched Central Resource Controller	The SCR2 arbitrates between concurrent accesses by the HTU1 and HTU2. A round-robin priority scheme is used between the HTU1 and HTU2.
SCR3: VBUSP SCR	VBUSP Switched Central Resource Controller	The SCR3 arbitrates between concurrent accesses by the EMAC and another module that is not available in this configuration of the device.
SCR4: VBUSP SCR	VBUSP Switched Central Resource Controller	The SCR4 is used to decode the accesses to the bus slaves for the EMAC and EMIF modules. SCR4 is a bus slave in this device.
SPIx	Serial Peripheral Interface	The SPIx module provides a clocked serial communication interface for reliable communication between the device and other serial devices with the standard SPI interface. There are two SPI modules on this device.

Table 2-1. Definition of Terms (continued)

Acronym / Term	Full Form	Description
USB	Universal Serial Bus	This device provides several varieties of USB functionality, including: • One full-speed USB device port compatible with the USB Specification Revision 2.0 and USB Specification Revision 1.1 • Two USB host ports compatible with USB Specification Revision 2.0, which is based on the OHCI Specification For USB Release 1.0
VBUSM SCR	VBUSM Switched Central Resource Controller	This is the main device SCR. It arbitrates between the accesses from multiple bus masters to the bus slaves using a round robin priority scheme.

2.1.3 Bus Master / Slave Access Privileges

This device implements some restrictions on the bus slave access privileges, see Table 2-2, in order to improve the overall throughput of the interconnect shown in Figure 2-1.

Table 2-2. Bus Master / Slave Access Privileges

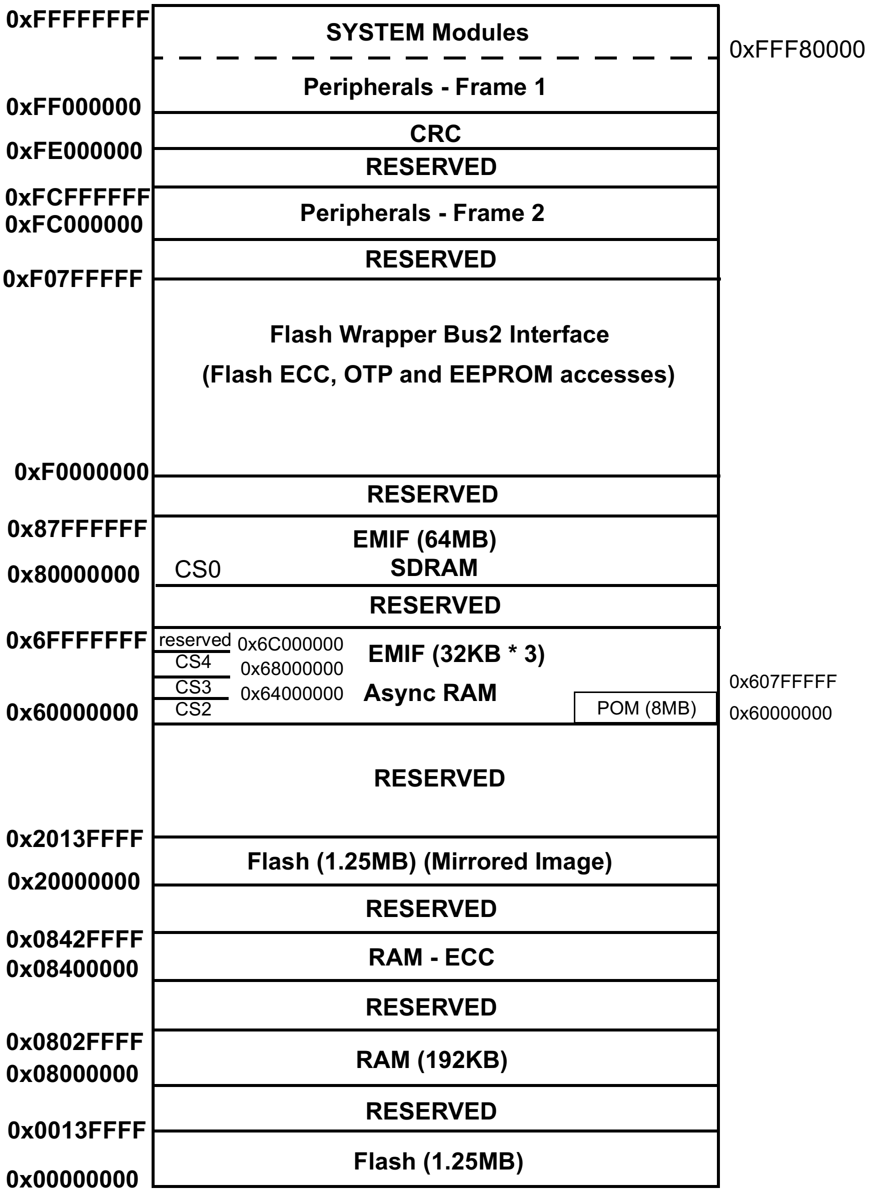
			Bus Slaves Being Accessed				
Masters	Master ID	Access Mode	EEPROM Bank, ECC Bits, OTP Regions	Non-CPU Accesses to CPU Flash and RAM	CRC Module	EMAC, EMIF, and USB Slaves	PCR Modules
CPU Read	0	User/Privilege	Allowed	Allowed	Allowed	Allowed	Allowed
CPU Write	1	User/Privilege	Not allowed	Allowed	Allowed	Allowed	Allowed
POM	2	User	Allowed	Allowed	Allowed	Allowed	Allowed
DMA	3	User	Allowed	Allowed	Allowed	Allowed	Allowed
DAP	5	Privilege	Allowed	Allowed	Allowed	Allowed	Allowed
HTU1	6	Privilege	Not allowed	Allowed	Allowed	Allowed	Allowed
HTU2	8	Privilege	Not allowed	Allowed	Allowed	Allowed	Allowed
EMAC	9	User	Not allowed	Allowed	Not allowed	Allowed	Not allowed
OHCI	10	User	Not allowed	Allowed	Not allowed	Allowed	Not allowed

2.2 Memory Organization

2.2.1 Memory-Map Overview

The Cortex-R4F uses a 32-bit address bus, giving it access to a memory space of 4GB. This space is divided into several regions, each addressed by different memory selects. Figure 2-2 shows the memory-map of the microcontroller.

Figure 2-2. Memory-Map



The main flash instruction memory is addressed starting at 0x00000000 by default. This is also the reset vector location – the ARM Cortex-R4F processor core starts execution from the reset vector address of 0x00000000 whenever the core gets reset.

The CPU data RAM is addressed starting at 0x08000000 by default.

The device also supports the swapping of the CPU instruction memory (flash) and data memory (RAM). This can be done by configuring the MEM SWAP field of the Bus Matrix Module Control Register 1 (BMMCR1).

After swapping, the data RAM is accessed starting from 0x00000000 and the RAM ECC locations are accessed starting from 0x00400000. The flash memory is now accessed starting from 0x08000000.

2.2.2 Memory-Map Table

The control and status registers for each module are mapped within the CPU's 4GB memory space. Some modules also have associated memories, which are also mapped within this space.

Table 2-3 shows the starting and ending addresses of each module's register frame and any associated memory. The table also shows the response generated by the module or the interconnect whenever an access is made to an unimplemented location inside the register or memory frame.

Table 2-3. Module Registers / Memories Memory-Map

Name	Memory Select	Frame Address		Frame Size	Actual Size	Response for Access to Unimplemented Location in Frame
		Start	End			
CPU Tightly-Coupled Memories						
TCM Flash	CS0	0x0000_0000	0x00FF_FFFF	16MB	1.25MB	Abort
TCM RAM + RAM ECC	CSRAM0	0x0800_0000	0x0BFF_FFFF	64MB	192KB	
Mirrored Flash	Flash mirror frame	0x2000_0000	0x20FF_FFFF	16MB	1.25MB	
External Memory Accesses						
EMIF Chip Select 2 (asynchronous)	EMIF select 2	0x6000_0000	0x63FF_FFFF	64MB	16MB	Access to "Reserved" space will generate Abort
EMIF Chip Select 3 (asynchronous)	EMIF select 3	0x6400_0000	0x67FF_FFFF	64MB	16MB	
EMIF Chip Select 4 (asynchronous)	EMIF select 4	0x6800_0000	0x6BFF_FFFF	64MB	16MB	
EMIF Chip Select 0 (synchronous)	EMIF select 0	0x8000_0000	0x87FF_FFFF	128MB	128MB	
Flash Bus2 Interface: OTP, ECC, EEPROM Bank						
Customer OTP, TCM Flash Bank		0xF000_0000	0xF000_1FFF	8KB	4KB	Abort
Customer OTP, EEPROM Bank		0xF000_E000	0xF000_FFFF	8KB	4KB	
Customer OTP–ECC, TCM Flash Bank		0xF004_0000	0xF004_03FF	1KB	512B	
Customer OTP–ECC, EEPROM Bank		0xF004_1C00	0xF004_1FFF	1KB	512B	
TI OTP, TCM Flash Bank		0xF008_0000	0xF008_1FFF	8KB	4KB	
TI OTP, EEPROM Bank		0xF008_E000	0xF008_FFFF	8KB	4KB	
TI OTP–ECC, TCM Flash Bank		0xF00C_0000	0xF00C_03FF	1KB	512B	
TI OTP–ECC, EEPROM Bank		0xF00C_1C00	0xF00C_1FFF	1KB	512B	
EEPROM Bank–ECC		0xF010_0000	0xF013_FFFF	256KB	8KB	
EEPROM Bank		0xF020_0000	0xF03F_FFFF	2MB	64KB	
Flash Data Space ECC		0xF040_0000	0xF04F_FFFF	1MB	160KB	
EMAC and EMIF Slaves						
CPPI Memory Slave (Ethernet RAM)		0xFC52_0000	0xFC52_1FFF	8KB	8KB	Abort
CPGMAC Slave (Ethernet Slave)		0xFCF7_8000	0xFCF7_87FF	2KB	2KB	No error
CPGMACSS Wrapper (Ethernet Wrapper)		0xFCF7_8800	0xFCF7_88FF	256B	256B	No error

Table 2-3. Module Registers / Memories Memory-Map (continued)

Name	Memory Select	Frame Address		Frame Size	Actual Size	Response for Access to Unimplemented Location in Frame
		Start	End			
Ethernet MDIO Interface		0xFCF7_8900	0xFCF7_89FF	256B	256B	No error
W2FC (USB device controller registers)		0xFCF7_8A00	0xFCF7_8A7F	128B	128B	Abort
OHCI (USB Host controller registers)		0xFCF7_8B00	0xFCF7_8BFF	256B	256B	Abort
ePWM1		0xFCF7_8C00	0xFCF7_8CFF	256B	256B	Abort
ePWM2		0xFCF7_8D00	0xFCF7_8DFF	256B	256B	Abort
ePWM3		0xFCF7_8E00	0xFCF7_8EFF	256B	256B	Abort
ePWM4		0xFCF7_8F00	0xFCF7_8FFF	256B	256B	Abort
ePWM5		0xFCF7_9000	0xFCF7_90FF	256B	256B	Abort
ePWM6		0xFCF7_9100	0xFCF7_91FF	256B	256B	Abort
ePWM7		0xFCF7_9200	0xFCF7_92FF	256B	256B	Abort
eCAP1		0xFCF7_9300	0xFCF7_94FF	256B	256B	Abort
eCAP2		0xFCF7_9400	0xFCF7_95FF	256B	256B	Abort
eCAP3		0xFCF7_9500	0xFCF7_96FF	256B	256B	Abort
eCAP4		0xFCF7_9600	0xFCF7_97FF	256B	256B	Abort
eCAP5		0xFCF7_9700	0xFCF7_98FF	256B	256B	Abort
eCAP6		0xFCF7_9800	0xFCF7_99FF	256B	256B	Abort
eQEP1		0xFCF7_9900	0xFCF7_9AFF	256B	256B	Abort
eQEP2		0xFCF7_9A00	0xFCF7_9BFF	256B	256B	Abort
EMIF Registers		0xFCFF_E800	0xFCFF_E8FF	256B	256B	Abort
Cyclic Redundancy Checker (CRC) Module Register Frame						
CRC	CRC frame	0xFE00_0000	0xFEFF_FFFF	16MB	512B	Accesses above 0x200 generate abort.
Peripheral Memories						
MIBSPI5 RAM	PCS[5]	0xFF0A_0000	0xFF0B_FFFF	128KB	2KB	Abort for accesses above 2KB
MIBSPI3 RAM	PCS[6]	0xFF0C_0000	0xFF0D_FFFF	128KB	2KB	Abort for accesses above 2KB
MIBSPI1 RAM	PCS[7]	0xFF0E_0000	0xFF0F_FFFF	128KB	2KB	Abort for accesses above 2KB
DCAN3 RAM	PCS[13]	0xFF1A_0000	0xFF1B_FFFF	128KB	2KB	Wrap around for accesses to unimplemented address offsets lower than 0x7FF. Abort generated for accesses beyond offset 0x800.
DCAN2 RAM	PCS[14]	0xFF1C_0000	0xFF1D_FFFF	128KB	2KB	Wrap around for accesses to unimplemented address offsets lower than 0x7FF. Abort generated for accesses beyond offset 0x800.
DCAN1 RAM	PCS[15]	0xFF1E_0000	0xFF1F_FFFF	128KB	2KB	Wrap around for accesses to unimplemented address offsets lower than 0x7FF. Abort generated for accesses beyond offset 0x800.

Table 2-3. Module Registers / Memories Memory-Map (continued)

Name	Memory Select	Frame Address		Frame Size	Actual Size	Response for Access to Unimplemented Location in Frame
		Start	End			
MIBADC2 RAM	PCS[29]	0xFF3A_0000	0xFF3B_FFFF	128KB	8KB	Wrap around for accesses to unimplemented address offsets lower than 0x1FFF. Abort generated for accesses beyond 0x1FFF.
MIBADC1 RAM	PCS[31]	0xFF3E_0000	0xFF3F_FFFF	128KB	8KB	Wrap around for accesses to unimplemented address offsets lower than 0x1FFF. Abort generated for accesses beyond 0x1FFF.
N2HET2 RAM	PCS[34]	0xFF44_0000	0xFF45_FFFF	128KB	16KB	Wrap around for accesses to unimplemented address offsets lower than 0x3FFF. Abort generated for accesses beyond 0x3FFF.
N2HET1 RAM	PCS[35]	0xFF46_0000	0xFF47_FFFF	128KB	16KB	Wrap around for accesses to unimplemented address offsets lower than 0x3FFF. Abort generated for accesses beyond 0x3FFF.
HET TU2 RAM	PCS[38]	0xFF4C_0000	0xFF4D_FFFF	128KB	1KB	Abort
HET TU1 RAM	PCS[39]	0xFF4E_0000	0xFF4F_FFFF	128KB	1KB	Abort
Debug Components						
CoreSight Debug ROM	CSCS0	0xFFA0_0000	0xFFA0_0FFF	4KB	4KB	Reads return zeros, writes have no effect
Cortex-R4F Debug	CSCS1	0xFFA0_1000	0xFFA0_1FFF	4KB	4KB	Reads return zeros, writes have no effect
POM	CSCS4	0xFFA0_4000	0xFFA0_4FFF	4KB	4KB	Abort
Peripheral Control Registers						
HTU1	PS[22]	0xFFF7_A400	0xFFF7_A4FF	256B	256B	Reads return zeros, writes have no effect
HTU2	PS[22]	0xFFF7_A500	0xFFF7_A5FF	256B	256B	Reads return zeros, writes have no effect
N2HET1	PS[17]	0xFFF7_B800	0xFFF7_B8FF	256B	256B	Reads return zeros, writes have no effect
N2HET2	PS[17]	0xFFF7_B900	0xFFF7_B9FF	256B	256B	Reads return zeros, writes have no effect
GIO	PS[16]	0xFFF7_BC00	0xFFF7_BCFF	256B	256B	Reads return zeros, writes have no effect
MIBADC1	PS[15]	0xFFF7_C000	0xFFF7_C1FF	512B	512B	Reads return zeros, writes have no effect
MIBADC2	PS[15]	0xFFF7_C200	0xFFF7_C3FF	512B	512B	Reads return zeros, writes have no effect
I2C	PS[10]	0xFFF7_D400	0xFFF7_D4FF	256B	256B	Reads return zeros, writes have no effect
DCAN1	PS[8]	0xFFF7_DC00	0xFFF7_DDFF	512B	512B	Reads return zeros, writes have no effect
DCAN2	PS[8]	0xFFF7_DE00	0xFFF7_DFFF	512B	512B	Reads return zeros, writes have no effect

Table 2-3. Module Registers / Memories Memory-Map (continued)

Name	Memory Select	Frame Address		Frame Size	Actual Size	Response for Access to Unimplemented Location in Frame
		Start	End			
DCAN3	PS[7]	0xFFF7_E000	0xFFF7_E1FF	512B	512B	Reads return zeros, writes have no effect
LIN	PS[6]	0xFFF7_E400	0xFFF7_E4FF	256B	256B	Reads return zeros, writes have no effect
SCI	PS[6]	0xFFF7_E500	0xFFF7_E5FF	256B	256B	Reads return zeros, writes have no effect
MibSPI1	PS[2]	0xFFF7_F400	0xFFF7_F5FF	512B	512B	Reads return zeros, writes have no effect
SPI2	PS[2]	0xFFF7_F600	0xFFF7_F7FF	512B	512B	Reads return zeros, writes have no effect
MibSPI3	PS[1]	0xFFF7_F800	0xFFF7_F9FF	512B	512B	Reads return zeros, writes have no effect
SPI4	PS[1]	0xFFF7_FA00	0xFFF7_FBFF	512B	512B	Reads return zeros, writes have no effect
MibSPI5	PS[0]	0xFFF7_FC00	0xFFF7_FDFF	512B	512B	Reads return zeros, writes have no effect
System Modules Control Registers and Memories						
DMA RAM	PPCS0	0xFFF8_0000	0xFFF8_0FFF	4KB	4KB	Abort
VIM RAM	PPCS2	0xFFF8_2000	0xFFF8_2FFF	4KB	1KB	Wrap around for accesses to unimplemented address offsets lower than 0x3FF. Abort generated for accesses beyond 0x3FF.
Flash Wrapper	PPCS7	0xFFF8_7000	0xFFF8_7FFF	4KB	4KB	Abort
eFuse Farm Controller	PPCS12	0xFFF8_C000	0xFFF8_CFFF	4KB	4KB	Abort
Power Management Module (PMM)	PPSE0	0xFFFF_0000	0xFFFF_01FF	512B	512B	Abort
Test Controller (FMTM)	PPSE1	0xFFFF_0400	0xFFFF_07FF	1KB	1KB	Reads return zeros, writes have no effect
PCR registers	PPS0	0xFFFF_E000	0xFFFF_E0FF	256B	256B	Reads return zeros, writes have no effect
System Module - Frame 2 (see platform architecture specification)	PPS0	0xFFFF_E100	0xFFFF_E1FF	256B	256B	Reads return zeros, writes have no effect
PBIST	PPS1	0xFFFF_E400	0xFFFF_E5FF	512B	512B	Reads return zeros, writes have no effect
STC	PPS1	0xFFFF_E600	0xFFFF_E6FF	256B	256B	Reads return zeros, writes have no effect
IOMM Multiplexing Control Module	PPS2	0xFFFF_EA00	0xFFFF_EBFF	512B	512B	Generates address error interrupt, if enabled
DCC1	PPS3	0xFFFF_EC00	0xFFFF_ECFF	256B	256B	Reads return zeros, writes have no effect
DMA	PPS4	0xFFFF_F000	0xFFFF_F3FF	1KB	1KB	Reads return zeros, writes have no effect
DCC2	PPS5	0xFFFF_F400	0xFFFF_F4FF	256B	256B	Reads return zeros, writes have no effect
ESM	PPS5	0xFFFF_F500	0xFFFF_F5FF	256B	256B	Reads return zeros, writes have no effect
CCMR4	PPS5	0xFFFF_F600	0xFFFF_F6FF	256B	256B	Reads return zeros, writes have no effect
RAM ECC even	PPS6	0xFFFF_F800	0xFFFF_F8FF	256B	256B	Reads return zeros, writes have no effect

Table 2-3. Module Registers / Memories Memory-Map (continued)

Name	Memory Select	Frame Address		Frame Size	Actual Size	Response for Access to Unimplemented Location in Frame
		Start	End			
RAM ECC odd	PPS6	0xFFFF_F900	0xFFFF_F9FF	256B	256B	Reads return zeros, writes have no effect
RTI + DWWD	PPS7	0xFFFF_FC00	0xFFFF_FCFF	256B	256B	Reads return zeros, writes have no effect
VIM Parity	PPS7	0xFFFF_FD00	0xFFFF_FDFF	256B	256B	Reads return zeros, writes have no effect
VIM	PPS7	0xFFFF_FE00	0xFFFF_FEFF	256B	256B	Reads return zeros, writes have no effect
System Module - Frame 1	PPS7	0xFFFF_FF00	0xFFFF_FFFF	256B	256B	Reads return zeros, writes have no effect

2.2.3 Flash on Microcontrollers

The RM46x microcontrollers support up to 1.25MB of flash for use as program memory. The microcontrollers also support a separate 64KB flash bank for use as emulated EEPROM.

2.2.3.1 Flash Bank Sectoring Configuration

The bank is divided into multiple sectors. A flash sector is the smallest region in the flash bank that must be erased. The sectoring configuration of each flash bank is shown in Table 2-4.

Table 2-4. Flash Memory Banks and Sectors [(1)]

Memory Arrays (or Banks)	Block Number	Sector Number	Segment	Low Address	High Address
BANK0 (1.25MBytes)	0	0	16K Bytes	0x0000_0000	0x0000_3FFF
		1	16K Bytes	0x0000_4000	0x0000_7FFF
		2	16K Bytes	0x0000_8000	0x0000_BFFF
		3	16K Bytes	0x0000_C000	0x0000_FFFF
		4	16K Bytes	0x0001_0000	0x0001_3FFF
		5	16K Bytes	0x0001_4000	0x0001_7FFF
		6	32K Bytes	0x0001_8000	0x0001_FFFF
		7	128K Bytes	0x0002_0000	0x0003_FFFF
	1	8	128K Bytes	0x0004_0000	0x0005_FFFF
		9	128K Bytes	0x0006_0000	0x0007_FFFF
	2	10	128K Bytes	0x0008_0000	0x0009_FFFF
		11	128K Bytes	0x000A_0000	0x000B_FFFF
	3	12	128K Bytes	0x000C_0000	0x000D_FFFF
		13	128K Bytes	0x000E_0000	0x000F_FFFF
	4	14	128K Bytes	0x0010_0000	0x0011_FFFF
		15	128K Bytes	0x0012_0000	0x0013_FFFF
BANK7 (64KBytes) for EEPROM emulation	0	0	16K Bytes	0xF020_0000	0xF020_3FFF
	1	1	16K Bytes	0xF020_4000	0xF020_7FFF
	2	2	16K Bytes	0xF020_8000	0xF020_BFFF
	3	3	16K Bytes	0xF020_C000	0xF020_FFFF

[(1)]

- The Flash banks are 144-bit wide bank with ECC support.
- The flash bank7 can be programmed while executing code from flash bank0.
- Code execution is not allowed from flash bank7.

Refer to the device datasheet for electrical and timing specifications related to the flash module.

2.2.3.2 ECC Protection for Flash Accesses

The RM46x microcontrollers protect all accesses to the on-chip flash memory by dedicated Single-Bit Error Correction Double-Bit Error Detection (SECDED) logic.

The access to the program memory – flash bank 0, is protected by SECDED logic implemented inside the ARM Cortex-R4F CPU. Accesses to the EEPROM emulation flash bank (bank 7) are protected by dedicated SECDED logic inside the digital interface to the flash banks.

Both the SECDED logic implementations use Error Correction Codes (ECC) for correcting single-bit errors and for detecting double-bit errors in the values read from the flash arrays. There is an 8-bit ECC for every 64 bits of data. The ECC for the flash memory contents needs to be calculated by an external tool such as nowECC. The ECC can then be programmed into the flash array along with the actual application code.

The ECC for the flash array is stored in the flash itself, and is mapped to a region starting at 0xF0400000 for the main flash bank 0, and to a region starting at 0xF0100000 for the EEPROM emulation flash bank 7.

NOTE: ECC Protection Not Enabled By Default

The SECDED logic inside the CPU is not enabled by default and must be enabled by the application.

Code Example for Enabling ECC Protection for Main Flash Accesses:

The following code example can be used to enable the ECC protection for accesses to the main flash array:

```
MRC p15, #0, r1, c1, c0, #1
ORR r1, r1, #0x02000000       ;Enable ECC checking for ATCM
DMB
MCR p15, #0, r1, c1, c0, #1
```

The ECC protection for accesses to the EEPROM emulation flash bank can be enabled by writing 0xA to the EDACEN field of the flash module’s Error Correction Control Register 1 (FEDACCTRL1). See Chapter 5 for more details.

When the CPU detects an ECC single-, or double-bit error on a read from the flash memory, it signals this on a dedicated “*Event*” bus. This event bus signaling is also not enabled by default and must be enabled by the application. The following code example can be used to enable the CPU event signaling:

```
MRC p15,#0,r1,c9,c12,#0       ;Enabling Event monitor states
ORR r1, r1, #0x00000010
MCR p15,#0,r1,c9,c12,#0       ;Set 4th bit ('X') of PMNC register
MRC p15,#0,r1,c9,c12,#0
```

The digital logic that interfaces the ARM Cortex-R4F CPU to the flash banks captures the ECC error events signaled by the CPU, and in turn generates error signals that are input to the central Error Signaling Module (ESM).

2.2.4 On-Chip SRAM

Several SRAM modules are implemented on the device to support the functionality of the modules included.

Reads from the CPU data RAM are protected by ECC calculated inside the CPU. Reads from all other memories are protected by configurable odd or even parity that is evaluated in parallel with the actual read.

The RM46x microcontrollers are targeted towards safety-critical applications, and it is critical for any failures in the on-chip SRAM modules to be identified before these modules are used for safety-critical functions. These microcontrollers support a Programmable Built-In Self-Test (PBIST) mechanism that is used to test each on-chip SRAM module for faults. The PBIST is usually run on device start-up as it is a destructive test and all contents of the tested SRAM module are overwritten during the test.

The microcontrollers also support a hardware-based auto-initialization of on-chip SRAM modules. This process also takes into account the read protection scheme implemented for each SRAM module – ECC or parity.

TI recommends that the PBIST routines be executed on the SRAM modules prior to the auto-initialization. The following section describe these two processes.

2.2.4.1 PBIST RAM Grouping and Algorithm Mapping For On-Chip SRAM Modules

Table 2-5 shows the groupings of the various on-chip memories for PBIST. It also lists the memory types and their assigned RAM Group Select (RGS) and Return Data Select (RDS). See Chapter 7 for more details on the usage of the RGS and RDS information.

Table 2-5. PBIST Memory Grouping

Memory	RAM Group #	Memory Type	RGS	RDS
PBIST_ROM	1	ROM	1	0
STC_ROM	2	ROM	2	0
DCAN1	3	Dual-port	3	0 .. 5
DCAN2	4	Dual-port	4	0 .. 5
DCAN3	5	Dual-port	5	0 .. 5
ESRAM1	6	Single-port	6	0/1 .. 4
MIBSPI1	7	Dual-port	7	0 .. 3
MIBSPI3	8	Dual-port	7	4 .. 7
MIBSPI5	9	Dual-port	7	8 .. 11
VIM	10	Dual-port	8	0 .. 1
MIBADC1	11	Dual-port	9	0
DMA	12	Dual-port	10	0 .. 5
N2HET1	13	Dual-port	11	0 .. 11
HET TU1	14	Dual-port	12	0 .. 5
MIBADC2	18	Dual-port	15	0
N2HET2	19	Dual-port	16	0 .. 11
HET TU2	20	Dual-port	17	0 .. 5
ESRAM5	21	Single-port	18	0/1 .. 4
ESRAM6	22	Single-port	19	0/1 .. 4
ETHERNET	23	Dual-port	20	0 .. 6
	24	Dual-port		
	25	Single-port		
USB	26	Dual-port	21	0 .. 2
	27	Single-port		

Table 2-6 maps the different algorithms supported in application mode for the RAM groups. The table also lists the background pattern options available for each algorithm.

Table 2-6. PBIST Algorithm Mapping

No.	ALGO Register Value	Algorithm	Memories Tested	Available Background Patterns	Valid RAM Groups	Valid RINFO Register Value
1	0x00000001	triple_read_slow_read	ROM		1,2	0x00000003
2	0x00000002	triple_read_fast_read	ROM		1,2	0x00000003
3	0x00000004	march13n	Dual-port	0x00000000, 0x96699669, 0x0F0F0F0F, 0xAA55AA55, 0xC3C3C3C3	3,4,5,7,8,9,10,11,12,13,14,18,19,20,23,24,26	0x02CE3FDC
4	0x00000008	march13n	Single-port	0x00000000, 0x96699669, 0x0F0F0F0F, 0xAA55AA55, 0xC3C3C3C3	6,21,22,25,27	0x05300020
5	0x00000010	down1A_red	Dual-port	0xFFFFFFFF, 0xAAAAAAAA	3,4,5,7,8,9,10,11,12,13,14,18,19,20,23,24,26	0x02CE3FDC
6	0x00000020	down1A_red	Single-port	0xFFFFFFFF, 0xAAAAAAAA	6,21,22,25,27	0x05300020
7	0x00000040	mapcolumn	Dual-port	0xFFFFFFFF, 0x00000000	3,4,5,7,8,9,10,11,12,13,14,18,19,20,23,24,26	0x02CE3FDC
8	0x00000080	mapcolumn	Single-port	0xFFFFFFFF, 0x00000000	6,21,22,25,27	0x05300020
9	0x00000100	precharge	Dual-port	0xFFFFFFFF, 0x00000000	3,4,5,7,8,9,10,11,12,13,14,18,19,20,23,24,26	0x02CE3FDC
10	0x00000200	precharge	Single-port	0xFFFFFFFF, 0x00000000	6,21,22,25,27	0x05300020
11	0x00000400	dtxn2	Dual-port	0xFFFFFFFF, 0x00000000	3,4,5,7,8,9,10,11,12,13,14,18,19,20,23,24,26	0x02CE3FDC
12	0x00000800	dtxn2	Single-port	0xFFFFFFFF, 0x00000000	6,21,22,25,27	0x05300020
13	0x00001000	pmos_open	Dual-port	0xFFFFFFFF, 0x00000000	3,4,5,7,8,9,11,14,18,20,24,26	0x028A25DC
14	0x00002000	pmos_open	Single-port	0xFFFFFFFF, 0x00000000	6,21,22,25,27	0x05300020
15	0x00004000	pmos_open_slice1	Dual-port	0xFFFFFFFF, 0x00000000	10,12,13,19,23,26	0x00441A00
16	0x00008000	pmos_open_slice2	Dual-port	0xFFFFFFFF, 0x00000000	10,12,13,19,23,26	0x00441A00
17	0x00010000	flip10	Dual-port	0xFFFFFFFF	3,4,5,7,8,9,10,11,12,13,14,18,19,20,23,24,26	0x02CE3FDC
18	0x00020000	flip10	Single-port	0xFFFFFFFF	6,21,22,25,27	0x05300020
19	0x00040000	iddq	Dual-port	0x00000000	3,4,5,7,8,9,10,11,12,13,14,18,19,20,23,24,26	0x02CE3FDC
20	0x00080000	iddq	Single-port	0x00000000	6,21,22,25,27	0x05300020
21	0x00100000	retention	Dual-port	0x00000000	3,4,5,7,8,9,10,11,12,13,14,18,19,20,23,24,26	0x02CE3FDC
22	0x00200000	retention	Single-port	0x00000000	6,21,22,25,27	0x05300020
23	0x00400000	iddq	Dual-port	0xFFFFFFFF	3,4,5,7,8,9,10,11,12,13,14,18,19,20,23,24,26	0x02CE3FDC
24	0x00800000	iddq	Single-port	0xFFFFFFFF	6,21,22,25,27	0x05300020
25	0x01000000	retention	Dual-port	0xFFFFFFFF	3,4,5,7,8,9,10,11,12,13,14,18,19,20,23,24,26	0x02CE3FDC
26	0x02000000	retention	Single-port	0xFFFFFFFF	6,21,22,25,27	0x05300020

Table 2-6. PBIST Algorithm Mapping (continued)

No.	ALGO Register Value	Algorithm	Memories Tested	Available Background Patterns	Valid RAM Groups	Valid RINFO Register Value
27	0x04000000	iddqrowstripe	Dual-port	0x00000000	3,4,5,7,8,9,10,11,12,13,14, 18,19,20,23,24,26	0x02CE3FDC
28	0x08000000	iddqrowstripe	Single-port	0x00000000	6,21,22,25,27	0x05300020
29	0x10000000	iddqrowstripe	Dual-port	0xFFFFFFFF	3,4,5,7,8,9,10,11,12,13,14, 18,19,20,23,24,26	0x02CE3FDC
30	0x20000000	iddqrowstripe	Single-port	0xFFFFFFFF	6,21,22,25,27	0x05300020
31	0x40000000	powerup_invpowerup	Dual-port	0xAAAAAAAA	33,34,35,36,37,38,39,40, 41,42,43,45,46,47,48,49, 50,51	0x0007F7FF
32	0x80000000	powerup_invpowerup	Single-port	0xAAAAAAAA	52,53,54,56,57,58	0x03B80000

NOTE: Recommended Memory Test Algorithm

March13 is the most recommended algorithm for the memory self-test.

For HCLK = 180 MHz, VCLK = 90 MHz, PBIST ROM_CLK = 90 MHz, the March13 algorithm takes 14.02 ms to run on all on-chip SRAMs.

2.2.4.2 Auto-Initialization of On-Chip SRAM Modules

The device system provides the capability to perform a hardware initialization on most memories on the system bus and on the peripheral bus.

The intent of having the hardware initialization is to program the memory arrays with error detection capability to a known state based on their error detection scheme – odd/even parity or ECC. For example, the contents of the CPU data RAM after power-on reset is unknown. A hardware auto-initialization can be started to that there is no ECC error.

NOTE: Effect of ECC or Parity on Memory Auto-Initialization

The ECC or parity should be enabled on the RAMs before hardware auto-initialization starts if parity or ECC is being used.

Auto-Initialization Sequence:

1. Enable the global hardware memory initialization key by programming 0xA into MINITGCR[3:0], the Memory Initialization Key field (MINITGENA) of the Memory Hardware Initialization Global Control Register (MINITGCR) register.
2. Select the module on which the memory hardware initialization has to be performed by programming the appropriate value into the MSINENA(31–0) bits in the MSINENA register. See Table 2-7.
3. If the global auto-initialization scheme is enabled, the corresponding module will initialize its memories based on its memory error checking scheme (even parity or odd parity or ECC).
4. When the memory initialization is complete, the module will signal “memory initialization done”, which sets the corresponding bit in the system module MIDONE field of the MINISTAT register to indicate the completion of its memory initialization.
5. When the memory hardware initialization completes for all modules, (indicated by each module’s MIDONE bit being set), the memory hardware initialization done bit (MINIDONE) is set in the MSTCGSTAT register.

Figure 2-3. Hardware Memory Initialization Protocol

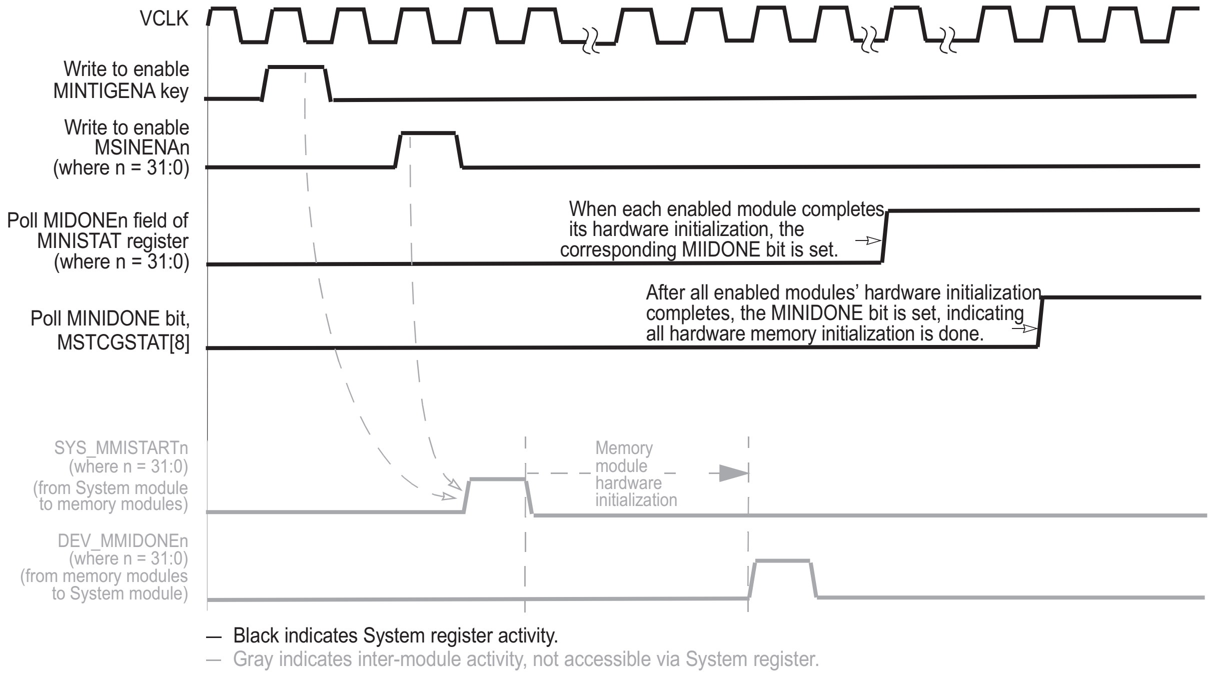

VCLK
Write to enable MINTIGENA key
Write to enable MSINENAn (where n = 31:0)
Poll MIDONEn field of MINISTAT register (where n = 31:0)
When each enabled module completes its hardware initialization, the corresponding MIIDONE bit is set.
Poll MINIDONE bit, MSTCGSTAT[8]
After all enabled modules’ hardware initialization completes, the MINIDONE bit is set, indicating all hardware memory initialization is done.
SYS_MMISTARTn (where n = 31:0) (from System module to memory modules)
Memory module hardware initialization
DEV_MMIDONEn (where n = 31:0) (from memory modules to System module)
— Black indicates System register activity.
— Gray indicates inter-module activity, not accessible via System register.

Table 2-7. Memory Initialization Select Mapping [1][2][3]

Memory	Address Range		MSINENA Register Bit #
	Start	End	
RAM (PD#1)	0x08000000	0x08013FFF	0
RAM (RAM_PD#1)	0x08010000	0x0801FFFF	0
RAM (RAM_PD#2)	0x08020000	0x0802FFFF	0
MIBSPI5 RAM	0xFF0A0000	0xFF0BFFFF	12[4]
MIBSPI3 RAM	0xFF0C0000	0xFF0DFFFF	11[4]
MIBSPI1 RAM	0xFF0E0000	0xFF0FFFFF	7[4]
DCAN3 RAM	0xFF1A0000	0xFF1BFFFF	10
DCAN2 RAM	0xFF1C0000	0xFF1DFFFF	6
DCAN1 RAM	0xFF1E0000	0xFF1FFFFF	5
MIBADC2 RAM	0xFF3A0000	0xFF3BFFFF	14
MIBADC1 RAM	0xFF3E0000	0xFF3FFFFF	8
N2HET2 RAM	0xFF440000	0xFF45FFFF	15
N2HET1 RAM	0xFF460000	0xFF47FFFF	3
HET TU2 RAM	0xFF4C0000	0xFF4DFFFF	16
HET TU1 RAM	0xFF4E0000	0xFF4FFFFF	4
DMA RAM	0xFFF80000	0xFFF80FFF	1
VIM RAM	0xFFF82000	0xFFF82FFF	2
USB Device RAM	RAM is not CPU-Addressable		N/A
Ethernet RAM (CPPI Memory Slave)	0xFC520000	0xFC521FFF	N/A

[1] If ECC protection is enabled for the CPU data RAM, then the auto-initialization process also initializes the corresponding ECC space.

[2] If parity protection is enabled for the peripheral SRAM modules, then the parity bits will also be initialized along with the SRAM modules.

[3] The CPU RAM interface module has separate control bit-fields to select the memory power domain that is to be initialized.

[4] The MibSPIx modules perform an initialization of the transmit and receive RAMs as soon as the multi-buffered mode is enabled. This is independent of whether the application has already initialized these RAMs using the auto-initialization method or not. The MibSPIx modules need to be released from reset by writing 1 to their SPIGCR0 registers before starting auto-initialization on their respective RAMs.

2.3 Exceptions

An “Exception” is an event that makes the processor temporarily halt the normal flow of program execution, for example, to service an interrupt from a peripheral. Before attempting to handle an exception, the processor preserves the critical parts of the current processor state so that the original program can resume when the handler routine has finished.

The following sections describe three exceptions – Reset, Abort and the System Software Interrupts.

For complete details on all exceptions, refer to the ARM® Cortex®-R4F Technical Reference Manual.

2.3.1 Resets

The RM46x microcontroller can be reset by either of the conditions described in Table 2-8. Each reset condition is indicated in the System Exception Status Register (SYSESR).

The device nRST terminal is an I/O. It can be driven low by an external circuit to force a warm reset on the microcontroller. This terminal will be driven low as an output for a minimum of 32 peripheral clock (VCLK) cycles for any device system reset condition. As a result the EXTRST bit in the SYSESR register, SYSESR[3], gets set for all reset conditions listed in Table 2-8. The nRST is driven low as an output for a longer duration during device power-up or whenever the power-on reset (nPORRST) is driven low externally. Refer the device datasheet for the electrical and timing specifications for the nRST.

Table 2-8. Causes of Resets

Condition	Description
Driving nPORRST pin low externally	Cold reset, or power-on reset. This reset signal is typically driven by an external voltage supervisor. This reset is flagged by the PORST bit in the SYSESR register, SYSESR[15].
Voltage Monitor reset	The microcontroller has an embedded voltage monitor that generates a power-on reset when the core voltage gets out of a valid range, or when the I/O voltage falls below a threshold. This reset is also flagged by the PORST bit in the SYSESR register, SYSESR[15]. **Note:** The voltage monitor is not an alternative for an external voltage supervisor.
Driving nRST pin low externally	Warm reset. This reset input is typically used in a system with multiple ICs and which requires that the microcontroller also gets reset whenever the other IC detects a fault condition. This reset is flagged by the EXTRST bit in the SYSESR, register SYSESR[3].
Oscillator failure	This reset is generated by the system module when the clock monitor detects an oscillator fail condition. Whether or not a reset is generated is also dictated by a register in the system module. This reset is flagged by the OSCRST bit in the SYSESR register, SYSESR[14].
Software reset	This reset is generated by the application software writing a 1 to bit 15 of System Exception Control Register (SYSECR) or a 0 to bit 14 of SYSECR. It is typically used by a bootloader type of code that uses a software reset to allow the code execution to branch to the application code once it is programmed into the program memory. This reset is flagged by the SWRST bit in the SYSESR register, SYSESR[4].
CPU reset	This reset is generated by the CPU self-test controller (LBIST) or by changing the memory protection (MMU/MPU) configuration in the CPURSTCR register. This reset is flagged by the CPURST bit in the SYSESR register, SYSESR[5].
Debug reset	The debug logic implemented on the microcontroller allows a system reset to be generated. This reset is flagged by the WDRST bit in the SYSESR register, SYSESR[13].
Watchdog reset	This reset is generated by the digital windowed watchdog (DWWD) module on the microcontroller. The DWWD can generate a reset whenever the watchdog service window is violated. This reset is flagged by the WDRST bit in the SYSESR register, SYSESR[13].

2.3.2 Aborts

When the ARM Cortex-R4F processor's memory system cannot complete a memory access successfully, an abort is generated. An error occurring on an instruction fetch generates a ***prefetch abort***. Errors occurring on data accesses generate ***data aborts***. Aborts are also categorized as being either ***precise*** or ***imprecise***.

2.3.2.1 Prefetch Aborts

When a Prefetch Abort (PABT) occurs, the processor marks the prefetched instruction as invalid, but does not take the exception until the instruction is to be executed. If the instruction is not executed, for example because a branch occurs while it is in the pipeline, the abort does not take place.

All prefetch aborts are precise aborts.

2.3.2.2 Data Aborts

An error occurring on a data memory access can generate a data abort. If the instruction generating the memory access is not executed, for example, because it fails its condition codes, or is interrupted, the data abort does not take place.

A Data Abort (DABT) can be either precise or imprecise, depending on the type of fault that caused it.

2.3.2.3 Precise Aborts

A precise abort, also known as a synchronous abort, is one for which the exception is guaranteed to be taken on the instruction that generated the aborting memory access. The abort handler can use the value in the Link Register (r14_abt) to determine which instruction generated the abort, and the value in the Saved Program Status Register (SPSR_abt) to determine the state of the processor when the abort occurred.

2.3.2.4 Imprecise Aborts

An imprecise abort, also known as an asynchronous abort, is one for which the exception is taken on a later instruction to the instruction that generated the aborting memory access. The abort handler cannot determine which instruction generated the abort, or the state of the processor when the abort occurred. Therefore, imprecise aborts are normally fatal.

Imprecise aborts can be generated by store instructions to normal-type or device-type memory. When the store instruction is committed, the data is normally written into a buffer that holds the data until the memory system has sufficient bandwidth to perform the write access. This gives read accesses higher priority. The write data can be held in the buffer for a long period, during which many other instructions can complete. If an error occurs when the write is finally performed, this generates an imprecise abort.

The RM46x microcontroller architecture applies techniques at the system level to mitigate the impact of imprecise aborts. System level adoption of write status sidebands to the data path allow bus masters to comprehend imprecise aborts, turning them into precise aborts. In cases where this approach is not feasible, buffering bridges or other sources of imprecision may build a FIFO of current transactions such that an imprecise abort may be registered at the point of imprecision for later analysis.

Masking Of Imprecise Aborts:

The nature of imprecise aborts means that they can occur while the processor is handling a different abort. If an imprecise abort generates a new exception in such a situation, the banked link register (R14_abt) and the Saved Processor Status Register (SPSR_abt) values are overwritten. If this occurs before the data is pushed to the stack in memory, the state information about the first abort is lost. To prevent this from happening, the Current Processor Status Register (CPSR) contains a mask bit to indicate that an imprecise abort cannot be accepted, the A-bit. When the A-bit is set, any imprecise abort that occurs is held pending by the processor until the A-bit is cleared, when the exception is actually taken. The A-bit is automatically set when abort, IRQ or FIQ exceptions are taken, and on reset. The application must only clear the A-bit in an abort handler after the state information has either been stacked to memory, or is no longer required.

NOTE: Default Behavior for Imprecise Aborts

The A-bit in the CPSR is set by default. This means that no imprecise abort exception will occur. The application must enable imprecise abort exception generation by clearing the A-bit of the CPSR.

2.3.2.5 Conditions That Generate Aborts

An Abort is generated under the following conditions on the RM46x microcontrollers.

- Access to an illegal address (a nonimplemented address)
- Access to a protected address (protection violation)
- Parity / ECC / Time-out Error on a valid access

Illegal Addresses:

The illegal addresses and the responses to an access to these addresses are defined in Table 2-3.

Addresses Protected By MPU:

Memory access permissions can be configured via the ARM Cortex-R4F processor's Memory Protection Unit (MPU). For more details on the MPU configuration, refer to the ARM® Cortex®-R4F Technical Reference Manual.

A memory access violation is logged as a permission fault in the CPU's fault status register and the virtual address of the access is logged into the CPU's fault address register.

Protection of Peripheral Register and Memory Frames:

Accesses to the peripheral register and memory frames can be protected either by configuring the MPU or by configuring the Peripheral Central Resource (PCR) controller registers.

The PCR module PPROTSETx registers contain one bit per peripheral select quadrant. These bits define the access permissions to the peripheral register frames. If the CPU attempts to write to a peripheral register for which it does not have the correct permissions, a protection violation is detected and an Abort occurs.

Some modules also enforce register updates to only be allowed when the CPU is in a privileged mode of operation. If the CPU writes to these registers in user mode, the writes are ignored.

The PCR module PMPROTSETx registers contain one bit per peripheral memory frame. These bits define the access permissions to the peripheral memory frames. If the CPU attempts to write to a peripheral memory for which it does not have the correct permissions, a protection violation is detected and an Abort occurs.

NOTE: No Access Protection for Reads

The PCR PPROTSETx and PMPROTSETx registers protect the peripheral registers and memories against illegal writes by the CPU. The CPU can read from the peripheral registers and memories in both user and privileged modes.

2.3.3 System Software Interrupts

The system module provides the capability of generating up to four software interrupts. A software interrupt is generated by writing the correct key value to either of the four System Software Interrupt Registers (SSIRx). The SSI registers also allow the application to provide a label for that software interrupt. This label is an 8-bit value that can then be used by the interrupt service routine to perform the required task based on the value provided. The source of the system software interrupt is reflected in the system software interrupt vector (SSIVEC) register.

2.4 Clocks

This section describes the clocking structure of the RM46x microcontrollers.

2.4.1 Clock Sources

The devices support up to 7 clock sources. These are shown in Table 2-9. The electrical specifications as well as timing requirements for each of the clock sources are specified in the device datasheet.

Table 2-9. Clock Sources

Clock Source #	Clock Source Name	Description
0	OSCIN	Main oscillator. This is the primary clock for the microcontroller and is the only clock that is input to the phase-locked loops. The oscillator frequency must be between 5 and 20 MHz.
1	PLL1	This is the output of the main PLL. The PLL is capable of modulating its output frequency in a controlled manner to reduce the radiated emissions.
2	Reserved	This clock source is not available and must not be enabled or used as source for any clock domain.
3	EXTCLKIN1	External clock input 1. A square wave input can be applied to this device input and used as a clock source inside the device.
4	LF LPO (Low-Frequency LPO)	This is the low-frequency output of the internal reference oscillator. This is typically an 80 KHz signal which is used by the real-time interrupt module for generating periodic interrupts to wake up from a low power mode.
5	HF LPO (High-Frequency LPO)	This is the high-frequency output of the internal reference oscillator. This is typically a 10 MHz signal which is used by the clock monitor module as a reference clock to monitor the main oscillator frequency.
6	PLL2	This is the output of the second PLL. There is no option of modulating this PLL’s output signal. This separate non-modulating PLL allows the generation of an asynchronous clock source that is independent of the CPU clock frequency.
7	EXTCLKIN2	External clock input 2. A square wave input can be applied to this device input and used as a clock source inside the device.

2.4.1.1 Enabling / Disabling Clock Sources

Each clock source can be independently enabled or disabled using the set of Clock Source Disable registers – CSDIS, CSDISSET and CSDISCLR.

Each bit in these registers corresponds to the clock source number indicated in Table 2-9. For example, setting bit 1 in the CSDIS or CSDISSET registers disables the PLL#1.

NOTE: Disabling the Main Oscillator or HF LPO

By default, the clock monitoring circuit is enabled and checks for the main oscillator frequency to be within a certain range using the HF LPO as a reference. If the main oscillator and/or the HF LPO are disabled with the clock monitoring still enabled, the clock monitor will indicate an oscillator fault. The clock monitoring must be disabled before disabling the main oscillator or the HF LPO clock source(s).

The clock source is only disabled once there is no active clock domain that is using that clock source. Also, see Chapter 10 for more information on enabling/disabling the oscillator and PLL.

On the RM46x microcontrollers, the clock sources 0, 4 and 5 are enabled by default.

2.4.1.2 Checking for Valid Clock Sources

The application can check whether a clock source is valid or not by checking the corresponding bit to be set in the Clock Source Valid Status (CSVSTAT) register. For example, the application can check if bit 1 in CSVSTAT is set before using the output of PLL#1 as the source for any clock domain.

2.4.2 Clock Domains

The clocking on this device is divided into multiple clock domains for flexibility in control as well as clock source selection. There are 10 clock domains on this device. Each of these are described in Table 2-10.

Each of the control registers listed in Table 2-10 are defined in Section 2.5. The AC timing characteristics for each clock domain are specified in the device datasheet.

Table 2-10. Clock Domains

Clock Domain #	Clock Domain	Default Source	Source Selection Register	Special Considerations
0	GCLK	OSCIN	GHVSRC	• Clock domain used by one of the two Cortex-R4F CPUs operating in lock-step • Always the same frequency as HCLK • In phase with HCLK • Is disabled separately from HCLK via the CDDISx registers bit 0 • Can be divided by 1 up to 8 when running CPU selftest (LBIST) using the CLKDIV field of the STCCLKDIV register at address 0xFFFFE108
0	GCLK2	OSCIN	GHVSRC	• Clock domain used by the second Cortex-R4F CPU operating in lock-step • Always the same frequency as GCLK • 2 cycles delayed from GCLK • Is disabled along with GCLK • Gets divided by the same divider setting as that for GCLK when running CPU selftest (LBIST)
1	HCLK	OSCIN	GHVSRC	• Clock domain used by the high-speed system modules: Flash memory interfaces, TCRAM interface, Error Signaling Module (ESM), DMA • Is disabled via the CDDISx registers bit 1
2	VCLK	OSCIN	GHVSRC	• Clock domain used by some system modules (VIM), peripheral modules accessed via the Peripheral Central Resource (PCR) controller, and all other register interfaces also accessed via the PCR • Divided down from HCLK • Can be HCLK/1, HCLK/2,... or HCLK/16 • Is disabled separately from HCLK via the CDDISx registers bit 2
3	VCLK2	OSCIN	GHVSRC	• Clock domain used by the timer modules: N2HET1, N2HET2 and the dedicated transfer units: HTU1, HTU2 • Divided down from HCLK • Can be HCLK/1, HCLK/2,... or HCLK/16 • Frequency must be an integer multiple of VCLK frequency • Is disabled separately from HCLK via the CDDISx registers bit 3
8	VCLK3	OSCIN	GHVSRC	• Clock domain used for EMAC, USB and EMIF slave interfaces • Divided down from HCLK • Can be HCLK/1, HCLK/2,... or HCLK/16 • Is disabled separately from HCLK via the CDDISx registers bit 8
	VCLK4	OSCIN	GHVSRC	• Clock domain used for ePWM, eCAP, and eQEP modules • Divided down from HCLK • Can be HCLK/1, HCLK/2,... or HCLK/16 • Is disabled separately from HCLK via the CDDISx registers bit 9
4	VCLKA1	VCLK	VCLKASRC	• Clock domain dedicated for the CAN controllers: DCAN1, DCAN2, DCAN3, used for baud-rate generation • Defaults to VCLK as the source • Frequency can be as fast as HCLK frequency • Is disabled via the CDDISx registers bit 4

Table 2-10. Clock Domains (continued)

Clock Domain #	Clock Domain	Default Source	Source Selection Register	Special Considerations
10	VCLKA3	VCLK	VCLKACON1	• Clock domain dedicated to provide the 48MHz clock to the USB host controller and device controller • Defaults to VCLK as the source • Frequency can be as fast as HCLK frequency • Is disabled via the CDDISx registers bit 10
10	VCLKA3_DIVR	VCLK	VCLKACON1	• Clock domain dedicated to provide the 12MHz clock to the USB host controller • Divided down from VCLKA3 using the VCLKA3R field of the VCLKACON1 register • Frequency can be VCLKA3/1, VCLKA3/2, ..., or VCLKA3/8 • Default frequency is VCLKA3/2 • Is disabled separately via the VCLKACON1 register's VCLKA3_DIV_CDDIS bit, if the VCLKA3 is not already disabled
11	VCLKA4	VCLK	VCLKACON1	• Clock domain dedicated for the Ethernet controller (EMAC) when the TX and RX clocks are internally generated • Defaults to VCLK as the source • Frequency can be as fast as HCLK frequency • Is disabled via the CDDISx registers bit 11
11	VCLKA4_ DIVR	VCLK	VCLKACON1	• Divided down from VCLKA4 using the VCLKA4R field of the VCLKACON1 register • Frequency can be VCLKA4/1, VCLKA4/2, ..., or VCLKA4/8 • Default frequency is VCLKA4/2 • Is disabled separately via the VCLKACON1 register's VCLKA4_DIV_CDDIS bit, if the VCLKA4 is not already disabled
6	RTICLK	VCLK	RCLKSRC	• Clock domain dedicated for timebase generation in the Real-Time Interrupt (RTI) generation module • Defaults to VCLK as the source • If a clock source other than VCLK is selected for RTICLK, then the RTICLK frequency must be less than or equal to VCLK/3 • Application can ensure this by programming the RTI1DIV field of the RCLKSRC register, if necessary • Is disabled via the CDDISx registers bit 6

2.4.2.1 Enabling / Disabling Clock Domains

Each clock domain can be independently enabled or disabled using the set of Clock Domain Disable registers – CDDIS, CDDISSET, and CDDISCLR.

Each bit in these registers corresponds to the clock domain number indicated in Table 2-10. For example, setting bit 1 in the CDDIS or CDDISSET registers disables the HCLK clock domain. The clock domain will be turned off only when every module that uses the HCLK domain gives the “permission” for HCLK to be turned off.

All clock domains are enabled by default, or upon a system reset, or whenever a wake up condition is detected.

2.4.2.2 Mapping Clock Sources to Clock Domains

Each clock domain needs to be mapped to a valid clock source. There are control registers that allow an application to choose the clock sources for each clock domain.

- ***Selecting clock source for GCLK, HCLK and VCLKx domains***

The CPU clock (GCLK), the system module clock (HCLK), and the peripheral bus clocks (VCLKx) all use the same clock source. This clock source is selected via the GHVSRC register. The default source for the GCLK, HCLK and VCLKx is the main oscillator. That is, after power up, the GCLK and HCLK are running at the OSCIN frequency, while the VCLKx frequency is the OSCIN frequency divided by 2.

- ***Selecting clock source for VCLKA1 domains***

The clock source for VCLKA1 domains is selected via the VCLKASRC register. The default source for the VCLKA1 domains is the VCLK.

- ***Selecting clock source for VCLKA4 domain***

The clock source for VCLKA4 domain is selected via the VCLKACON1 register. The default source for the VCLKA4 domain is the VCLK.

- ***Selecting clock source for RTICLK domain***

The clock source for RTICLK domain is selected via the RCLKSRC register. The default source for the RTICLK domain is the VCLK.

NOTE: Selecting a clock source for RTICLK that is not VCLK

When the application chooses a clock source for RTICLK domain that is not VCLK, then the application must ensure that the resulting RTICLK frequency must be less than or equal to VCLK frequency divided by 3. The application can configure the RTI1DIV field of the RCLKSRC register for dividing the selected clock source frequency by 1, 2, 4 or 8 to meet this requirement.

2.4.3 Low Power Modes

All clock domains are active in the normal operating mode. This is the default mode of operation. As described in Section 2.4.1.1 and Section 2.4.2.1, the application can choose to disable any particular clock source and domain that it does not plan to use. Also, the peripheral central resource controller (PCR) has control registers to enable / disable the peripheral clock (VCLK) for each peripheral select. This offers the application a large number of choices for enabling / disabling clock sources, or clock domains, or clocks to specific peripherals.

This section describes three particular low-power modes and their typical characteristics. They are not the only low-power modes configurable by the application, as just described.

Table 2-11. Typical Low-Power Modes

Mode Name	Active Clock Source(s)	Active Clock Domain(s)	Wake Up Options	Suggested Wake Up Clock Source(s)	Wake Up Time(wake up detected -to- CPU code execution start)
Doze	Main oscillator	RTICLK	RTI interrupt, GIO interrupt, CAN message, SCI message	Main oscillator	Flash pump sleep -> standby transition time + Flash bank sleep -> standby transition time + Flash bank standby -> active transition time
Snooze	LF LPO	RTICLK	RTI interrupt, GIO interrupt, CAN message, SCI message	HF LPO	HF LPO warm start-up time + Flash pump sleep -> standby transition time + Flash bank sleep -> standby transition time + Flash bank standby -> active transition time
Sleep	None	None	GIO interrupt, CAN message, SCI message	HF LPO	HF LPO warm start-up time + Flash pump sleep -> standby transition time + Flash bank sleep -> standby transition time + Flash bank standby -> active transition time

2.4.3.1 Typical Software Sequence to Enter a Low-Power Mode

1. Program the flash banks and flash pump fall-back modes to be "sleep".

   The flash pump transitions from active to sleep mode only after all the flash banks have switched from active to sleep mode. The flash banks start switching from active to sleep mode only after the banks are not accessed for at least a duration defined by the Active Grace Period (AGP) parameter configured for the banks. See Chapter 5 for more details.

2. Disable the clock sources that are not required to be kept active.

   A clock source does not get disabled until all clock domains using that clock source are disabled first, or are configured to use an alternate clock source.

3. Disable the clock domains that are not required to be kept active.

   A clock domain does not get disabled until all modules using that clock domain "give their permission" for that clock domain to be turned off.

4. Idle the Cortex-R4F core.

   The ARM Cortex-R4F CPU has internal power management logic, and requires a dedicated instruction to be used in order to enter a low power mode. This is the Wait For Interrupt (WFI) instruction.

   When a WFI instruction is executed, the Cortex-R4F core flushes its pipeline, flushes all write buffers, and completes all pending bus transactions. At this time the core indicates to the system that the clock to the core can be stopped. This indication is used by the Global Clock Module (GCM) to turn off the CPU clock domain (GCLK) if the CDDIS register bit 0 is set.

2.4.3.2 Special Considerations for Entry to Low Power Modes

Some bus master modules – High-End Timer Transfer Units (HTUx), and Parameter Overlay Module (POM), can have ongoing transactions when the application wants to enter a low power mode to turn off the clocks to those modules. This is not recommended as it could leave the device in an unpredictable state. Refer to the individual module user guides for more information about the sequence to be followed to safely enter a low-power mode.

2.4.3.3 Selecting Clock Source Upon Wake Up

The domains for CPU clock (GCLK), the system clock (HCLK) and the peripheral clock (VCLKx) use the same clock source selected via the GHVSRC field of the GHVSRC register. The GHVSRC register also allows the application to choose the clock source after wake up via the GHVWAKE field.

When a wake up condition is detected, if the selected wake up clock source is not already active, the global clock module (GCM) will enable this selected clock source, wait for it to become valid, and then use it for the GCLK, HCLK and VCLKx domains. The other clock domains VCLKAx and RTICLK retain the configuration for their clock source selection registers – VCLKASRC, VCLKACON1 and RCLKSRC.

2.4.4 Clock Test Mode

The RM46x microcontrollers support a test mode which allows a user to bring out several different clock sources and clock domains on to the ECLK terminal. This is very useful information for debug purposes. Each clock source also has a corresponding clock source valid status flag in the Clock Source Valid Status (CSVSTAT) register. The clock source valid status flags can also be brought out on to the N2HET1[12] terminal in this clock test mode.

The clock test mode is controlled by the CLKTEST register (Section 2.5.1.32) in the system module register frame.

The clock test mode is enabled by writing 5h to the CLK_TEST_EN field.

The signal to be brought out on to the ECLK terminal is defined by the SEL_ECP_PIN field and the signal to be brought out on to the N2HET1[12] terminal is defined by the SEL_GIO_PIN field.

The choices for these selections are defined in Table 2-12.

Table 2-12. Clock Test Mode Options

SEL_ECP_PIN	Signal on ECLK	SEL_GIO_PIN	Signal on N2HET1[12]
0000	Oscillator	0000	Oscillator Valid status
0001	PLL1 free-running clock output	0001	PLL1 Valid status
0010	Reserved	0010	Reserved
0011	EXTCLKIN1	0011	Reserved
0100	Low-frequency LPO (Low-Power Oscillator) clock	0100	Reserved
0101	High-frequency LPO (Low-Power Oscillator) clock	0101	HF LPO Valid status
0110	PLL2 free-running clock output	0110	PLL2 Valid Status
0111	EXTCLKIN2	0111	Reserved
1000	GCLK	1000	LF LPO Valid status
1001	RTI Base	1001	Oscillator Valid status
1010	Reserved	1010	Oscillator Valid status
1011	VCLKA1	1011	Oscillator Valid status
1100	Reserved	1100	Oscillator Valid status
1101	VCLKA3_DIVR	1101	VCLKA3
1110	VCLKA4_DIVR	1110	VCLKA4
1111	Reserved	1111	Oscillator Valid status

2.4.5 Safety Considerations for Clocks

The RM46x microcontrollers are targeted for use in several safety-critical applications. The following sections describe the internal or external monitoring mechanisms that detect and signal clock source failures.

2.4.5.1 Oscillator Monitor

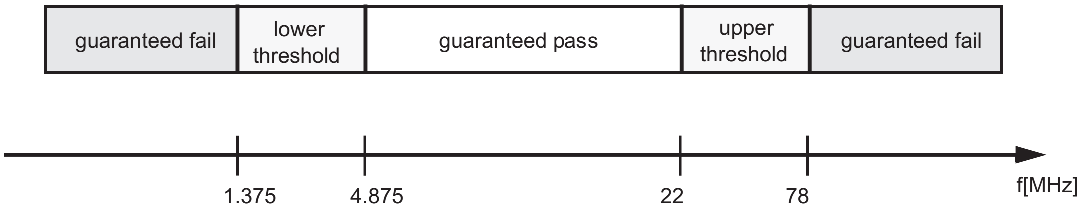
The oscillator clock frequency is monitored by a dedicated circuitry called CLKDET using the HF LPO as the reference clock. The CLKDET flags an oscillator fail condition whenever the OSCIN frequency falls outside of a range which is defined by the HF LPO frequency.

The valid OSCIN range is defined as a minimum of $f_{(HF\ LPO)}$ / 4 to a maximum of $f_{(HF\ LPO)}$ × 4.

The untrimmed HF LPO frequency on this device can range from 5.5MHz to 19.5MHz. This results in a valid OSCIN frequency range depicted in Figure 2-4.

The application can select the device response to an oscillator fail indication. See Chapter 10 for more details on the oscillator monitoring and the system response choices.

Figure 2-4. LPO and Clock Detection, Untrimmed HF LPO



2.4.5.2 PLL Slip Detector

Both the PLL macros implemented on the microcontrollers have an embedded slip detection circuit. A PLL slip is detected by the slip detector under the following conditions:

1. Reference cycle slip, RFSLIP — the output clock is running *too fast* relative to the reference clock
2. Feedback cycle slip, FBSLIP — the output clock is running *too slow* relative to the reference clock

The device also includes optional filters that can be enabled before a slip indication from the PLL is actually logged in the system module Global Status Register (GLBSTAT). Also, once a PLL slip condition is logged in the system module global status register, the application can choose the device’s response to the slip indication. Refer to the Oscillator and PLL chapter for more details on PLL slip and the system response choices.

2.4.5.3 External Clock Monitor

The microcontrollers support a terminal called ECLK – External Clock, which is used to output a slow frequency which is divided down from the device system clock frequency. An external circuit can monitor the ECLK frequency in order to check that the device is operating at the correct frequency.

The frequency of the signal output on the ECLK pin can be divided down by 1 to 65536 from the peripheral clock (VCLK) frequency using the External Clock Prescaler Control Register (ECPCNTL). The actual clock output on ECLK is enabled by setting the ECP CLK FUN bit of the SYSPC1 control register.

2.4.5.4 Dual-Clock Comparators

The microcontrollers include two instances of the dual-clock comparator module. This module includes two down counters which independently count from two separate seed values at the rate of two independent clock frequencies. One of the clock inputs is a reference clock input, selectable between the main oscillator or the HF LPO in functional mode. The second clock input is selectable from among a set of defined signals as described in Section 2.4.5.4.1 and Section 2.4.5.4.2. This mechanism can be used to use a known-good clock to measure the frequency of another clock.

2.4.5.4.1 DCC1

Table 2-13. DCC1 Counter 0 Clock Inputs

Clock Source [3-0]	Clock / Signal Name
others	oscillator (OSCIN)
0x5	HF LPO
0xA	test clock (TCK)

Table 2-14. DCC1 Counter 1 Clock / Signal Inputs

Key [3-0]	Clock Source [3-0]	Clock / Signal Name
0xA	0x0	PLL1 free-running clock output
	0x1	PLL2 free-running clock output
	0x2	LF LPO
	0x3	HF LPO
	0x4	Reserved
	0x5	EXTCLKIN1
	0x6	EXTCLKIN2
	0x7	Reserved
	0x8-0xF	VCLK
others	any value	N2HET1[31]

As can be seen, the main oscillator (OSCIN) can be used for counter 0 as a “known-good” reference clock. The clock for counter 1 can be selected from among 8 options. See Chapter 11 for more details on the DCC usage.

2.4.5.4.2 DCC2

Table 2-15. DCC2 Counter 0 Clock Inputs

Clock Source [3-0]	Clock / Signal Name
others	oscillator (OSCIN)
0xA	test clock (TCK)

Table 2-16. DCC2 Counter 1 Clock / Signal Inputs

Key [3-0]	Clock Source [3-0]	Clock / Signal Name
0xA	0x0-0x7	Reserved
	0x8-0xF	VCLK
others	any value	N2HET2[0]

As can be seen, the main oscillator (OSCIN) can be used for counter 0 as a “known-good” reference clock. The clock for counter 1 can be selected from among 2 options. See Chapter 11 for more details on the DCC usage.

2.5 System and Peripheral Control Registers

The following sections describe the system and peripheral control registers of the RM46x microcontroller.

2.5.1 Primary System Control Registers (SYS)

This section describes the SYSTEM registers. These registers are divided into two separate frames. The start address of the primary system module frame is FFFF FF00h. The start address of the secondary system module frame is FFFF E100h. The registers support 32-, 16-, and 8-bit writes. The offset is relative to the system module frame start address.

Table 2-17 contains a list of the primary system control registers.

Table 2-17. Primary System Control Registers

Offset	Acronym	Register Description	Section
00h	SYSPC1	SYS Pin Control Register 1	Section 2.5.1.1
04h	SYSPC2	SYS Pin Control Register 2	Section 2.5.1.2
08h	SYSPC3	SYS Pin Control Register 3	Section 2.5.1.3
0Ch	SYSPC4	SYS Pin Control Register 4	Section 2.5.1.4
10h	SYSPC5	SYS Pin Control Register 5	Section 2.5.1.5
14h	SYSPC6	SYS Pin Control Register 6	Section 2.5.1.6
18h	SYSPC7	SYS Pin Control Register 7	Section 2.5.1.7
1Ch	SYSPC8	SYS Pin Control Register 8	Section 2.5.1.8
20h	SYSPC9	SYS Pin Control Register 9	Section 2.5.1.9
30h	CSDIS	Clock Source Disable Register	Section 2.5.1.10
34h	CSDISSET	Clock Source Disable Set Register	Section 2.5.1.11
38h	CSDISCLR	Clock Source Disable Clear Register	Section 2.5.1.12
3Ch	CDDIS	Clock Domain Disable Register	Section 2.5.1.13
40h	CDDISSET	Clock Domain Disable Set Register	Section 2.5.1.14
44h	CDDISCLR	Clock Domain Disable Clear Register	Section 2.5.1.15
48h	GHVSRC	GCLK, HCLK, VCLK, and VCLK2 Source Register	Section 2.5.1.16
4Ch	VCLKASRC	Peripheral Asynchronous Clock Source Register	Section 2.5.1.17
50h	RCLKSRC	RTI Clock Source Register	Section 2.5.1.18
54h	CSVSTAT	Clock Source Valid Status Register	Section 2.5.1.19
58h	MSTGCR	Memory Self-Test Global Control Register	Section 2.5.1.20
5Ch	MINITGCR	Memory Hardware Initialization Global Control Register	Section 2.5.1.21
60h	MSINENA	Memory Self-Test/Initialization Enable Register	Section 2.5.1.22
64h	MSTFAIL	Memory Self-Test Fail Status Register	Section 2.5.1.23
68h	MSTCGSTAT	MSTC Global Status Register	Section 2.5.1.24
6Ch	MINISTAT	Memory Hardware Initialization Status Register	Section 2.5.1.25
70h	PLLCTL1	PLL Control Register 1	Section 2.5.1.26
74h	PLLCTL2	PLL Control Register 2	Section 2.5.1.27
78h	SYSPC10	SYS Pin Control Register 10	Section 2.5.1.28
7Ch	DIEIDL	Die Identification Register, Lower Word	Section 2.5.1.29
80h	DIEIDH	Die Identification Register, Upper Word	Section 2.5.1.30
88h	LPOMONCTL	LPO/Clock Monitor Control Register	Section 2.5.1.31
8Ch	CLKTEST	Clock Test Register	Section 2.5.1.32
90h	DFTCTRLREG	DFT Control Register	Section 2.5.1.33
94h	DFTCTRLREG2	DFT Control Register 2	Section 2.5.1.34
98h-9Ch	Reserved	Reserved	
A0h	GPREG1	General Purpose Register	Section 2.5.1.35
A8h	IMPFASTS	Imprecise Fault Status Register	Section 2.5.1.36

Table 2-17. Primary System Control Registers (continued)

Offset	Acronym	Register Description	Section
ACh	IMPFTADD	Imprecise Fault Write Address Register	Section 2.5.1.37
B0h	SSIR1	System Software Interrupt Request 1 Register	Section 2.5.1.38
B4h	SSIR2	System Software Interrupt Request 2 Register	Section 2.5.1.39
B8h	SSIR3	System Software Interrupt Request 3 Register	Section 2.5.1.40
BCh	SSIR4	System Software Interrupt Request 4 Register	Section 2.5.1.41
C0h	RAMGCR	RAM Control Register	Section 2.5.1.42
C4h	BMMCR1	Bus Matrix Module Control Register 1	Section 2.5.1.43
C8h	Reserved	Reserved	
CCh	CPURSTCR	CPU Reset Control Register	Section 2.5.1.44
D0h	CLKCNTL	Clock Control Register	Section 2.5.1.45
D4h	ECPCNTL	ECP Control Register	Section 2.5.1.46
DCh	DEVCR1	DEV Parity Control Register 1	Section 2.5.1.47
E0h	SYSECR	System Exception Control Register	Section 2.5.1.48
E4h	SYSESR	System Exception Status Register	Section 2.5.1.49
E8h	SYSTASR	System Test Abort Status Register	Section 2.5.1.50
ECh	GLBSTAT	Global Status Register	Section 2.5.1.51
F0h	DEVID	Device Identification Register	Section 2.5.1.52
F4h	SSIVEC	Software Interrupt Vector Register	Section 2.5.1.53
F8h	SSIF	System Software Interrupt Flag Register	Section 2.5.1.54

2.5.1.1 SYS Pin Control Register 1 (SYSPC1)

The SYSPC1 register, shown in Figure 2-5 and described in Table 2-18, controls the function of the ECLK pin.

Figure 2-5. SYS Pin Control Register 1 (SYSPC1) [offset = 00]

31	16
Reserved	
R-0	

15	1	0
Reserved		ECPCLKFUN
R-0		R/W-0

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

Table 2-18. SYS Pin Control Register 1 (SYSPC1) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ECPCLKFUN		ECLK function. This bit changes the function of the ECLK pin.
		0	ECLK is in GIO mode.
		1	ECLK is in functional mode as a clock output. **Note: Proper ECLK duty cycle is not assured until 1 ECLK cycle has elapsed after switching into functional mode.**

2.5.1.2 SYS Pin Control Register 2 (SYSPC2)

The SYSPC2 register, shown in Figure 2-6 and described in Table 2-19, controls whether the pin is an input or an output when in GIO mode.

Figure 2-6. SYS Pin Control Register 2 (SYSPC2) [offset = 04h]

31	16
Reserved	
R-0	

15	1	0
Reserved		ECPCLKDIR
R-0		R/W-0

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

Table 2-19. SYS Pin Control Register 2 (SYSPC2) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ECPCLKDIR		ECLK data direction. This bit controls the direction of the ECLK pin when it is configured to be in GIO mode only.
		0	The ECLK pin is an input. **Note: If the pin direction is set as an input, the output buffer is tristated.**
		1	The ECLK pin is an output. **Note: The ECLK pin is placed into GIO mode by clearing the ECPCLKFUN bit to 0 in the SYSPC1 register.**

2.5.1.3 SYS Pin Control Register 3 (SYSPC3)

The SYSPC3 register, shown in Figure 2-7 and described in Table 2-20, displays the logic state of the ECLK pin when it is in GIO mode.

Figure 2-7. SYS Pin Control Register 3 (SYSPC3) [offset = 08h]

31	16
Reserved	
R-0	

15	1	0
Reserved		ECPCLKDIN
R-0		R-U

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

Table 2-20. SYS Pin Control Register 3 (SYSPC3) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ECPCLKDIN		ECLK data in. This bit displays the logic state of the ECLK pin when it is configured to be in GIO mode.
		0	The ECLK pin is at logic low (0).
		1	The ECLK pin is at logic high (1).

2.5.1.4 SYS Pin Control Register 4 (SYSPC4)

The SYSPC4 register, shown in Figure 2-8 and described in Table 2-21, controls the logic level output function of the ECLK pin when it is configured as an output in GIO mode.

Figure 2-8. SYS Pin Control Register 4 (SYSPC4) [offset = 0Ch]

31	16
Reserved	
R-0	

15	1	0
Reserved		ECPCLKDOUT
R-0		R/W-0

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

Table 2-21. SYS Pin Control Register 4 (SYSPC4) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ECPCLKDOUT		ECLK data out write. This bit is only active when the ECLK pin is configured to be in GIO mode. Writes to this bit will only take effect when the ECLK pin is configured as an output in GIO mode. The current logic state of the ECLK pin will be displayed by this bit in both input and output GIO mode.
		0	The ECLK pin is driven to logic low (0).
		1	The ECLK pin is driven to logic high (1).
			Note: The ECLK pin is placed into GIO mode by setting the ECPCLKFUN bit to 0 in the SYSPC1 register. The ECLK pin is placed in output mode by setting the ECPCLKDIR bit to 1 in the SYSPC2 register.

2.5.1.5 SYS Pin Control Register 5 (SYSPC5)

The SYSPC5 register, shown in Figure 2-9 and described in Table 2-22, controls the set function of the ECLK pin when it is configured as an output in GIO mode.

Figure 2-9. SYS Pin Control Register 5 (SYSPC5) [offset = 10h]

31–16
Reserved
R-0

15–1	0
Reserved	ECPCLKSET
R-0	R/W-0

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

Table 2-22. SYS Pin Control Register 5 (SYSPC5) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ECPCLKSET		ECLK data out set. This bit drives the output of the ECLK pin high when set in GIO output mode.
		0	*Write:* Writing a 0 has no effect.
		1	*Write:* The ECLK pin is driven to logic high (1).
			Note: The current logic state of the ECPCLKDOUT bit will also be displayed by this bit when the pin is configured in GIO output mode.
			Note: The ECLK pin is placed into GIO mode by setting the ECPCLKFUN bit to 0 in the SYSPC1 register. The ECLK pin is placed in output mode by setting the ECPCLKDIR bit to 1 in the SYSPC2 register.

2.5.1.6 SYS Pin Control Register 6 (SYSPC6)

The SYSPC6 register, shown in Figure 2-10 and described in Table 2-23, controls the clear function of the ECLK pin when it is configured as an output in GIO mode..

Figure 2-10. SYS Pin Control Register 6 (SYSPC6) [offset = 14h]

31–16
Reserved
R-0

15–1	0
Reserved	ECPCLKCLR
R-0	R/W-0

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

Table 2-23. SYS Pin Control Register 6 (SYSPC6) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ECPCLKCLR		ECLK data out clear. This bit drives the output of the ECLK pin low when set in GIO output mode.
		0	*Write:* The ECLK pin value is unchanged.
		1	*Write:* The ECLK pin is driven to logic low (0).
			Note: The current logic state of the ECPCLKDOUT bit will also be displayed by this bit when the pin is configured in GIO output mode.
			Note: The ECLK pin is placed into GIO mode by setting the ECPCLKFUN bit to 0 in the SYSPC1 register. The ECLK pin is placed in output mode by setting the ECPCLKDIR bit to 1 in the SYSPC2 register.

2.5.1.7 SYS Pin Control Register 7 (SYSPC7)

The SYSPC7 register, shown in Figure 2-11 and described in Table 2-24, controls the open drain function of the ECLK pin.

Figure 2-11. SYS Pin Control Register 7 (SYSPC7) [offset = 18h]

31 … 16
Reserved
R-0

15 … 1	0
Reserved	ECPCLKODE
R-0	R/W-0

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

Table 2-24. SYS Pin Control Register 7 (SYSPC7) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ECPCLKODE		ECLK open drain enable. This bit is only active when ECLK is configured to be in GIO mode.
		0	The ECLK pin is configured in push/pull (normal GIO) mode.
		1	The ECLK pin is configured in open drain mode. The ECPCLKDOUT bit in the SYSPC4 register controls the state of the ECLK output buffer: ECPCLKDOUT = 0: The ECLK output buffer is driven low. ECPCLKDOUT = 1: The ECLK output buffer is tristated. **Note: The ECLK pin is placed into GIO mode by setting the ECPCLKFUN bit to 0 in the SYSPC1 register.**

2.5.1.8 SYS Pin Control Register 8 (SYSPC8)

The SYSPC8 register, shown in Figure 2-12 and described in Table 2-25, controls the pull enable function of the ECLK pin when it is configured as an input in GIO mode.

Figure 2-12. SYS Pin Control Register 8 (SYSPC8) [offset = 1Ch]

31 … 16
Reserved
R-0

15 … 1	0
Reserved	ECPCLKPUE
R-0	R/W-D

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

Table 2-25. SYS Pin Control Register 8 (SYSPC8) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ECPCLKPUE		ECLK pull enable. Writes to this bit will only take effect when the ECLK pin is configured as an input in GIO mode.
		0	ECLK pull enable is active.
		1	ECLK pull enable is inactive. **Note: The pull direction (up/down) is selected by the ECPCLKPS bit in the SYSPC9 register.** **Note: The ECLK pin is placed into GIO mode by setting the ECPCLKFUN bit to 0 in the SYSPC1 register. The ECLK pin is placed in input mode by setting the ECPCLKDIR bit to 0 in the SYSPC2 register.**

2.5.1.9 SYS Pin Control Register 9 (SYSPC9)

The SYSPC9 register, shown in Figure 2-13 and described in Table 2-26, controls the pull up/pull down configuration of the ECLK pin when it is configured as an input in GIO mode.

Figure 2-13. SYS Pin Control Register 9 (SYSPC9) [offset = 20h]

31–16
Reserved
R-0

15–1	0
Reserved	ECPCLKPS
R-0	R/W-0

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

Table 2-26. SYS Pin Control Register 9 (SYSPC9) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ECPCLKPS		ECLK pull up/pull down select. This bit is only active when ECLK is configured as an input in GIO mode and the pull up/pull down logic is enabled.
		0	ECLK pull down is selected, when pull up/pull down logic is enabled.
		1	ECLK pull up is selected, when pull up/pull down logic is enabled.
			Note: The ECLK pin pull up/pull down logic is enabled by setting the ECPCLKPUE bit to 0 in the SYSPC8 register.
			Note: The ECLK pin is placed into GIO mode by setting the ECPCLKFUN bit to 0 in the SYSPC1 register. The ECLK pin is placed in input mode by setting the ECPCLKDIR bit to 0 in the SYSPC2 register.

2.5.1.10 Clock Source Disable Register (CSDIS)

The CSDIS register, shown in Figure 2-14 and described in Table 2-27, controls and displays the state of the device clock sources.

Figure 2-14. Clock Source Disable Register (CSDIS) [offset = 30h]

31–8
Reserved
R-0

7	6	5	4	3	2	1	0
CLKSR7OFF	CLKSR6OFF	CLKSR5OFF	CLKSR4OFF	CLKSR3OFF	Reserved	CLKSR1OFF	CLKSR0OFF
R/WP-1	R/WP-1	R/WP-0	R/WP-0	R/WP-1	R-1	R/WP-1	R/WP-0

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

Table 2-27. Clock Source Disable Register (CSDIS) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-3	CLKSR[7-3]OFF		Clock source[7-3] off.
		0	Clock source[7-3] is enabled.
		1	Clock source[7-3] is disabled.
			Note: On wakeup, only clock sources 0, 4, and 5 are enabled.
2	Reserved		Reads return 1. Writes have no effect.
1-0	CLKSR[1-0]OFF		Clock source[1-0] off.
		0	Clock source[1-0] is enabled.
		1	Clock source[1-0] is disabled.
			Note: On wakeup, only clock sources 0, 4, and 5 are enabled.

Table 2-28. Clock Sources Table

Clock Source #	Clock Source Name
Clock Source 0	Oscillator
Clock Source 1	PLL1
Clock Source 2	Not Implemented
Clock Source 3	EXTCLKIN
Clock Source 4	Low Frequency LPO (Low Power Oscillator) clock
Clock Source 5	High Frequency LPO (Low Power Oscillator) clock
Clock Source 6	PLL2
Clock Source 7	EXTCLKIN2

NOTE: Nonimplemented clock sources should not be enabled or used.

2.5.1.11 Clock Source Disable Set Register (CSDISSET)

The CSDISSET register, shown in Figure 2-15 and described in Table 2-29, sets clock sources to the disabled state.

Figure 2-15. Clock Source Disable Set Register (CSDISSET) [offset = 34h]

31-8
Reserved
R-0

7	6	5	4	3	2	1	0
SETCLKSR7 OFF	SETCLKSR6 OFF	SETCLKSR5 OFF	SETCLKSR4 OFF	SETCLKSR3 OFF	Reserved	SETCLKSR1 OFF	SETCLKSR0 OFF
R/WP-1	R/WP-1	R/WP-0	R/WP-0	R/WP-1	R-1	R/WP-1	R/WP-0

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

Table 2-29. Clock Source Disable Set Register (CSDISSET) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-3	SETCLKSR[7-3]OFF		Set clock source[7-3] to the disabled state.
		0	*Read:* Clock source[7-3] is enabled. *Write:* Clock source[7-3] is unchanged.
		1	*Read:* Clock source[7-3] is disabled. *Write:* Clock source[7-3] is set to the disabled state. **Note: After a new clock source disable bit is set via the CSDISSET register, the new status of the bit will be reflected in the CSDIS register (offset 30h), the CSDISSET register (offset 34h) and the CSDISCLR register (offset 38h).**
2	Reserved		Reads return 1. Writes have no effect.
1-0	SETCLKSR[1-0]OFF		Set clock source[1-0] to the disabled state.
		0	*Read:* Clock source[1-0] is enabled. *Write:* Clock source[1-0] is unchanged.
		1	*Read:* Clock source[1-0] is disabled. *Write:* Clock source[1-0] is set to the disabled state. **Note: After a new clock source disable bit is set via the CSDISSET register, the new status of the bit will be reflected in the CSDIS register (offset 30h), the CSDISSET register (offset 34h) and the CSDISCLR register (offset 38h).**

NOTE: A list of the available clock sources is shown in Table 2-28.

2.5.1.12 Clock Source Disable Clear Register (CSDISCLR)

The CSDISCLR register, shown in Figure 2-16 and described in Table 2-30, clears clock sources to the enabled state.

Figure 2-16. Clock Source Disable Clear Register (CSDISCLR) [offset = 38h]

31-8
Reserved
R-0

7	6	5	4	3	2	1	0
CLRCLKSR7 OFF	CLRCLKSR6 OFF	CLRCLKSR5 OFF	CLRCLKSR4 OFF	CLRCLKSR3 OFF	Reserved	CLRCLKSR1 OFF	CLRCLKSR0 OFF
R/WP-1	R/WP-1	R/WP-0	R/WP-0	R/WP-1	R-1	R/WP-1	R/WP-0

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

Table 2-30. Clock Source Disable Clear Register (CSDISCLR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-3	CLRCLKSR[7-3]OFF		Enables clock source[7-3].
		0	*Read:* Clock source[7-3] is enabled. *Write:* Clock source[7-3] is unchanged.
		1	*Read:* Clock source[7-3] is enabled. *Write:* Clock source[7-3] is set to the enabled state. **Note: After a new clock source disable bit is set via the CSDISSET register, the new status of the bit will be reflected in the CSDIS register (offset 30h), the CSDISSET register (offset 34h) and the CSDISCLR register (offset 38h).**
2	Reserved		Reads return 1. Writes have no effect.
1-0	CLRCLKSR[1-0]OFF		Enables clock source[1-0].
		0	*Read:* Clock source[1-0] is enabled. *Write:* Clock source[1-0] is unchanged.
		1	*Read:* Clock source[1-0] is enabled. *Write:* Clock source[1-0] is set to the enabled state. **Note: After a new clock source disable bit is set via the CSDISSET register, the new status of the bit will be reflected in the CSDIS register (offset 30h), the CSDISSET register (offset 34h) and the CSDISCLR register (offset 38h).**

NOTE: A list of the available clock sources is shown in Table 2-28.

2.5.1.13 Clock Domain Disable Register (CDDIS)

The CDDIS register, shown in Figure 2-17 and described in Table 2-31, controls the state of the clock domains.

NOTE: All the clock domains are enabled on wakeup.

The application should guarantee that when HCLK and VCLK_sys are turned off through the HCLKOFF bit, the GCLK domain is also turned off.

The register bits in CDDIS are designated as high-integrity bits and have been implemented with error-correcting logic such that each bit, although read and written as a single bit, is actually a multi-bit key with error correction capability. As such, single-bit flips within the “key” can be corrected allowing protection of the system as a whole. An error detected is signaled to the ESM module.

Figure 2-17. Clock Domain Disable Register (CDDIS) [offset = 3Ch]

31-16
Reserved
R-0

15-12	11	10	9	8
Reserved	VCLKA4OFF	VCLKA3OFF	VCLK4OFF	VCLK3OFF
R-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

7	6	5	4	3	2	1	0
Reserved	RTICLK1OFF	Reserved	VCLKA1OFF	VCLK2OFF	VCLKPOFF	HCLKOFF	GCLKOFF
R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

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

Table 2-31. Clock Domain Disable Register (CDDIS) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reads return 0. Writes have no effect.
11-10	VCLKA[4-3]OFF		VCLKA[4-3] domain off.
		0	The VCLKA[4-3] domain is enabled.
		1	The VCLKA[4-3] domain is disabled.
9-8	VCLK[4-3]OFF		VCLK[4-3] domain off.
		0	The VCLK[4-3] domain is enabled.
		1	The VCLK[4-3] domain is disabled.
7	Reserved	0-1	Reads return 0 or 1 and privilege mode writes allowed.
6	RTICLK1OFF		RTICLK1 domain off.
		0	The RTICLK1 domain is enabled.
		1	The RTICLK1 domain is disabled.
5	Reserved	0-1	Reads return 0 or 1 and privilege mode writes allowed.
4	VCLKA1OFF		VCLKA1 domain off.
		0	The VCLKA1 domain is enabled.
		1	The VCLKA1 domain is disabled.
3	VCLK2OFF		VCLK2 domain off.
		0	The VCLK2 domain is enabled.
		1	The VCLK2 domain is disabled.
2	VCLKPOFF		VCLK_periph domain off.
		0	The VCLK_periph domain is enabled.
		1	The VCLK_periph domain is disabled.

Table 2-31. Clock Domain Disable Register (CDDIS) Field Descriptions (continued)

Bit	Field	Value	Description
1	HCLKOFF		HCLK and VCLK_sys domains off.
		0	The HCLK and VCLK_sys domains are enabled.
		1	The HCLK and VCLK_sys domains are disabled.
0	GCLKOFF		GCLK domain off.
		0	The GCLK domain is enabled.
		1	The GCLK domain is disabled.

2.5.1.14 Clock Domain Disable Set Register (CDDISSET)

This CDDISSET register, shown in Figure 2-18 and described in Table 2-32, sets clock domains to the disabled state.

Figure 2-18. Clock Domain Disable Set Register (CDDISSET) [offset = 40h]

31	16
Reserved	
R-0	

15	12	11	10	9	8
Reserved		SETVCLKA4 OFF	SETVCLKA3 OFF	SETVCLK4 OFF	SETVCLK3 OFF
R-0		R/WP-0	R/WP-0	R/WP-0	R/WP-0

7	6	5	4	3	2	1	0
Reserved	SETRTI1CLK OFF	Reserved	SETVCLKA1 OFF	SETVCLK2 OFF	SETVCLKP OFF	SETHCLK OFF	SETGCLK OFF
R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

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

Table 2-32. Clock Domain Disable Set Register (CDDISSET) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reads return 0. Writes have no effect.
11-10	SETVCLKA[4-3]OFF		Set VCLKA[4-3] domain.
		0	*Read:* The VCLKA[4-3] domain is enabled. *Write:* The VCLKA[4-3] domain is unchanged.
		1	*Read:* The VCLKA[4-3] domain is disabled. *Write:* The VCLKA[4-3] domain is set to the enabled state.
9-8	SETVCLK[4-3]OFF		Set VCLK[4-3] domain.
		0	*Read:* The VCLK[4-3] domain is enabled. *Write:* The VCLK[4-3] domain is unchanged.
		1	*Read:* The VCLK[4-3] domain is disabled. *Write:* The VCLK[4-3] domain is set to the enabled state.
7	Reserved	0-1	Reads return 0 or 1 and privilege mode writes allowed.
6	SETRTI1CLKOFF		Set RTICLK1 domain.
		0	*Read:* The RTICLK1 domain is enabled. *Write:* The RTICLK1 domain is unchanged.
		1	*Read:* The RTICLK1 domain is disabled. *Write:* The RTICLK1 domain is set to the enabled state.
5	Reserved	0-1	Reads return 0 or 1 and privilege mode writes allowed.
4	SETVCLKA1OFF		Set VCLKA1 domain.
		0	*Read:* The VCLKA1 domain is enabled. *Write:* The VCLKA1 domain is unchanged.
		1	*Read:* The VCLKA1 domain is disabled. *Write:* The VCLKA1 domain is set to the enabled state.
3	SETVCLK2OFF		Set VCLK2 domain.
		0	*Read:* The VCLK2 domain is enabled. *Write:* The VCLK2 domain is unchanged.
		1	*Read:* The VCLK2 domain is disabled. *Write:* The VCLK2 domain is set to the enabled state.

Table 2-32. Clock Domain Disable Set Register (CDDISSET) Field Descriptions (continued)

Bit	Field	Value	Description
2	SETVCLKPOFF		Set VCLK_periph domain.
		0	*Read:* The VCLK_periph domain is enabled. *Write:* The VCLK_periph domain is unchanged.
		1	*Read:* The VCLK_periph domain is disabled. *Write:* The VCLK_periph domain is set to the enabled state.
1	SETHCLKOFF		Set HCLK and VCLK_sys domains.
		0	*Read:* The HCLK and VCLK_sys domain is enabled. *Write:* The HCLK and VCLK_sys domain is unchanged.
		1	*Read:* The HCLK and VCLK_sys domain is disabled. *Write:* The HCLK and VCLK_sys domain is set to the enabled state.
0	SETGCLKOFF		Set GCLK domain.
		0	*Read:* The GCLK domain is enabled. *Write:* The GCLK domain is unchanged.
		1	*Read:* The GCLK domain is disabled. *Write:* The GCLK domain is set to the enabled state.

2.5.1.15 Clock Domain Disable Clear Register (CDDISCLR)

The CDDISCLR register, shown in Figure 2-19 and described in Table 2-33, clears clock domains to the enabled state.

Figure 2-19. Clock Domain Disable Clear Register (CDDISCLR) [offset = 44h]

31 … 16
Reserved
R-0

15 … 12	11	10	9	8
Reserved	CLRVCLKA4 OFF	CLRVCLKA3 OFF	CLRVCLK4 OFF	CLRVCLK3 OFF
R-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

7	6	5	4	3	2	1	0
Reserved	CLRRTI1CLK OFF	Reserved	CLRVCLKA1 OFF	CLRVCLK2 OFF	CLRVCLKP OFF	CLRHCLK OFF	CLRGCLK OFF
R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

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

Table 2-33. Clock Domain Disable Clear Register (CDDISCLR) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reads return 0. Writes have no effect.
11-10	CLRVCLKA[4-3]OFF		Clear VCLKA[4-3] domain.
		0	*Read:* The VCLKA[4-3] domain is enabled. *Write:* The VCLKA[4-3] domain is unchanged.
		1	*Read:* The VCLKA[4-3] domain is disabled. *Write:* The VCLKA[4-3] domain is cleared to the enabled state.
9-8	CLRVCLK[4-3]OFF		Clear VCLK[4-3] domain.
		0	*Read:* The VCLK4-3] domain is enabled. *Write:* The VCLK[4-3] domain is unchanged.
		1	*Read:* The VCLK[4-3] domain is disabled. *Write:* The VCLK[4-3] domain is cleared to the enabled state.
7	Reserved	0-1	Reads return 0 or 1 and privilege mode writes allowed.
6	CLRRTI1CLKOFF		Clear RTICLK1 domain.
		0	*Read:* The RTICLK1 domain is enabled. *Write:* The RTICLK1 domain is unchanged.
		1	*Read:* The RTICLK1 domain is disabled. *Write:* The RTICLK1 domain is cleared to the enabled state.
5	Reserved	0-1	Reads return 0 or 1 and privilege mode writes allowed.
4	CLRVCLKA1OFF		Clear VCLKA1 domain.
		0	*Read:* The VCLKA1 domain is enabled. *Write:* The VCLKA1 domain is unchanged.
		1	*Read:* The VCLKA1 domain is disabled. *Write:* The VCLKA1 domain is cleared to the enabled state.
3	CLRVCLK2OFF		Clear VCLK2 domain.
		0	*Read:* The VCLK2 domain is enabled. *Write:* The VCLK2 domain is unchanged.
		1	*Read:* The VCLK2 domain is disabled. *Write:* The VCLK2 domain is cleared to the enabled state.

Table 2-33. Clock Domain Disable Clear Register (CDDISCLR) Field Descriptions (continued)

Bit	Field	Value	Description
2	CLRVCLKPOFF		Clear VCLK_periph domain.
		0	*Read:* The VCLK_periph domain is enabled. *Write:* The VCLK_periph domain is unchanged.
		1	*Read:* The VCLK_periph domain is disabled. *Write:* The VCLK_periph domain is cleared to the enabled state.
1	CLRHCLKOFF		Clear HCLK and VCLK_sys domains.
		0	*Read:* The HCLK and VCLK_sys domain is enabled. *Write:* The HCLK and VCLK_sys domain is unchanged.
		1	*Read:* The HCLK and VCLK_sys domain is disabled. *Write:* The HCLK and VCLK_sys domain is cleared to the enabled state.
0	CLRGCLKOFF		Clear GCLK domain.
		0	*Read:* The GCLK domain is enabled. *Write:* The GCLK domain is unchanged.
		1	*Read:* The GCLK domain is disabled. *Write:* The GCLK domain is cleared to the enabled state.

2.5.1.16 GCLK, HCLK, VCLK, and VCLK2 Source Register (GHVSRC)

The GHVSRC register, shown in Figure 2-20 and described in Table 2-34, controls the clock source configuration for the GCLK, HCLK, VCLK and VCLK2 clock domains.

Figure 2-20. GCLK, HCLK, VCLK, and VCLK2 Source Register (GHVSRC) [offset = 48h]

<table>
<tr><td>31</td><td>28</td><td>27</td><td>24</td><td>23</td><td>20</td><td>19</td><td>16</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="2">GHVWAKE</td><td colspan="2">Reserved</td><td colspan="2">HVLPM</td></tr>
<tr><td colspan="2">R-0</td><td colspan="2">R/WP-0</td><td colspan="2">R-0</td><td colspan="2">R/WP-0</td></tr>
</table>

<table>
<tr><td>15</td><td>4</td><td>3</td><td>0</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="2">GHVSRC</td></tr>
<tr><td colspan="2">R-0</td><td colspan="2">R/WP-0</td></tr>
</table>

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

Table 2-34. GCLK, HCLK, VCLK, and VCLK2 Source Register (GHVSRC) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reads return 0. Writes have no effect.
27-24	GHVWAKE		GCLK, HCLK, VCLK, VCLK2 source on wakeup.
		0	Clock source0 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		1h	Clock source1 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		2h	Clock source2 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		3h	Clock source3 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		4h	Clock source4 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		5h	Clock source5 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		6h	Clock source6 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		7h	Clock source7 is the source for GCLK, HCLK, VCLK, VCLK2 on wakeup.
		8h-Fh	Reserved
23-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	HVLPM		HCLK, VCLK, VCLK2 source on wakeup when GCLK is turned off.
		0	Clock source0 is the source for HCLK, VCLK, VCLK2 on wakeup.
		1h	Clock source1 is the source for HCLK, VCLK, VCLK2 on wakeup.
		2h	Clock source2 is the source for HCLK, VCLK, VCLK2 on wakeup.
		3h	Clock source3 is the source for HCLK, VCLK, VCLK2 on wakeup.
		4h	Clock source4 is the source for HCLK, VCLK, VCLK2 on wakeup.
		5h	Clock source5 is the source for HCLK, VCLK, VCLK2 on wakeup.
		6h	Clock source6 is the source for HCLK, VCLK, VCLK2 on wakeup.
		7h	Clock source7 is the source for HCLK, VCLK, VCLK2 on wakeup.
		8h-Fh	Reserved
15-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	GHVSRC		GCLK, HCLK, VCLK, VCLK2 current source. **Note: The GHVSRC[3-0] bits are updated with the HVLPM[3-0] setting when GCLK is turned off, and are updated with the GHVWAKE[3-0] setting on system wakeup.**
		0	Clock source0 is the source for GCLK, HCLK, VCLK, VCLK2.
		1h	Clock source1 is the source for GCLK, HCLK, VCLK, VCLK2.
		2h	Clock source2 is the source for GCLK, HCLK, VCLK, VCLK2.
		3h	Clock source3 is the source for GCLK, HCLK, VCLK, VCLK2.
		4h	Clock source4 is the source for GCLK, HCLK, VCLK, VCLK2.
		5h	Clock source5 is the source for GCLK, HCLK, VCLK, VCLK2.
		6h	Clock source6 is the source for GCLK, HCLK, VCLK, VCLK2.
		7h	Clock source7 is the source for GCLK, HCLK, VCLK, VCLK2.
		8h-Fh	Reserved

NOTE: Non implemented clock sources should not be enabled or used. A list of the available clock sources is shown in Table 2-28.

2.5.1.17 Peripheral Asynchronous Clock Source Register (VCLKASRC)

The VCLKASRC register, shown in Figure 2-21 and described in Table 2-35, sets the clock source for the asynchronous peripheral clock domains to be configured to run from a specific clock source.

Figure 2-21. Peripheral Asynchronous Clock Source Register (VCLKASRC) [offset = 4Ch]

31 … 16
Reserved
R-0

15 … 12	11 … 8	7 … 4	3 … 0
Reserved	Reserved	Reserved	VCLKA1S
R-0	R/WP-9h	R-0	R/WP-9h

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

Table 2-35. Peripheral Asynchronous Clock Source Register (VCLKASRC) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	VCLKA1S		Peripheral asynchronous clock1 source.
		0	Clock source0 is the source for peripheral asynchronous clock1.
		1h	Clock source1 is the source for peripheral asynchronous clock1.
		2h	Clock source2 is the source for peripheral asynchronous clock1.
		3h	Clock source3 is the source for peripheral asynchronous clock1.
		4h	Clock source4 is the source for peripheral asynchronous clock1.
		5h	Clock source5 is the source for peripheral asynchronous clock1.
		6h	Clock source6 is the source for peripheral asynchronous clock1.
		7h	Clock source7 is the source for peripheral asynchronous clock1.
		8h-Fh	VCLK is the source for peripheral asynchronous clock1.

NOTE: Non implemented clock sources should not be enabled or used. A list of the available clock sources is shown in Table 2-28.

2.5.1.18 RTI Clock Source Register (RCLKSRC)

The RCLKSRC register, shown in Figure 2-22 and described in Table 2-36, controls the RTI (Real Time Interrupt) clock source selection.

NOTE: Important constraint when the RTI clock source is not VCLK

If the RTIx clock source is chosen to be anything other than the default VCLK, then the RTI clock needs to be at least three times slower than the VCLK. This can be achieved by configuring the RTIxCLK divider in this register. This divider is internally bypassed when the RTIx clock source is VCLK.

Figure 2-22. RTI Clock Source Register (RCLKSRC) [offset = 50h]

31 — 26	25 — 24	23 — 20	19 — 16
Reserved	Reserved	Reserved	Reserved
R-0	R/WP-1h	R-0	R/WP-9h

15 — 10	9 — 8	7 — 4	3 — 0
Reserved	RTI1DIV	Reserved	RTI1SRC
R-0	R/WP-1h	R-0	R/WP-9h

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

Table 2-36. RTI Clock Source Register (RCLKSRC) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	Reads return 0. Writes have no effect.
25-24	Reserved	0	Reads return value and privilege mode writes allowed.
23-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	Reserved	0	Reads return value and privilege mode writes allowed.
15-10	Reserved	0	Reads return 0. Writes have no effect.
9-8	RTI1DIV		RTI clock1 Divider.
		0	RTICLK1 divider value is 1.
		1h	RTICLK1 divider value is 2.
		2h	RTICLK1 divider value is 4.
		3h	RTICLK1 divider value is 8.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	RTI1SRC		RTI clock1 source.
		0	Clock source0 is the source for RTICLK1.
		1h	Clock source1 is the source for RTICLK1.
		2h	Clock source2 is the source for RTICLK1.
		3h	Clock source3 is the source for RTICLK1.
		4h	Clock source4 is the source for RTICLK1.
		5h	Clock source5 is the source for RTICLK1.
		6h	Clock source6 is the source for RTICLK1.
		7h	Clock source7 is the source for RTICLK1.
		8h-Fh	VCLK is the source for RTICLK1.

NOTE: A list of the available clock sources is shown in Table 2-28.

2.5.1.19 Clock Source Valid Status Register (CSVSTAT)

The CSVSTAT register, shown in Figure 2-23 and described in Table 2-37, indicates the status of usable clock sources.

Figure 2-23. Clock Source Valid Status Register (CSVSTAT) [offset = 54h]

31	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
CLKSR7V	CLKSR6V	CLKSR5V	CLKSR4V	CLKSR3V	Reserved	CLKSR1V	CLKSR0V
R-1	R-0	R-0	R-1	R-1	R-0	R-0	R-0

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

Table 2-37. Clock Source Valid Register (CSVSTAT) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved.	0	Reads return 0. Writes have no effect.
7-3	CLKSR[7-3]V		Clock source[7-3] valid.
		0	Clock source[7-3] is not valid.
		1	Clock source[7-3] is valid.
			Note: If the valid bit of the source of a clock domain is not set (that is, the clock source is not fully stable), the respective clock domain is disabled by the Global Clock Module (GCM).
2	Reserved.	0	Reads return 0. Writes have no effect.
1-0	CLKSR[1-0]V		Clock source[1-0] valid.
		0	Clock source[1-0] is not valid.
		1	Clock source[1-0] is valid.
			Note: If the valid bit of the source of a clock domain is not set (that is, the clock source is not fully stable), the respective clock domain is disabled.

NOTE: A list of the available clock sources is shown in Table 2-28.

2.5.1.20 Memory Self-Test Global Control Register (MSTGCR)

The MSTGCR register, shown in Figure 2-24 and described in Table 2-38, controls several aspects of the PBIST (Programmable Built-In Self Test) memory controller.

Figure 2-24. Memory Self-Test Global Control Register (MSTGCR) [offset = 58h]

31 24	23 16
Reserved	MBIST_ALGSEL
R-0	R/WP-0

15 10	9 8	7 4	3 0
Reserved	ROM_DIV	Reserved	MSTGENA
R-0	R/WP-0	R-0	R/WP-5h

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

Table 2-38. Memory Self-Test Global Control Register (MSTGCR) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23-16	MBIST_ALGSEL		Selects different test algorithm for PBIST
		0	No Algorithm is selected.
		1h	Checkerboard is selected.
		2h	March 13N with background of all 0s and all 1s is selected.
		4h	March 11N with background of hex 5 and As is selected.
		8h	March 13N with backgrounds of hex 3 and Cs, hex 0F and F0s, and 69 and 96s are selected.
		10h	PMOS Open Address Decode is selected.
		20h	No Algorithm is selected.
		40h	No Algorithm is selected.
		80h	No Algorithm is selected.
15-10	Reserved	0	Reads return 0. Writes have no effect.
9-8	ROM_DIV		Prescaler divider bits for ROM clock source.
		0	ROM clock source is HCLK divided by 1. PBIST will reset for 16 VBUS cycles.
		1h	ROM clock source is HCLK divided by 2. PBIST will reset for 32 VBUS cycles.
		2h	ROM clock source is HCLK divided by 4. PBIST will reset for 64 VBUS cycles.
		3h	ROM clock source is HCLK divided by 8. PBIST will reset for 96 VBUS cycles.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	MSTGENA		Memory self-test controller global enable key **Note: Enabling the MSTGENA key will generate a reset to the state machine of the selected PBIST controller.**
		Ah	Memory self-test controller is enabled.
		Others	Memory self-test controller is disabled. **Note: It is recommended that a value of 5h be used to disable the memory self-test controller. This value will give maximum protection from a bit flip inducing event that would inadvertently enable the controller.**

2.5.1.21 Memory Hardware Initialization Global Control Register (MINITGCR)

The MINITGCR register, shown in Figure 2-25 and described in Table 2-39, enables automatic hardware memory initialization.

Figure 2-25. Memory Hardware Initialization Global Control Register (MINITGCR) [offset = 5Ch]

31	16
Reserved	
R-0	

15	4	3	0
Reserved		MINITGENA	
R-0		R/WP-5h	

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

Table 2-39. Memory Hardware Initialization Global Control Register (MINITGCR) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	MINITGENA		Memory hardware initialization global enable key.
		Ah	Global memory hardware initialization is enabled.
		Others	Global memory hardware initialization is disabled.
			Note: It is recommended that a value of 5h be used to disable memory hardware initialization. This value will give maximum protection from an event that would inadvertently enable the controller.

2.5.1.22 MBIST Controller/ Memory Initialization Enable Register (MSINENA)

The MSINENA register, shown in Figure 2-26 and described in Table 2-40, enables PBIST controllers for memory self test and the memory modules initialized during automatic hardware memory initialization.

Figure 2-26. MBIST Controller/Memory Initialization Enable Register (MSINENA) [offset = 60h]

31 … 16
MSIENA
R/WP-0

15 … 0
MSIENA
R/WP-0

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

Table 2-40. MBIST Controller/Memory Initialization Enable Register (MSINENA) Field Descriptions

Bit	Field	Value	Description
31-0	MSIENA		PBIST controller and memory initialization enable register. In memory self-test mode, all the corresponding bits of the memories to be tested should be set before enabling the global memory self-test controller key (MSTGENA) in the MSTGCR register (offset 58h). The reason for this is that MSTGENA, in addition to being the global enable for all individual PBIST controllers, is the source for the reset generation to all the PBIST controller state machines. Disabling the MSTGENA or MINITGENA key (by writing from Ah to any other value) will reset all the MSIENA[31-0] bits to their default values.

		0	In memory self-test mode (MSTGENA = Ah):
			PBIST controller [31-0] is disabled.
			In memory Initialization mode (MINITGENA = Ah):
			Memory module [31-0] auto hardware initialization is disabled.

		1	In memory self-test mode (MSTGENA = Ah): PBIST controller [31-0] is enabled. In memory Initialization mode (MINITGENA = Ah): Memory module [31-0] auto hardware initialization is enabled. **Note: Software should ensure that both the memory self-test global enable key (MSTGENA) and the memory hardware initialization global key (MINITGENA) are not enabled at the same time.**

2.5.1.23 Memory Self-Test Fail Status Register (MSTFAIL)

The MSTFAIL register, shown in Figure 2-27 and described in Table 2-41, shows the fail status of the memory self-tests.

Figure 2-27. Memory Self-Test Fail Status Register (MSTFAIL) [offset = 64h]

31	16
MSTF	
R/WP-0	

15	0
MSTF	
R/WP-0	

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

Table 2-41. Memory Self-Test Fail Status Register (MSTFAIL) Field Descriptions

Bit	Field	Value	Description
31-0	MSTF		Memory self-test fail status bit.
		0	*Read:* PBIST controller [31-0] run did not fail. *Write:* A write of 0 has no effect.
		1	*Read:* PBIST controller [31-0] run failed. *Write:* The bit is cleared to 0. **Note: Disabling the MSTGENA key (by writing from Ah to any other value) will reset all the individual fail status bits to their default values.**

2.5.1.24 MSTC Global Status Register (MSTCGSTAT)

The MSTCGSTAT register, shown in Figure 2-28 and described in Table 2-42, shows the status of the memory hardware initialization and the memory seft-test.

Figure 2-28. MSTC Global Status Register (MSTCGSTAT) [offset = 68h]

31–16
Reserved
R-0

15–9	8	7–1	0
Reserved	MINIDONE	Reserved	MSTDONE
R-0	R/WPC-0	R-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; C = Clear; WP = Write in privileged mode only; -*n* = value after reset

Table 2-42. MSTC Global Status Register (MSTCGSTAT) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reads return 0. Writes have no effect.
8	MINIDONE		Memory hardware initialization complete status. **Note: Disabling the MINITGENA key (By writing from Ah to any other value) will clear the MINIDONE status bit to 0.** **Note: Individual memory initialization status is shown in the MINISTAT register.**
		0	*Read:* Memory hardware initialization is not complete for all memory. *Write:* A write of 0 has no effect.
		1	*Read:* Hardware initialization of all memory is completed. *Write:* The bit is cleared to 0.
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	MSTDONE		Memory self-test run complete status. **Note: Disabling the MSTGENA key (by writing from Ah to any other value) will clear the MSTDONE status bit to 0.**
		0	*Read:* Memory self-test is not completed. *Write:* A write of 0 has no effect.
		1	*Read:* Memory self-test is completed. *Write:* The bit is cleared to 0.

2.5.1.25 Memory Hardware Initialization Status Register (MINISTAT)

The MINISTAT register, shown in Figure 2-29 and described in Table 2-43, indicates the status of hardware memory initialization.

Figure 2-29. Memory Hardware Initialization Status Register (MINISTAT) [offset = 6Ch]

31 16
MIDONE
R/WP-0

15 0
MIDONE
R/WP-0

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

Table 2-43. Memory Hardware Initialization Status Register (MINISTAT) Field Descriptions

Bit	Field	Value	Description
31-0	MIDONE		Memory hardware initialization status bit.
		0	*Read:* Memory module[31-0] hardware initialization is not completed. *Write:* A write of 0 has no effect.
		1	*Read:* Memory module[31-0] hardware initialization is completed. *Write:* The bit is cleared to 0. **Note: Disabling the MINITGENA key (by writing from Ah to any other value) will reset all the individual status bits to 0.**

2.5.1.26 PLL Control Register 1 (PLLCTL1)

The PLLCTL1 register, shown in Figure 2-30 and described in Table 2-44, controls the output frequency of PLL1 (Clock Source 1 - FMzPLL). It also controls the behavior of the device if a PLL slip or oscillator failure is detected.

Figure 2-30. PLL Control Register 1 (PLLCTL1) [offset = 70h]

31	30 29	28 24
ROS	MASK_SLIP	PLLDIV
R/WP-0	R/WP-1h	R/WP-Fh

23	22	21 16
ROF	Reserved	REFCLKDIV
R/WP-0	R-0	R/WP-2h

15 0
PLLMUL
R/WP-5F00h

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

Table 2-44. PLL Control Register 1 (PLLCTL1) Field Descriptions

Bit	Field	Value	Description
31	ROS		Reset on PLL Slip
		0	Do not reset system when PLL slip is detected.
		1	Reset when PLL slip is detected .
			Note: BPOS (Bits 30-29) must also be enabled for ROS to be enabled.
30-29	MASK_SLIP		Mask detection of PLL slip
		2h	Bypass on PLL Slip is disabled. If a PLL Slip is detected no action is taken.
		Others	Bypass on PLL Slip is enabled. If a PLL Slip is detected the device will automatically bypass the PLL and use the oscillator to provide the device clock.
			Note: If ROS (Bit 31) is set to 1, the device will be reset if a PLL Slip and the PLL will be bypassed after the reset occurs.
28-24	PLLDIV		PLL Output Clock Divider
			R = PLLDIV + 1 $f_{PLL\ CLK} = f_{post\_ODCLK} / R$
		0	$f_{PLL\ CLK} = f_{post\text{-}ODCLK} / 1$
		1h	$f_{PLL\ CLK} = f_{post\text{-}ODCLK} / 2$
		:	:
		1Fh	$f_{PLL\ CLK} = f_{post\text{-}ODCLK} / 32$
23	ROF		Reset on Oscillator Fail
		0	Do not reset system when oscillator is out of range.
		1	Reset when oscillator is out of range.
22	Reserved	0	Value has no effect on PLL operation.
21-16	REFCLKDIV		Reference Clock Divider
			NR = REFCLKDIV + 1 $f_{INT\ CLK} = f_{OSCIN} / NR$
		0	$f_{INT\ CLK} = f_{OSCIN} / 1$
		1h	$f_{INT\ CLK} = f_{OSCIN} / 2$
		:	:
		3Fh	$f_{INT\ CLK} = f_{OSCIN} / 64$
15-0	PLLMUL		PLL Multiplication Factor
			NF = (PLLMUL / 256) + 1, valid multiplication factors are from 1 to 256. $f_{VCO\ CLK} = f_{INT\ CLK} \times NF$
		0h	$f_{VCO\ CLK} = f_{INT\ CLK} \times 1$
		100h	$f_{VCO\ CLK} = f_{INT\ CLK} \times 2$
		:	:
		5B00h	$f_{VCO\ CLK} = f_{INT\ CLK} \times 92$
		5C00h	$f_{VCO\ CLK} = f_{INT\ CLK} \times 93$
		:	:
		FF00h	$f_{VCO\ CLK} = f_{INT\ CLK} \times 256$

2.5.1.27 PLL Control Register 2 (PLLCTL2)

The PLLCTL2 register, shown in Figure 2-31 and described in Table 2-45, controls the modulation characteristics and the output divider of the PLL.

Figure 2-31. PLL Control Register 2 (PLLCTL2) [offset = 74h]

31	30	22	21	20	16
FMENA	SPREADINGRATE		Rsvd	MULMOD	
R/WP-0	R/WP-1FFh		R-0	R/WP-0	

15	12	11	9	8	0
MULMOD		ODPLL		SPR_AMOUNT	
R/WP-7h		R/WP-1h		R/WP-0	

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

Table 2-45. PLL Control Register 2 (PLLCTL2) Field Descriptions

Bit	Field	Value	Description
31	FMENA		Frequency Modulation Enable.
		0	Disable frequency modulation.
		1	Enable frequency modulation.
30-22	SPREADINGRATE		NS = SPREADINGRATE + 1 $f_{mod} = f_{s} = f_{INT\ CLK}/(2 \times NS)$
		0	$f_{mod} = f_{s} = f_{INT\ CLK} / (2 \times 1)$
		1h	$f_{mod} = f_{s} = f_{INT\ CLK} / (2 \times 2)$
		:	:
		1FFh	$f_{mod} = f_{s} = f_{INT\ CLK} / (2 \times 512)$
21	Reserved	0	Value has no effect on PLL operation.
20-12	MULMOD		Multiplier Correction when Frequency Modulation is enabled When FMENA = 0, MUL_when_MOD = 0; when FMENA = 1, MUL_when_MOD = (MULMOD / 256)
		0	No adder to NF
		8h	MUL_when_MOD = 8/256
		9h	MUL_when_MOD = 9/256
		:	:
		1FFh	MUL_when_MOD = 511/256
11-9	ODPLL		Internal PLL Output Divider. OD = ODPLL + 1 $f_{post\text{-}ODCLK} = f_{VCO\ CLK}/OD$ Note: PLL output clock is gated off if ODPLL is changed while the PLL is active.
		0	$f_{post\text{-}ODCLK} = f_{VCO\ CLK} / 1$
		1h	$f_{post\text{-}ODCLK} = f_{VCO\ CLK} / 2$
		:	:
		7h	$f_{post\text{-}ODCLK} = f_{VCO\ CLK} / 8$ **Note: PLL output clock is gated off, if ODPLL is changed while the PLL is active.**

Table 2-45. PLL Control Register 2 (PLLCTL2) Field Descriptions (continued)

Bit	Field	Value	Description
8-0	SPR_AMOUNT		Spreading Amount. NV = (SPR_AMOUNT + 1)/2048 NV ranges from 1/2048 to 512/2048 Note that the PLL output clock is disabled for 1 modulation period if the SPR_AMOUNT field is changed while the frequency modulation is enabled. If frequency modulation is disabled and SPR_AMOUNT is changed, there is no effect on the PLL output clock.
		0	NV = 1/2048
		1h	NV = 2/2048
		:	:
		1FFh	NV = 512/2048

2.5.1.28 SYS Pin Control Register 10 (SYSPC10)

The SYSPC10 register, shown in Figure 2-32 and described in Table 2-46, controls the function of the ECPCLK slew mode.

Figure 2-32. SYS Pin Control Register 10 (SYSPC10) [offset = 78h]

31-16
Reserved
R-0

15-1	0
Reserved	ECPCLK_SLEW
R-0	R/W-0

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

Table 2-46. SYS Pin Control Register 10 (SYSPC10) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	ECPCLK_SLEW		ECPCLK slew control. This bit controls between the fast or slow slew mode.
		0	Fast mode is enabled; the normal output buffer is used for this pin.
		1	Slow mode is enabled; slew rate control is used for this pin.

2.5.1.29 Die Identification Register Lower Word (DIEIDL)

The DIEIDL register, shown in Figure 2-33 and described in Table 2-47, contains information about the die lot number, wafer number, and X, Y wafer coordinates.

Figure 2-33. Die Identification Register, Lower Word (DIEIDL) [offset = 7Ch]

31 — 24	23 — 16
WAFER #	Y WAFER COORDINATE
R-D	R-D

15 — 12	11 — 0
Y WAFER COORDINATE	X WAFER COORDINATE
R-D	R-D

LEGEND: R = Read only; -*n* = value after reset; D = value is device specific

Table 2-47. Die Identification Register, Lower Word (DIEIDL) Field Descriptions

Bit	Field	Description
31-24	WAFER #	These read-only bits contain the wafer number of the device.
23-12	Y WAFER COORDINATE	These read-only bits contain the Y wafer coordinate of the device.
11-0	X WAFER COORDINATE	These read-only bits contain the X wafer coordinate of the device.

NOTE: Die Identification Information

The die identification information will vary from unit to unit. This information is programmed by TI as part of the initial device test procedure.

2.5.1.30 Die Identification Register Upper Word (DIEIDH)

The DIEIDH register, shown in Figure 2-34 and described in Table 2-48, contains information about the die lot number.

Figure 2-34. Die Identification Register, Upper Word (DIEIDH) [offset = 80h]

31 — 24	23 — 16
Reserved	LOT #
R-0	R-D

15 — 0
LOT #
R-D

LEGEND: R = Read only; -*n* = value after reset; D = value is device specific

Table 2-48. Die Identification Register, Upper Word (DIEIDH) Field Descriptions

Bit	Field	Description
31-24	Reserved	Reserved for TI use. Writes have no effect.
23-0	LOT #	This read-only register contains the device lot number.

NOTE: Die Identification Information

The die identification information will vary from unit to unit. This information is programmed by TI as part of the initial device test procedure.

2.5.1.31 LPO/Clock Monitor Control Register (LPOMONCTL)

The LPOMONCTL register, shown in Figure 2-35 and described in Table 2-49, controls the Low Frequency (Clock Source 4) and High Frequency (Clock Source 5) Low Power Oscillator's trim values.

Figure 2-35. LPO/Clock Monitor Control Register (LPOMONCTL) [offset = 88h]

31 25	24	23 17	16
Reserved	BIAS ENABLE	Reserved	OSCFRQCONFIGCNT
R-0	R/WP-1	R-0	R-0

15 13	12 8	7 5	4 0
Reserved	HFTRIM	Reserved	LFTRIM
R-0	R/WP-10h	R-0	R/WP-10h

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

Table 2-49. LPO/Clock Monitor Control Register (LPOMONCTL) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	BIAS ENABLE		Bias enable.
		0	The bias circuit inside the low-power oscillator (LPO) is disabled.
		1	The bias circuit inside LPO is enabled.
23-17	Reserved	0	Reads return 0. Writes have no effect.
16	OSCFRQCONFIGCNT		Configures the counter based on OSC frequency.
		0	OSC freq is ≤ 20MHz
		1	OSC freq is > 20MHz and ≤ 80MHz
15-13	Reserved	0	Reads return 0. Writes have no effect.

Table 2-49. LPO/Clock Monitor Control Register (LPOMONCTL) Field Descriptions (continued)

Bit	Field	Value	Description
12-8	HFTRIM		High-frequency oscillator trim value. This four-bit value is used to center the HF oscillator's frequency. **Caution: This value should only be changed when the HF oscillator is not the source for a clock domain, otherwise a system failure could result.** The following values are the ratio: f / fo in the F021 process.
		0	29.52
		1h	34.24%
		2h	38.85%
		3h	43.45%
		4h	47.99%
		5h	52.55%
		6h	57.02%
		7h	61.46%
		8h	65.92%
		9h	70.17
		Ah	74.55%
		Bh	78.92%
		Ch	83.17%
		Dh	87.43%
		Eh	91.75%
		Fh	95.89%
		10h	100.00% (Default at Reset)
		11h	104.09
		12h	108.17
		13h	112.32
		14h	116.41
		15h	120.67
		16h	124.42
		17h	128.38
		18h	132.24
		19h	136.15
		1Ah	140.15
		1Bh	143.94
		1Ch	148.02
		1Dh	151.80x
		1Eh	155.50x
		1Fh	159.35%
7-5	Reserved	0	Reads return 0. Writes have no effect.

Table 2-49. LPO/Clock Monitor Control Register (LPOMONCTL) Field Descriptions (continued)

Bit	Field	Value	Description
4-0	LFTRIM		Low-frequency oscillator trim value. This four-bit value is used to center the LF oscillator's frequency. **Caution: This value should only be changed when the LF oscillator is not the source for a clock domain, otherwise a system failure could result.** The following values are the ratio: f / fo in the F021 process.
		0	20.67
		1h	25.76
		2h	30.84
		3h	35.90
		4h	40.93
		5h	45.95
		6h	50.97
		7h	55.91
		8h	60.86
		9h	65.78
		Ah	70.75
		Bh	75.63
		Ch	80.61
		Dh	85.39
		Eh	90.23
		Fh	95.11
		10h	100.00% (Default at Reset)
		11h	104.84
		12h	109.51
		13h	114.31
		14h	119.01
		15h	123.75
		16h	128.62
		17h	133.31
		18h	138.03
		19h	142.75
		1Ah	147.32
		1Bh	152.02
		1Ch	156.63
		1Dh	161.38
		1Eh	165.90
		1Fh	170.42

2.5.1.32 Clock Test Register (CLKTEST)

The CLKTEST register, shown in Figure 2-36 and described in Table 2-50, controls the clock signal that is supplied to the ECLK pin for test and debug purposes.

NOTE: Clock Test Register Usage

This register should only be used for test and debug purposes.

Figure 2-36. Clock Test Register (CLKTEST) [offset = 8Ch]

31 27	26	25	24
Reserved	ALTLIMPCLOCK ENABLE	RANGEDET CTRL	RANGEDET ENASSEL
R-0	R/WP-0	R/WP-0	R/WP-0

23 20	19 16
Reserved	CLK_TEST_EN
R-0	R/WP-Ah

15 12	11 8	7 4	3 0
Reserved	SEL_GIO_PIN	Reserved	SEL_ECP_PIN
R-0	R/WP-0	R-0	R/WP-0

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

Table 2-50. Clock Test Register (CLKTEST) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reads return 0. Writes have no effect.
26	ALTLIMPCLOCKENABLE		This bit selects a clock driven by the GIOB[0] pin as an alternate limp clock to the clock monitor phase frequency detect (PFD).
		0	The 10-MHz LPO fast clock is the compare clock for the clock detect PFD circuit and the source to limp clock on a clock fail.
		1	The ALTLIMPCLOCK driven on the GIOB[0] pin is the compare clock for the clock detect PFD circuit and the source to limp clock on a clock fail.
25	RANGEDETCTRL		Range detection control. This bit's functionality is dependant on the state of the RANGEDETENSSEL bit (Bit 24) of the CLKTEST register.
		0	The clock monitor range detection circuitry (RANGEDETECTENABLE) is disabled.
		1	The clock monitor range detection circuitry (RANGEDETECTENABLE) is enabled.
24	RANGEDETENASSEL		Selects range detection enable. This bit resets asynchronously on power on reset.
		0	The range detect enable is generated by the hardware in the clock monitor wrapper.
		1	The range detect enable is controlled by the RANGEDETCTRL bit (Bit 25) of the CLKTEST register.
23-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	CLK_TEST_EN		Clock test enable. This bit enables the clock going to the ECLK pin. **Note: The ECLK pin must also be placed into Functional mode by setting the ECPCLKFUN bit to 1 in the SYSPC1 register.**
		5h	Clock going to ECLK pin is enabled.
		Others	Clock going to ECLK pin is disabled.
15-12	Reserved	0	Reads return 0. Writes have no effect.

Table 2-50. Clock Test Register (CLKTEST) Field Descriptions (continued)

Bit	Field	Value	Description
11-8	SEL_GIO_PIN		GIOB[0] pin clock source valid, clock source select
		0	Oscillator valid status
		1h	PLL1 valid status
		2h-4h	Reserved
		5h	High-frequency LPO (Low-Power Oscillator) clock valid status
		6h	PLL2 valid status
		7h	Reserved
		8h	Low-frequency LPO (Low-Power Oscillator) clock valid status
		9h-Ch	Oscillator valid status
		Dh	VCLKA3
		Eh	VCLKA4
		Fh	Oscillator valid status
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	SEL_ECP_PIN		ECLK pin clock source select **Note: Only valid clock sources can be selected for the ECLK pin. Valid clock sources are displayed by the CSVSTAT register.**
		0	Oscillator clock
		1h	PLL1 clock output
		2h	Reserved
		3h	EXTCLKIN1
		4h	Low-frequency LPO (Low-Power Oscillator) clock
		5h	High-frequency LPO (Low-Power Oscillator) clock
		6h	Secondary PLL free-running clock output
		7h	EXTCLKIN2
		8h	GCLK
		9h	RTI Base
		Ah	Reserved
		Bh	VCLKA1
		Ch	Reserved
		Dh	VCLKA3_DIVR
		Eh	VCLKA4
		Fh	Reserved

NOTE: Nonimplemented clock sources should not be enabled or used.

2.5.1.33 DFT Control Register (DFTCTRLREG)

This register is shown in Figure 2-37 and described in Table 2-51.

Figure 2-37. DFT Control Register (DFTCTRLREG) [offset = 90h]

31	16
Reserved	
R-0	

15	14	13	12	11	10	9	8	7	4	3	0
Reserved		DFTWRITE		Reserved		DFTREAD		Reserved		TEST_MODE_KEY	
R-0		R/WP-1h		R-0		R/WP-1h		R-0		R/WP-5h	

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

Table 2-51. DFT Control Register (DFTCTRLREG) Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Reads return 0. Writes have no effect.
13-12	DFTWRITE		DFT logic access. For F021: DFTWRITE[0] = 0 and DFTREAD[0] = 0 configured in stress mode DFTWRITE[1] = 0 and DFTREAD[1] = 0 configured in stress mode DFTWRITE[0] = 0 and DFTREAD[0] = 0 configured in fast mode DFTWRITE[1] = 1 and DFTREAD[1] = 1 configured in fast mode DFTWRITE[0] = 1 and DFTREAD[0] = 1 configured in slow mode DFTWRITE[1] = 0 and DFTREAD[1] = 0 configured in slow mode DFTWRITE[0] = 1 and DFTREAD[0] = 1 configured in screen mode DFTWRITE[1] = 1 and DFTREAD[1] = 1 configured in screen mode
11-10	Reserved	0	Reads return 0. Writes have no effect.
9-8	DFTREAD		DFT logic access. For F021: DFTWRITE[0] = 0 and DFTREAD[0] = 0 configured in stress mode DFTWRITE[1] = 0 and DFTREAD[1] = 0 configured in stress mode DFTWRITE[0] = 0 and DFTREAD[0] = 0 configured in fast mode DFTWRITE[1] = 1 and DFTREAD[1] = 1 configured in fast mode DFTWRITE[0] = 1 and DFTREAD[0] = 1 configured in slow mode DFTWRITE[1] = 0 and DFTREAD[1] = 0 configured in slow mode DFTWRITE[0] = 1 and DFTREAD[0] = 1 configured in screen mode DFTWRITE[1] = 1 and DFTREAD[1] = 1 configured in screen mode
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	TEST_MODE_KEY		Test mode key. This register is for internal TI use only.
		0-Fh (except Ah)	Register key disable. All bits in this register will maintain their default value and cannot be written.
		Ah	Register key enable. All the bits can be written to only when the key is enabled. On reset, these bits will be set to 5h.

2.5.1.34 DFT Control Register 2 (DFTCTRLREG2)

This register is shown in Figure 2-38 and described in Table 2-52.

Figure 2-38. DFT Control Register 2 (DFTCTRLREG2) [offset = 94h]

31	16
IMPDF(27:12)	
R/WP-0	

15	4	3	0
IMPDF(11:0)		TEST_MODE_KEY	
R/WP-0		R/WP-5h	

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

Table 2-52. DFT Control Register 2 (DFTCTRLREG2) Field Descriptions

Bit	Field	Value	Description
31-4	IMPDF[27:0]		DFT Implementation defined bits.
		0	IMPDF[27:0] - Disabled
		1	IMPDF[27:0] - Enabled
3-0	TEST_MODE_KEY		Test mode key. This register is for internal TI use only.
		0-Fh (except Ah)	Register key disable. All bits in this register will maintain their default value and cannot be written.
		Ah	Register key enable. All the bits can be written to only when the key is enabled.

2.5.1.35 General Purpose Register (GPREG1)

This register is shown in Figure 2-39 and described in Table 2-53.

Figure 2-39. General Purpose Register (GPREG1) [offset = A0h]

31	30 … 26	25 … 20	19 … 16
EMIF_FUNC	Reserved	PLL1_FBSLIP_FILTER_COUNT	PLL1_FBSLIP_FILTER_KEY
R/WP-0	R-0	R/WP-0	R/WP-5h

15 … 0
OUTPUT_BUFFER_LOW_EMI_MODE
R/WP-FFFFh

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

Table 2-53. General Purpose Register (GPREG1) Field Descriptions

Bit	Field	Value	Description
31	EMIF_FUNC	0	Enable EMIF functions to be output. EMIF_ADDR[0], EMIF_ADDR[1], EMIF_ADDR[6], EMIF_ADDR[7], EMIF_ADDR[8], EMIF_BA[1], EMIF_nCS[0], and EMIF_nCS[3] are multiplexed with N2HET2 signals. By default, these terminals are tri-stated and pulled down. Any application that requires the EMIF functionality must set the EMIF_FUNC bit. This allows these 8 EMIF module outputs to be driven on to the assigned balls.
30-26	Reserved	0	Reads return 0. Writes have no effect.
25-20	PLL1_FBSLIP_FILTER_ COUNT		FBSLIP down counter programmed value.
		0-3Fh	Configures the system response when a FBSLIP is indicated by the PLL macro. When PLL1_RFSLIP_FILTER_KEY is not Ah, the down counter counts from the programmed value on every LPO high frequency clock once PLL macro indicates FBSLIP. When the count reaches 0, if the synchronized FBSLIP signal is still high, an FBSLIP condition is indicated to the system module and is captured in the global status register. When the FBSLIP signal from the PLL macro is de-asserted before the count reaches 0, the counter is reloaded with the programmed value. On reset, counter value is 0. Counter must be programmed to a non-zero value and enabled for the filtering to be enabled.
		0	Filtering is disabled.
		1h	Filtering is enabled. Every slip is recognized.
		2h	Filtering is enabled. The slip must be at least 2 HF LPO cycles wide in order to be recognized as a slip.
		:	:
		3Fh	Filtering is enabled. The slip must be at least 63 HF LPO cycles wide in order to be recognized as a slip.
19-16	PLL1_FBSLIP_FILTER_KEY		Enable the FBSLIP filtering.
		5h	On reset, the FBSLIP filter is disabled and the FBSLIP passes through.
		Fh	This is an unsupported value. You should avoid writing this value to this bit field.
		All other values	FBSLIP filtering is enabled. Recommended to program Ah in this bit field. Enabling of the FBSLIP occurs when the KEY is programmed and a non-zero value is present in the COUNT field.

Table 2-53. General Purpose Register (GPREG1) Field Descriptions (continued)

Bit	Field	Value	Description
15-0	OUTPUT_BUFFER_LOW_EMI_MODE		Control field for the low-EMI mode of output buffers for module/signals: bit[0] controls MiBSPI1 bit[1] controls SPI2 bit[2] controls MiBSPI3 bit[3] controls SPI4 bit[4] controls MiBSPI5 bit[5] Reserved bit[6] controls EMIF bit[7] Reserved bit[8] controls signal TMS bit[9] controls signal TDI bit[10] controls signal TDO bit[11] controls signal RTCK bit[12] controls signal TEST bit[13] controls signal nERROR bit[14] controls signal ADEVT bit[15] Reserved
		0	Enable EMI mode for each connected output buffers.
		1h-FFFEh	Enable/Disable EMI mode for connected output buffers.
		FFFFh	Disable EMI mode for each connected output buffer.

2.5.1.36 Imprecise Fault Status Register (IMPFASTS)

The IMPFASTS register, shown in Figure 2-40 and described in Table 2-54, displays information about imprecise aborts that have occurred.

Figure 2-40. Imprecise Fault Status Register (IMPFASTS) [offset = A8h]

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

15-11	10	9	8	7-1	0
Reserved	EMIFA	NCBA	VBUSA	Reserved	ATYPE
R-0	R-0	R-0	R-0	R-0	R/WC-0

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

Table 2-54. Imprecise Fault Status Register (IMPFASTS) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23-16	MASTERID	0-FFh	Master ID. This register indicates which master is responsible for the imprecise abort. The master ID value depends on device implementation- see Table 2-2 for MASTERID values for each bus master. **Notes:** • These bits are only updated when an imprecise abort occurs • These bits are cleared to 0 only on power-on reset. The value of these bits remains unchanged after all other resets.
15-11	Reserved	0	Reads return 0. Writes have no effect.
10	EMIFA		EMIF imprecise abort. This register indicates the imprecise abort was generated writing into the EMIF. **Notes:** • This bit is only updated when an imprecise abort occurs • This bit is cleared to 0 only on power-on reset. The value of this register remains unchanged after all other resets.
		0	EMIF did not generate the last imprecise abort.
		1	EMIF was written with an illegal address and generated an imprecise abort.
9	NCBA		Non-cacheable, bufferable abort (NCBA). This register indicates the imprecise abort was generated by a non-cacheable, bufferable write or shared device write through the write buffer of the CPU. **Notes:** • This bit is only updated when an imprecise abort generated by a non-cacheable, bufferable write or shared device write occurs. • This bit is cleared to 0 only on power-on reset. The value of this register remains unchanged after all other resets.
		0	A NCBA is not responsible for the last imprecise abort.
		1	A NCBA was written with an illegal address and generated an imprecise abort.
8	VBUSA		VBUS abort. This register indicates the imprecise abort was generated when writing into the peripheral frame. **Notes:** • This bit is only updated when an imprecise abort is generated when writing into the peripheral frame • This bit is cleared to 0 only on power-on reset. The value of this register remains unchanged after all other resets.
		0	The peripheral frame did not generate the last imprecise abort.
		1	The peripheral frame was written with an illegal address and generated an imprecise abort.
7-1	Reserved	0	Reads return 0. Writes have no effect.

Table 2-54. Imprecise Fault Status Register (IMPFASTS) Field Descriptions (continued)

Bit	Field	Value	Description
0	ATYPE		Abort type. This bit indicates to the CPU whether the last abort was an imprecise abort or a precise abort. **Notes:** • This bit is updated after each abort is generated to the CPU. • This bit is cleared on CPU read. • This bit is cleared to 0 only on power-on reset. The value of this bit remains unchanged after all other resets
		0	The last abort generated was a precise abort. MASTERID, VBUSA, NCBA, EMIFA and IMPFTADD were not updated.
		1	The last abort generated was an imprecise abort. MASTERID, VBUSA, NCBA, EMIFA and IMPFTADD were updated. **Note: Once ATYPE is set, the IMPFAWADD and IMPFASTS bits are not updated by subsequent ABORT signals.**

NOTE: The DMA and the peripheral master port will also generate an imprecise abort to the CPU when writing to the peripheral region or to the EMIF region. This will be indicated in the Master ID field of this register.

2.5.1.37 Imprecise Fault Address Register (IMPFTADD)

This IMPFTADD register, shown in Figure 2-41 and described in Table 2-55, shows the address at which an imprecise abort occurred.

Figure 2-41. Imprecise Fault Write Address Register (IMPFTADD) [offset = ACh]

31	16
IMPFTADD	
R-0	

15	0
IMPFTADD	
R-0	

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

Table 2-55. Imprecise Fault Write Address Register (IMPFTADD) Field Descriptions

Bit	Field	Value	Description
31-0	IMPFTADD	0- FFFF FFFFh	These bits contain the fault address when an imprecise abort occurs. **Note: These bits are only updated when an imprecise abort occurs.** **Note: These bits are cleared to 0 only on power-on reset. The value of this register remains unchanged after all other resets.**

2.5.1.38 System Software Interrupt Request 1 Register (SSIR1)

The SSIR1 register, shown in Figure 2-42 and described in Table 2-56, is used for software interrupt generation.

Figure 2-42. System Software Interrupt Request 1 Register (SSIR1) [offset = B0h]

31	16
Reserved	
R-0	

15	8	7	0
SSKEY1		SSDATA1	
R/W-0		R/W-0	

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

Table 2-56. System Software Interrupt Request 1 Register (SSIR1) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-8	SSKEY1	0-FFh	System software interrupt request key. A 075h written to these bits initiates IRQ/FIQ interrupts. Data in this field is always read as 0. The SSKEY1 field can be written into only if the write data matches the key (75h). The SSDATA1 field can only be written into if the write data into this field, the SSKEY1 field, matches the key (75h).
7-0	SSDATA1	0-FFh	System software interrupt data. These bits contain user read/write register bits. They may be used by the application software as different entry points for the interrupt routine. The SSDATA1 field cannot be written into unless the write data into the SSKEY1 field matches the key (75h); therefore, byte writes cannot be performed on the SSDATA1 field.

NOTE: This register is mirrored at offset FCh for compatibility reasons.

2.5.1.39 System Software Interrupt Request 2 Register (SSIR2)

The SSIR2 register, shown in Figure 2-43 and described in Table 2-57, is used for software interrupt generation.

Figure 2-43. System Software Interrupt Request 2 Register (SSIR2) [offset = B4h]

31	16
Reserved	
R-0	

15	8	7	0
SSKEY2		SSDATA2	
R/W-0		R/W-0	

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

Table 2-57. System Software Interrupt Request 2 Register (SSIR2) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-8	SSKEY2	0-FFh	System software interrupt2 request key. A 84h written to these bits initiates IRQ/FIQ interrupts. Data in this field is always read as 0. The SSKEY2 field can be written into only if the write data matches the key (84h). The SSDATA2 field can only be written into if the write data into this field, the SSKEY2 field, matches the key (84h).
7-0	SSDATA2	0-FFh	System software interrupt data. These bits contain user read/write register bits. They may be used by the application software as different entry points for the interrupt routine. The SSDATA2 field cannot be written into unless the write data into the SSKEY2 field matches the key (84h); therefore, byte writes cannot be performed on the SSDATA2 field.

2.5.1.40 System Software Interrupt Request 3 Register (SSIR3)

The SSIR3 register, shown in Figure 2-44 and described in Table 2-58, is used for software interrupt generation.

Figure 2-44. System Software Interrupt Request 3 Register (SSIR3) [offset = B8h]

31	16
Reserved	
R-0	

15	8	7	0
SSKEY3		SSDATA3	
R/W-0		R/W-0	

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

Table 2-58. System Software Interrupt Request 3 Register (SSIR3) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-8	SSKEY3	0-FFh	System software interrupt request key. A 93h written to these bits initiates IRQ/FIQ interrupts. Data in this field is always read as 0. The SSKEY3 field can be written into only if the write data matches the key (93h). The SSDATA3 field can only be written into if the write data into this field, the SSKEY3 field, matches the key (93h).
7-0	SSDATA3	0-FFh	System software interrupt data. These bits contain user read/write register bits. They may be used by the application software as different entry points for the interrupt routine. The SSDATA3 field cannot be written into unless the write data into the SSKEY3 field matches the key (93h); therefore, byte writes cannot be performed on the SSDATA3 field.

2.5.1.41 System Software Interrupt Request 4 Register (SSIR4)

The SSIR4 register, shown in Figure 2-45 and described in Table 2-59, is used for software interrupt generation.

Figure 2-45. System Software Interrupt Request 4 Register (SSIR4) [offset = BCh]

31	16
Reserved	
R-0	

15	8	7	0
SSKEY4		SSDATA4	
R/W-0		R/W-0	

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

Table 2-59. System Software Interrupt Request 4 Register (SSIR4) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-8	SSKEY4	0-FFh	System software interrupt2 request key. A A2h written to these bits initiates IRQ/FIQ interrupts. Data in this field is always read as 0. The SSKEY4 field can be written into only if the write data matches the key (A2h). The SSDATA4 field can only be written into if the write data into this field, the SSKEY4 field, matches the key (A2h).
7-0	SSDATA4	0-FFh	System software interrupt data. These bits contain user read/write register bits. They may be used by the application software as different entry points for the interrupt routine. The SSDATA4 field cannot be written into unless the write data into the SSKEY4 field matches the key (A2h); therefore, byte writes cannot be performed on the SSDATA4 field.

2.5.1.42 RAM Control Register (RAMGCR)

The RAMGCR register, shown in Figure 2-46 and described in Table 2-60, is used to configure eSRAM data and address wait states.

NOTE: The RAM_DFT_EN bits are for TI internal use only.

The contents of the RAM_DFT_EN field should not be changed.

Figure 2-46. RAM Control Register (RAMGCR) [offset = C0h]

31-20	19-16
Reserved	RAM_DFT_EN
R-0	R/WP-5h

15-8
Reserved
R-0

7-3	2	1	0
Reserved	WST_AENA0	Reserved	WST_DENA0
R-0	R/WP-0	R-0	R/WP-0

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

Table 2-60. RAM Control Register (RAMGCR) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	RAM_DFT_EN		Functional mode RAM DFT (Design For Test) port enable key.
			Note: For TI internal use only.
		Ah	RAM DFT port is enabled.
		Others	RAM DFT port is disabled.
			Note: It is recommended that a value of 5h be used to disable the RAM DFT port. This value will give maximum protection from a bit flip inducing event that would inadvertently enable the controller.
15-3	Reserved	0	Reads return 0. Writes have no effect.
2	WST_AENA0		eSRAM data phase wait state enable bit.
		0	The default address setup time for eSRAM0 is used.
		1	The eSRAM address setup time is increased by one HCLK cycle.
1	Reserved	0	Reads return 0. Writes have no effect.
0	WST_DENA0		eSRAM data phase wait state enable bit.
		0	There are no wait states for eSRAM during the data phase.
		1	The eSRAM data phase setup time is increased by one HCLK cycle.

2.5.1.43 Bus Matrix Module Control Register 1 (BMMCR1)

The BMMCR1 register, shown in Figure 2-47 and described in Table 2-61, allows RAM and Program (Flash) memory addresses to be swapped.

Figure 2-47. Bus Matrix Module Control Register 1 (BMMCR) [offset = C4h]

31	16
Reserved	
R-0	

15	4	3	0
Reserved		MEMSW	
R-0		R/WP-Ah	

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

Table 2-61. Bus Matrix Module Control Register 1 (BMMCR) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	MEMSW		Memory swap key. **Note: A CPU reset must be issued after the memory swap key has been changed for the memory swap to occur. A CPU reset can be initiated by changing the state of the CPU RESET bit in the CPURSTCR register.**
		5h	Swapped memory-map: eSRAM starts at address 0. Program memory (Flash) starts at address 800 0000h.
		Ah	Default memory-map: Program memory (Flash) starts at address 0. eSRAM starts at address 800 0000h.
		Others	The device memory-map is unchanged.

2.5.1.44 CPU Reset Control Register (CPURSTCR)

The CPURSTCR register shown in Figure 2-48 and described in Table 2-62 allows a reset to the Cortex-R4F CPU to be generated.

NOTE: The register bits in CPURSTCR are designated as high-integrity bits and have been implemented with error-correcting logic such that each bit, although read and written as a single bit, is actually a multi-bit key with error correction capability. As such, single-bit flips within the “key” can be corrected allowing protection of the system as a whole. An error detected is signaled to the ESM module.

Figure 2-48. CPU Reset Control Register (CPURSTCR) [offset = CCh]

31	16
Reserved	
R-0	

15	1	0
Reserved		CPU RESET
R-0		R/WP-0

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

Table 2-62. CPU Reset Control Register (CPURSTGCR) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	CPU RESET		CPU Reset. Only the CPU is reset whenever this bit is toggled. There is no system reset.

2.5.1.45 Clock Control Register (CLKCNTL)

The CLKCNTL register, shown in Figure 2-49 and described in Table 2-63, controls peripheral reset and the peripheral clock divide ratios.

NOTE: VCLK and VCLK2 clock ratio restrictions.

The VCLK2 frequency must always be greater than or equal to the VCLK frequency. The VCLK2 frequency must be an integer multiple of the VCLK frequency.

In addition, the VCLK and VCLK2 clock ratios must not be changed simultaneously. When increasing the frequency (decreasing the divider), first change the VCLK2R field and then change the VCLKR field. When reducing the frequency (increasing the divider), first change the VCLKR field and then change the VCLK2R field.

You should do a read-back between the two writes. This assures that there are enough clock cycles between the two writes.

Figure 2-49. Clock Control Register (CLKCNTRL) [offset = D0h]

31 28	27 24	23 20	19 16
Reserved	VCLK2R	Reserved	VCLKR
R-0	R/WP-1h	R-0	R/WP-1h

15 9	8	7 0
Reserved	PENA	Reserved
R-0	R/WP-0	R-0

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

Table 2-63. Clock Control Register (CLKCNTRL) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reads return 0. Writes have no effect.
27-24	VCLK2R		VBUS clock2 ratio.
			Note: The VCLK2 frequency must always be greater than or equal to the VCLK frequency. The VCLK2 frequency must be an integer multiple of the VCLK frequency. In addition, the VCLK and VCLK2 clock ratios must not be changed simultaneously.
		0	The VCLK2 speed is HCLK divided by 1.
		:	:
		Fh	The VCLK2 speed is HCLK divided by 16.
23-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	VCLKR		VBUS clock ratio.
			Note: The VCLK2 frequency must always be greater than or equal to the VCLK frequency. The VCLK2 frequency must be an integer multiple of the VCLK frequency. In addition, the VCLK and VCLK2 clock ratios must not be changed simultaneously.
		0	The VCLK speed is HCLK divided by 1.
		:	:
		Fh	The VCLK speed is HCLK divided by 16.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	PENA		Peripheral enable bit. The application must set this bit before accessing any peripheral
		0	The global peripheral/peripheral memory frames are in reset.
		1	All peripheral/peripheral memory frames are out of reset.
7-0	Reserved	0	Reads return 0. Writes have no effect.

2.5.1.46 ECP Control Register (ECPCNTL)

The ECP register, shown in Figure 2-50 and described in Table 2-64, configures the ECLK pin in functional mode.

NOTE: ECLK Functional mode configuration.

The ECLK pin must be placed into Functional mode by setting the ECPCLKFUN bit to 1 in the SYSPC1 register before a clock source will be visible on the ECLK pin.

Figure 2-50. ECP Control Register (ECPCNTL) [offset = D4h]

31 25	24	23	22 18	17 16
Reserved	ECPSSEL	ECPCOS	Reserved	ECPINSEL
R-0	R/W-0	R/W-0	R-0	R/W-0

15 0
ECPDIV
R/W-0

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

Table 2-64. ECP Control Register (ECPCNTL) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	ECPSSEL		This bit allows the selection between VCLK and OSCIN as the clock source for ECLK. **Note: Other ECLK clock sources are available for debug purposes by configuring the CLKTEST register.**
		0	VCLK is selected as the ECP clock source.
		1	OSCIN is selected as the ECP clock source.
23	ECPCOS		ECP continue on suspend. **Note: Suspend mode is entered while performing certain JTAG debugging operations.**
		0	ECLK output is disabled in suspend mode. ECLK output will be shut off and will not be seen on the I/O pin of the device.
		1	ECLK output is not disabled in suspend mode. ECLK output will not be shut off and will be seen on the I/O pin of the device.
22-18	Reserved	0	Reads return 0. Writes have no effect.
17-16	ECPINSEL		Select ECP input clock source.
		0	Tied Low
		1h	HCLK
		2h	External clock
		3h	Tied Low
15-0	ECPDIV	0-FFFFh	ECP divider value. The value of ECPDIV bits determine the external clock (ECP clock) frequency as a ratio of VBUS clock or OSCIN as shown in the formula: $ECLK = \frac{VCLKorOSCIN}{(ECPDIV + 1)}$

2.5.1.47 DEV Parity Control Register 1 (DEVCR1)

This register is shown in Figure 2-51 and described in Table 2-65.

Figure 2-51. DEV Parity Control Register 1 (DEVCR1) [offset = DCh]

31–16
Reserved
R-0

15–4	3–0
Reserved	DEVPARSEL
R-0	R/WP-Ah

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

Table 2-65. DEV Parity Control Register 1 (DEVCR1) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	DEVPARSEL		Device parity select bit key. **Note: After an odd (DEVPARSEL = 5h) or even (DEVPARSEL = Ah) scheme is programmed into the DEVPARSEL register, any one bit change can be detected and will retain its programmed scheme. More than one bit changes in DEVPARSEL will cause a default to odd parity scheme.**
		5h	The device parity is even.
		Ah	The device parity is odd.

2.5.1.48 System Exception Control Register (SYSECR)

The SYSECR register, shown in Figure 2-52 and described in Table 2-66, is used to generate a software reset.

NOTE: The register bits in SYSECR are designated as high-integrity bits and have been implemented with error-correcting logic such that each bit, although read and written as a single bit, is actually a multi-bit key with error correction capability. As such, single-bit flips within the “key” can be corrected allowing protection of the system as a whole. An error detected is signaled to the ESM module.

Figure 2-52. System Exception Control Register (SYSECR) [offset = E0h]

31–16
Reserved
R-0

15	14	13–0
RESET1	RESET0	Reserved
R/WP-0	R/WP-1	R-0

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

Table 2-66. System Exception Control Register (SYSECR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-14	RESET[1-0]		Software reset bits. Setting RESET1 or clearing RESET0 causes a system software reset.
		1h	No reset will occur.
		1x, x0	A global system reset will occur.
13-0	Reserved	0	Reads return 0. Writes have no effect.

2.5.1.49 System Exception Status Register (SYSESR)

The SYSESR register, shown in Figure 2-53 and described in Table 2-67, shows the source for different resets encountered. Previous reset source status bits are not automatically cleared if new resets occur. After reading this register, the software should clear any flags that are set so that the source of future resets can be determined. Any bit in this register can be cleared by writing a 1 to the bit.

Figure 2-53. System Exception Status Register (SYSESR) [offset = E4h]

31	16
Reserved	
R-0	

15	14	13	12	8
PORST	OSCRST	WDRST	Reserved	
R/WC-X	R/WC-X*	R/WC-X*	R-0	

7	6	5	4	3	2	0
Reserved		CPURST	SWRST	EXTRST	Reserved	
R-0		R/WC-X*	R/WC-X*	R/WC-X*	R-0	

LEGEND: R/W = Read/Write; R = Read only; C= Clear; X = value unchanged after reset; X* = 0 after PORST but unchanged after other resets; -n = value after reset

Table 2-67. System Exception Status Register (SYSESR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15	PORST		Power-on reset. This bit is set when a power-on reset occurs, either internally asserted by the VMON or externally asserted by the nPORRST pin.
		0	No power-on reset has occurred since this bit was last cleared.
		1	A reset was caused by a power-on reset. (This bit should be cleared after being read so that subsequent resets can be properly identified as not being power-on resets.)
14	OSCRST		Reset caused by an oscillator failure or PLL cycle slip. This bit is set when a reset is caused by an oscillator failure or PLL slip. **Note: The action taken when an oscillator failure or PLL slip is detected must configured in the PLLCTL1 register.**
		0	No reset has occurred due to an oscillator failure or a PLL cycle slip.
		1	A reset was caused by an oscillator failure or a PLL cycle slip.
13	WDRST		Watchdog reset flag. This bit is set when the last reset was caused by the digital windowed watchdog. During debugging, the ICEPICK logic implemented on the microcontroller also allows a system reset to be generated via the debug logic (DBGRST). This DBGRST reset is also indicated on the WDRST bit of the SYSESR. This flag can also be set via a reset driven by ICEPICK.
		0	No reset has occurred because of the DWWD.
		1	A reset was caused by the DWWD.
12-6	Reserved	0	Reads return 0. Writes have no effect.
5	CPURST		CPU reset flag. This bit is set when the CPU is reset. **Note: A CPU reset can be initiated by the CPU self-test controller (LBIST) or by changing the memory protection (MMU/MPU) configuration in CPURSTCR register.**
		0	No CPU reset has occurred.
		1	A CPU reset occurred.
4	SWRST		Software reset flag. This bit is set when a software system reset has occurred. **Note: A software system reset can be initiated by writing to the RESET bits in the SYSECR register.**
		0	No software reset has occurred.
		1	A software reset occurred.

Table 2-67. System Exception Status Register (SYSESR) Field Descriptions (continued)

Bit	Field	Value	Description
3	EXTRST		External reset flag. This bit is set when a reset is caused by the external reset pin nRST or by any reset that also asserts the nRST pin (PORST, OSCRST, WDRST and SWRST).
		0	The external reset pin has not asserted a reset.
		1	A reset has been caused by the external reset pin.
2-0	Reserved	0	Reads return 0. Writes have no effect.

2.5.1.50 System Test Abort Status Register (SYSTASR)

This register is shown in Figure 2-54 and described in Table 2-68.

Figure 2-54. System Test Abort Status Register (SYSTASR) [offset = E8h]

31 … 16
Reserved
R-0

15 … 5	4 … 0
Reserved	EFUSE_Abort
R-0	R/WPC-X/0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; C = Clear; -X = Value unchanged after reset; -*n* = value after reset

Table 2-68. System Test Abort Status Register (SYSTASR) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reads return 0. Writes have no effect.
4-0	EFUSE_Abort		Test Abort status flag. These bits are set when test abort occurred:
		0	*Read:* The last operation (if any) completed successfully. This is also the value that the error/status register is set to after reset.
		1h	*Read:* Controller times out because there is no last row sent from the FuseROM.
		2h	*Read:* The autoload machine was started, either through the SYS_INITZ signal from the system or the JTAG data register. In either case, the autoload machine did not find enough FuseROM data to fill the scan chain.
		3h	*Read:* The autoload machine was started, either through the SYS_INITZ signal from the system or the JTAG data register. In either case, the autoload machine starts the scan chain with a signature it expects to see after the scan chain is full. The autoload machine was able to fill the scan chain, but the wrong signature was returned.
		4h	*Read:* The autoload machine was started, either through the SYS_INITZ signal from the system or the JTAG data register. In either case, the autoload machine was not able or not allowed to complete its operation.
		Others	*Read:* Reserved.
		1Fh	*Write:* These bits are cleared to 0.

2.5.1.51 Global Status Register (GLBSTAT)

The GLBSTAT register, shown in Figure 2-55 and described in Table 2-69, indicates the FMzPLL (PLL1) slip status and the oscillator fail status.

NOTE: PLL and OSC fail behavior

The device behavior after a PLL slip or an oscillator failure is configured in the PLLCTL1 register.

Figure 2-55. Global Status Register (GLBSTAT) [offset = ECh]

31-16
Reserved
R-0

15-10	9	8	7-1	0
Reserved	FBSLIP	RFSLIP	Reserved	OSCFAIL
R-0	R/W1C-n	R/W1C-n	R-0	R/W1C-n

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to Clear; -*n* = value after reset

Table 2-69. Global Status Register (GLBSTAT) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return 0. Writes have no effect.
9	FBSLIP		PLL over cycle slip detection. (cleared by nPORRST, maintains its previous value for all other resets)
		0	*Read:* No PLL over cycle slip has been detected. *Write:* The bit is unchanged.
		1	*Read:* A PLL over cycle slip has been detected. *Write:* The bit is cleared to 0.
8	RFSLIP		PLL under cycle slip detection. (cleared by nPORRST, maintains its previous value for all other resets)
		0	*Read:* No PLL under cycle slip has been detected. *Write:* The bit is unchanged.
		1	*Read:* A PLL under cycle slip has been detected. *Write:* The bit is cleared to 0.
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	OSCFAIL		Oscillator fail flag bit. (cleared by nPORRST, maintains its previous value for all other resets)
		0	*Read:* No oscillator failure has been detected. *Write:* The bit is unchanged.
		1	*Read:* An oscillator failure has been detected. *Write:* The bit is cleared to 0.

2.5.1.52 Device Identification Register (DEVID)

The DEVID is a read-only register. It contains device-specific information that is hard-coded during device manufacture. For the initial silicon version, the device identification code value is 802A AD05h. This register is shown in Figure 2-56 and described in Table 2-70.

Figure 2-56. Device Identification Register (DEVID) [offset = F0h]

31	30	17	16
CP15	UNIQUE ID		TECH
R-K	R-K		R-K

15	13	12	11	10	9	8
TECH		I/O VOLTAGE	PERIPHERAL PARITY	FLASH ECC		RAM ECC
R-K		R-K	R-K	R-K		R-K

7	3	2	1	0
VERSION		PLATFORM ID		
R-K		R-K		

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

Table 2-70. Device Identification Register (DEVID) Field Descriptions

Bit	Field	Value	Description
31	CP15		CP15 CPU. This bit indicates whether the CPU has a coprocessor 15 (CP15).
		0	The CPU has no CP15 present.
		1	The CPU has a CP15 present. The CPU ID can be read using the CP15 C0,C0,0 register.
30-17	UNIQUE ID	0-3FFFh	Device ID. The device ID is unique by device configuration.
16-13	TECH		These bits define the process technology by which the device was manufactured.
		0	Device manufactured in the C05 process technology.
		1h	Device manufactured in the F05 process technology.
		2h	Device manufactured in the C035 process technology.
		3h	Device manufactured in the F035 process technology.
		4h	Device manufactured in the C021 process technology.
		5h	Device manufactured in the F021 process technology.
		6h-Fh	Reserved
12	I/O VOLTAGE		Input/output voltage. This bit defines the I/O voltage of the device.
		0	The I/O voltage is 3.3 V.
		1	The I/O voltage is 5 V.
11	PERIPHERAL PARITY		Peripheral parity. This bit indicates whether or not peripheral memory parity is present.
		0	The peripheral memories have no parity.
		1	The peripheral memories have parity.
10-9	FLASH ECC		These bits indicate which parity is present for the program memory.
		0	No memory protection is present.
		1h	The program memory (Flash) has single-bit parity.
		2h	The program memory (Flash) has ECC.
		3h	This combination is reserved.
8	RAM ECC		RAM ECC. This bit indicates whether or not RAM memory ECC is present.
		0	The RAM memories do not have ECC.
		1	The RAM memories have ECC.
7-3	VERSION	0-1Fh	Version. These bits provide the revision of the device.
2-0	PLATFORM ID	5h	The device is part of the TMS570 family. The TMS570 ID is always 5h.

2.5.1.53 Software Interrupt Vector Register (SSIVEC)

The SSIVEC register, shown in Figure 2-57 and described in Table 2-71, contains information about software interrupts.

Figure 2-57. Software Interrupt Vector Register (SSIVEC) [offset = F4h]

31	16
Reserved	
R-0	

15	8	7	0
SSIDATA		SSIVECT	
R-0		R-0	

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

Table 2-71. Software Interrupt Vector Register (SSIVEC) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-8	SSIDATA	0-FFh	System software interrupt data key. These bits contain the data key value of the source for the system software interrupt, which is indicated by the vector in the SSIVEC[7-0] field.
7-0	SSIVECT		These bits contain the source for the system software interrupt. **Note: A read from the SSIVECT bits clears the corresponding SSI_FLAG[4-1] bit in the SSIF register, corresponding to the source vector of the system software interrupt.** **Note: The SSIR[4-1] interrupt has the following priority order:** SSIR1 has the highest priority. SSIR4 has the lowest priority.
		0	No software interrupt is pending.
		1h	A software interrupt has been generated by writing the correct key value to The SSIR1 register.
		2h	A software interrupt has been generated by writing the correct key value to The SSIR2 register.
		3h	A software interrupt has been generated by writing the correct key value to The SSIR3 register.
		4h	A software interrupt has been generated by writing the correct key value to The SSIR4 register.
		5h-FFh	Reserved

2.5.1.54 System Software Interrupt Flag Register (SSIF)

The SSIF register, shown in Figure 2-58 and described in Table 2-72, contains software interrupt flag status information.

Figure 2-58. System Software Interrupt Flag Register (SSIF) [offset = F8h]

31 16
Reserved
R-0

15 8
Reserved
R-0

7 4	3	2	1	0
Reserved	SSI_FLAG4	SSI_FLAG3	SSI_FLAG2	SSI_FLAG1
R-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0

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

Table 2-72. System Software Interrupt Flag Register (SSIF) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	SSI_FLAG[4-1]		System software interrupt flag[4-1]. This flag is set when the correct SSKEY is written to the SSIR register[4-1]. **Note: A read from the SSIVEC register clears the corresponding SSI_FLAG [4-1] bit in the SSIF, corresponding to the source vector of the system software interrupt.**
		0	*Read:* No IRQ/FIQ interrupt was generated since the bit was last cleared. *Write:* The bit is unchanged.
		1	*Read:* An IRQ/FIQ interrupt was generated. *Write:* The bit is cleared to 0.

2.5.2 Secondary System Control Registers (SYS2)

Table 2-73 lists the secondary frame of system registers. The start address of the second system module frame is FFFF E100h. The registers support 32-, 16-, and 8-bit writes. The offset is relative to the system module frame start address.

Table 2-73. Secondary System Control Registers

Offset	Acronym	Register Description	Section
00h	PLLCTL3	PLL Control Register 3	Section 2.5.2.1
04h	Reserved	Reserved	
08h	STCCLKDIV	CPU Logic BIST Clock Divider	Section 2.5.2.2
0Ch-20h	Reserved	Reserved	
24h	ECPCNTL	ECP Control Register	Section 2.5.1.46
28h-38h	Reserved	Reserved	
3Ch	CLK2CNTRL	Clock 2 Control Register	Section 2.5.2.3
40h	VCLKACON1	Peripheral Asynchronous Clock Configuration 1 Register	Section 2.5.2.4
44h-6Ch	Reserved	Reserved	
70h	CLKSLIP	Clock Slip Register	Section 2.5.2.5
74h-E8h	Reserved	Reserved	
ECh	EFC_CTLREG	EFUSE Controller Control Register	Section 2.5.2.6
F0h	DIEIDL_REG0	Die Identification Register Lower Word	Section 2.5.2.7
F4h	DIEIDH_REG1	Die Identification Register Upper Word	Section 2.5.2.8
F8h	DIEIDL_REG2	Die Identification Register Lower Word	Section 2.5.2.9
FCh	DIEIDH_REG3	Die Identification Register Upper Word	Section 2.5.2.10

NOTE: All additional registers in the secondary system frame are reserved.

2.5.2.1 PLL Control Register 3 (PLLCTL3)

The PLLCTL3 register is shown in Figure 2-59 and described in Table 2-74; controls the settings of PLL2 (Clock Source 6 - FPLL).

Figure 2-59. PLL Control Register 3 (PLLCTL3) [offset = 00]

31 29	28 24	23 22	21 16
ODPLL2	PLLDIV2	Reserved	REFCLKDIV2
R/WP-1h	R/WP-Fh	R-0	R/WP-2h

15 0
PLL_MUL2
R/WP-5F00h

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

Table 2-74. PLL Control Register 3 (PLLCTL3) Field Descriptions

Bit	Field	Value	Description
31-29	ODPLL2		Internal PLL Output Divider
			OD2 = ODPLL2 + 1, ranges from 1 to 8. $f_{post\_ODCLK2} = f_{output\_CLK2}$ / OD2
		0	$f_{post\_ODCLK2} = f_{output\_CLK2}$ / 1
		1h	$f_{post\_ODCLK2} = f_{output\_CLK2}$ / 2
		:	:
		7h	$f_{post\_ODCLK2} = f_{output\_CLK2}$ / 8
			Note: PLL output clock is gated off if ODPLL2 is changed while the PLL2 is active.
28-24	PLLDIV2		PLL#2 Output Clock Divider
			R2 = PLLDIV2 + 1, ranges from 1 to 32. $f_{PLL2\ CLK} = f_{post\_ODCLK2}$ / R2
		0	$f_{PLL2\ CLK} = f_{post\_ODCLK2}$ /1
		1h	$f_{PLL2\ CLK} = f_{post\_ODCLK2}$ /2
		:	:
		1Fh	$f_{PLL2\ CLK} = f_{post\_ODCLK2}$ /32
23-22	Reserved	0	Value has no effect on PLL operation.
21-16	REFCLKDIV2		Reference Clock Divider
			NR2 = REFCLKDIV2 + 1, ranges from 1 to 64. $f_{INTCLK2} = f_{OSCIN}$ / NR2
		0	$f_{INTCLK2} = f_{OSCIN}$ /1
		1h	$f_{INTCLK2} = f_{OSCIN}$ /2
		:	:
		3Fh	$f_{INTCLK2} = f_{OSCIN}$ /64
			Note: This value should not be changed while the PLL2 is active.
15-0	PLLMUL2		PLL2 Multiplication Factor
			NF2 = (PLLMUL2 / 256) + 1, valid multiplication factors are from 1 to 256. $f_{VCOCLK2} = f_{INTCLK2}$ x NF2
			User and privileged mode (read): Privileged mode (write):
		100h	$f_{VCOCLK2} = f_{INTCLK2}$ x 1
		:	:
		5B00h	$f_{VCOCLK2} = f_{INTCLK2}$ x 92
		5C00h	$f_{VCOCLK2} = f_{INTCLK2}$ x 93
		:	:
		FF00h	$f_{VCOCLK2} = f_{INTCLK2}$ x 256

2.5.2.2 CPU Logic Bist Clock Divider (STCLKDIV)

This register is shown in Figure 2-60 and described in Table 2-75.

Figure 2-60. CPU Logic BIST Clock Prescaler (STCLKDIV) [offset = 08h]

31 27	26 24	23 16
Reserved	CLKDIV	Reserved
R-0	R/WP-0	R-0

15 0
Reserved
R-0

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

Table 2-75. CPU Logic BIST Clock Prescaler (STCLKDIV) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reads return 0. Writes have no effect.
26-24	CLKDIV	0	Clock divider/prescaler for CPU clock during logic BIST
23-0	Reserved	0	Reads return 0. Writes have no effect.

2.5.2.3 Clock 2 Control Register (CLK2CNTRL)

This register is shown in Figure 2-61 and described in Table 2-76.

Figure 2-61. Clock 2 Control Register (CLK2CNTRL) [offset = 3Ch]

31 16
Reserved
R-0

15 12	11 8	7 4	3 0
Reserved	VCLK4R	Reserved	VCLK3R
R-0	R/WP-1h	R-0	R/WP-1h

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

Table 2-76. Clock 2 Control Register (CLK2CNTRL) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reads return 0. Writes have no effect.
11-8	VCLK4R		VBUS clock4 ratio.
		0	The ratio is HCLK divide by 1.
		:	:
		Fh	The ratio is HCLK divided by 16.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	VCLK3R		VBUS clock3 ratio.
		0	The ratio is HCLK divide by 1.
		:	:
		Fh	The ratio is HCLK divided by 16.

2.5.2.4 Peripheral Asynchronous Clock Configuration 1 Register (VCLKACON1)

This register is shown in Figure 2-62 and described in Table 2-77.

Figure 2-62. Peripheral Asynchronous Clock Configuration 1 Register (VCLKACON1) [offset = 40h]

31 27	26 24
Reserved	VCLKA4R
R-0	R/WP-1h

23 21	20	19 16
Reserved	VCLKA4_DIV_ CDDIS	VCLKA4S
R-0	R/WP-0	R/WP-9h

15 11	10 8
Reserved	VCLKA3R
R-0	R/WP-1h

7 5	4	3 0
Reserved	VCLKA3_DIV_ CDDIS	VCLKA3S
R-0	R/WP-0	R/WP-9h

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

Table 2-77. Peripheral Asynchronous Clock Configuration 1 Register (VCLKACON1) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reads return 0. Writes have no effect.
26-24	VCLKA4R		Clock divider for the VCLKA4 source. Output will be present on VCLKA4_DIVR.
			VCLKA4 domain will be enabled by writing to the CDDIS register and VCLKA4_DIV_CDDIS bit It can inferred that VCLKA4_DIV clock is disabled when VCLKA4 clock is disabled.
		0	The ratio is VCLKA4 divided by 1.
		:	:
		7h	The ratio is VCLKA4 divided by 8.
23-21	Reserved	0	Reads return 0. Writes have no effect.
20	VCLKA4_DIV_CDDIS		Disable the VCLKA4 divider output.
			VCLKA4 domain will be enabled by writing to the CDDIS register
		0	Enable the prescaled VCLKA4 clock on VCLKA4_DIVR.
		1	Disable the prescaled VCLKA4 clock on VCLKA4_DIVR.
19-16	VCLKA4S		Peripheral asynchronous clock4 source.
		0	Clock source0 is the source for peripheral asynchronous clock4.
		1h	Clock source1 is the source for peripheral asynchronous clock4.
		2h	Clock source2 is the source for peripheral asynchronous clock4.
		3h	Clock source3 is the source for peripheral asynchronous clock4.
		4h	Clock source4 is the source for peripheral asynchronous clock4.
		5h	Clock source5 is the source for peripheral asynchronous clock4.
		6h	Clock source6 is the source for peripheral asynchronous clock4.
		7h	Clock source7 is the source for peripheral asynchronous clock4.
		8h-Fh	VCLK or a divided VCLK is the source for peripheral asynchronous clock4. See the device-specific data manual for details.
15-11	Reserved	0	Reads return 0. Writes have no effect.

Table 2-77. Peripheral Asynchronous Clock Configuration 1 Register (VCLKACON1) Field Descriptions (continued)

Bit	Field	Value	Description
10-8	VCLKA3R		Clock divider for the VCLKA3 source. Output will be present on VCLKA3_DIVR.
			VCLKA3 domain will be enabled by writing to the CDDIS register and VCLKA3_DIV_CDDIS bit.
			It can inferred that VCLKA3_DIV clock is disabled when VCLKA3 clock is disabled.
		0	The ratio is VCLKA3 divided by 1.
		:	:
		7h	The ratio is VCLKA3 divided by 8.
7-5	Reserved	0	Reads return 0. Writes have no effect.
4	VCLKA3_DIV_CDDIS		Disable the VCLKA3 divider output.
			VCLKA3 domain will be enabled by writing to the CDDIS register
		0	Enable the prescaled VCLKA3 clock on VCLKA3_DIVR.
		1	Disable the prescaled VCLKA3 clock on VCLKA3_DIVR.
3-0	VCLKA3S		Peripheral asynchronous clock3 source.
		0	Clock source0 is the source for peripheral asynchronous clock3.
		1h	Clock source1 is the source for peripheral asynchronous clock3.
		2h	Clock source2 is the source for peripheral asynchronous clock3.
		3h	Clock source3 is the source for peripheral asynchronous clock3.
		4h	Clock source4 is the source for peripheral asynchronous clock3.
		5h	Clock source5 is the source for peripheral asynchronous clock3.
		6h	Clock source6 is the source for peripheral asynchronous clock3.
		7h	Clock source7 is the source for peripheral asynchronous clock3.
		8h-Fh	VCLK is the source for peripheral asynchronous clock3.

NOTE: Nonimplemented clock sources should not be enabled or used. A list of the available clock sources is shown in Table 2-9.

2.5.2.5 Clock Slip Register (CLKSLIP)

This register is shown in Figure 2-63 and described in Table 2-78.

Figure 2-63. Clock Slip Register (CLKSLIP) [offset = 70h]

31 … 16
Reserved
R-0

15 14	13 … 8	7 … 4	3 … 0
Reserved	PLL1_SLIP_FILTER_COUNT	Reserved	PLL1_SLIP_FILTER_KEY
R-0	R/WP-0	R-0	R/WP-5h

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

Table 2-78. Clock Slip Register (CLKSLIP) Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Reads return 0. Writes have no effect.
13-8	PLL1_SLIP_FILTER_COUNT		PLL SLIP down counter programmed value. Count is on 10M clock.
			On reset, counter value is 0. Counter must be programmed to a non-zero value and enabled for the filtering to be enabled.
		0	Filtering is disabled.
		1h	Filtering is enabled. Every slip is recognized.
		2h	Filtering is enabled. The slip must be at least 2 HF LPO cycles wide in order to be recognized as a slip.
		:	:
		3Fh	Filtering is enabled. The SLIP must be at least 63 HF LPO cycles wide in order to be recognized as a slip.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	PLL1_SLIP_FILTER_KEY		Enable the PLL SLIP filtering.
		5h	On reset, the PLL SLIP filter is disabled and the PLL SLIP passes through.
		Fh	This is an unsupported value. You should a void writing this value to this bit field.
		Others	PLL SLIP filtering is enabled. Recommended to program Ah in this bit field. Enabling of the PLL SLIP occurs when the KEY is programmed and a non-zero value is present in the COUNT field.

2.5.2.6 EFUSE Controller Control Register (EFC_CTLREG)

This register is shown in Figure 2-64 and described in Table 2-79.

Figure 2-64. EFUSE Controller Control Register (EFC_CTLREG) [offset = ECh]

31	16
Reserved	
R-0	

15	4	3	0
Reserved		EFC_INSTR_WEN	
R-0		R/WP-5h	

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

Table 2-79. EFUSE Controller Control Register (EFC_CTLREG) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	EFC_INSTR_WEN		Enable user write of 4 EFUSE controller instructions. SYS module generates the enable signal that will be tied to OCP_FROM_WRITE_DISABLE on efuse controller port
		Ah	Writing of instructions (Program, ProgramCRA, RunAutoload, and LoadFuseScanchain) to EFC registers is enabled.
		Others	Writing of instructions (Program, ProgramCRA, RunAutoload, and LoadFuseScanchain) in EFC registers is blocked.

2.5.2.7 Die Identification Register Lower Word (DIEIDL_REG0)

The DIEIDL_REG0 register is a duplicate of the DIEIDL register, see Section 2.5.1.29. The DIEIDL_REG0 register, shown in Figure 2-65 and described in Table 2-80, contains information about the die wafer number, and X, Y wafer coordinates.

Figure 2-65. Die Identification Register, Lower Word (DIEIDL_REG0) [offset = F0h]

31	24	23	16
WAFER #		Y WAFER COORDINATE	
R-D		R-D	

15	12	11	0
Y WAFER COORDINATE		X WAFER COORDINATE	
R-D		R-D	

LEGEND: R = Read only; -*n* = value after reset; D = value is device specific

Table 2-80. Die Identification Register, Lower Word (DIEIDL_REG0) Field Descriptions

Bit	Field	Description
31-24	WAFER #	These read-only bits contain the wafer number of the device.
23-12	Y WAFER COORDINATE	These read-only bits contain the Y wafer coordinate of the device.
11-0	X WAFER COORDINATE	These read-only bits contain the X wafer coordinate of the device.

NOTE: Die Identification Information

The die identification information will vary from unit to unit. This information is programmed by TI as part of the initial device test procedure.

2.5.2.8 Die Identification Register Upper Word (DIEIDH_REG1)

The DIEIDH_REG1 register is a duplicate of the DIEIDH register, see Section 2.5.1.30. The DIEIDH_REG1 register, shown in Figure 2-66 and described in Table 2-81, contains information about the die lot number.

Figure 2-66. Die Identification Register, Upper Word (DIEIDH_REG1) [offset = F4h]

31 24	23 16
Reserved	LOT #
R-0	R-D

15 0
LOT #
R-D

LEGEND: R = Read only; -*n* = value after reset; D = value is device dependent

Table 2-81. Die Identification Register, Upper Word (DIEIDH_REG1) Field Descriptions

Bit	Field	Description
31-24	Reserved	Reserved for TI use. Writes have no effect.
23-0	LOT #	This read-only register contains the device lot number.

NOTE: Die Identification Information

The die identification information will vary from unit to unit. This information is programmed by TI as part of the initial device test procedure.

2.5.2.9 Die Identification Register Lower Word (DIEIDL_REG2)

This register is shown in Figure 2-67 and described in Table 2-82.

Figure 2-67. Die Identification Register, Lower Word (DIEIDL_REG2) [offset = F8h]

31 0
DIEIDL2
R-X

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

Table 2-82. Die Identification Register, Lower Word (DIEIDL_REG2) Field Descriptions

Bit	Field	Value	Description
31-0	DIEIDL2(95-64)	0-FFFF FFFFh	This read-only register contains the lower word (95:64) of the die ID information. See the device data sheet for the content of the register.

2.5.2.10 Die Identification Register Upper Word (DIEIDH_REG3)

This register is shown in Figure 2-68 and described in Table 2-83.

Figure 2-68. Die Identification Register, Upper Word (DIEIDH_REG3) [offset = FCh]

31 0
DIEIDH2
R-X

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

Table 2-83. Die Identification Register, Upper Word (DIEIDH_REG3) Field Descriptions

Bit	Field	Value	Description
31-0	DIEIDH2(127-96)	0-FFFF FFFFh	This read-only register contains the upper word (127:97) of the die ID information. See the device data sheet for the content of the register.

2.5.3 Peripheral Central Resource (PCR) Control Registers

This section describes the Peripheral Central Resource (PCR) control registers. The start address of the PCR register frame is FFFF E000h. Table 2-84 summarizes the registers in the PCR, which are used to configure protection to the peripherals in PCS and PS regions. Not all chip selects exist on this device.

Table 2-84. Peripheral Central Resource Control Registers

Offset	Acronym	Register Description	Section
00h	PMPROTSET0	Peripheral Memory Protection Set Register 0	Section 2.5.3.1
04h	PMPROTSET1	Peripheral Memory Protection Set Register 1	Section 2.5.3.2
08h-0Ch	Reserved	Reserved	
10h	PMPROTCLR0	Peripheral Memory Protection Clear Register 0	Section 2.5.3.3
14h	PMPROTCLR1	Peripheral Memory Protection Clear Register 1	Section 2.5.3.4
18h-1Ch	Reserved	Reserved	
20h	PPROTSET0	Peripheral Protection Set Register 0	Section 2.5.3.5
24h	PPROTSET1	Peripheral Protection Set Register 1	Section 2.5.3.6
28h	PPROTSET2	Peripheral Protection Set Register 2	Section 2.5.3.7
2Ch	PPROTSET3	Peripheral Protection Set Register 3	Section 2.5.3.8
30h-3Ch	Reserved	Reserved	
40h	PPROTCLR0	Peripheral Protection Clear Register 0	Section 2.5.3.9
44h	PPROTCLR1	Peripheral Protection Clear Register 1	Section 2.5.3.10
48h	PPROTCLR2	Peripheral Protection Clear Register 2	Section 2.5.3.11
4Ch	PPROTCLR3	Peripheral Protection Clear Register 3	Section 2.5.3.12
50h-5Ch	Reserved	Reserved	
60h	PCSPWRDWNSET0	Peripheral Memory Power-Down Set Register 0	Section 2.5.3.13
64h	PCSPWRDWNSET1	Peripheral Memory Power-Down Set Register 1	Section 2.5.3.14
68h-6Ch	Reserved	Reserved	
70h	PCSPWRDWNCLR0	Peripheral Memory Power-Down Clear Register 0	Section 2.5.3.15
74h	PCSPWRDWNCLR1	Peripheral Memory Power-Down Clear Register 1	Section 2.5.3.16
78h-7Ch	Reserved	Reserved	
80h	PSPWRDWNSET0	Peripheral Power-Down Set Register 0	Section 2.5.3.17
84h	PSPWRDWNSET1	Peripheral Power-Down Set Register 1	Section 2.5.3.18
88h	PSPWRDWNSET2	Peripheral Power-Down Set Register 2	Section 2.5.3.19
8Ch	PSPWRDWNSET3	Peripheral Power-Down Set Register 3	Section 2.5.3.20
90h-9Ch	Reserved	Reserved	
A0h	PSPWRDWNCLR0	Peripheral Power-Down Clear Register 0	Section 2.5.3.21
A4h	PSPWRDWNCLR1	Peripheral Power-Down Clear Register 1	Section 2.5.3.22
A8h	PSPWRDWNCLR2	Peripheral Power-Down Clear Register 2	Section 2.5.3.23
ACh	PSPWRDWNCLR3	Peripheral Power-Down Clear Register 3	Section 2.5.3.24

2.5.3.1 Peripheral Memory Protection Set Register 0 (PMPROTSET0)

This register is shown in Figure 2-69 and described in Table 2-85.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-69. Peripheral Memory Protection Set Register 0 (PMPROTSET0) [offset = 00]

31 … 0
PCS[31-0]PROTSET
R/WP-0

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

Table 2-85. Peripheral Memory Protection Set Register 0 (PMPROTSET0) Field Descriptions

Bit	Field	Value	Description
31-0	PCS[31-0]PROTSET		Peripheral memory frame protection set.
		0	*Read:* The peripheral memory frame*n* can be written to and read from in both user and privileged modes. *Write:* The bit is unchanged.
		1	*Read:* The peripheral memory frame*n* can be written to only in privileged mode, but it can be read in both user and privileged modes. *Write:* The corresponding bit in PMPROTSET0 and PMPROTCLR0 registers is set to 1.

2.5.3.2 Peripheral Memory Protection Set Register 1 (PMPROTSET1)

This register is shown in Figure 2-70 and described in Table 2-86.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-70. Peripheral Memory Protection Set Register 1 (PMPROTSET1) [offset = 04h]

31 … 0
PCS[63-32]PROTSET
R/WP-0

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

Table 2-86. Peripheral Memory Protection Set Register 1 (PMPROTSET1) Field Descriptions

Bit	Field	Value	Description
31-0	PCS[63-32]PROTSET		Peripheral memory frame protection set.
		0	*Read:* The peripheral memory frame*n* can be written to and read from in both user and privileged modes. *Write:* The bit is unchanged.
		1	*Read:* The peripheral memory frame*n* can be written to only in privileged mode, but it can be read in both user and privileged modes. *Write:* The corresponding bit in PMPROTSET1 and PMPROTCLR1 registers is set to 1.

2.5.3.3 Peripheral Memory Protection Clear Register 0 (PMPROTCLR0)

This register is shown in Figure 2-71 and described in Table 2-87.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-71. Peripheral Memory Protection Clear Register 0 (PMPROTCLR0) [offset = 10h]

31	0
PCS[31-0]PROTCLR	
R/WP-0	

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

Table 2-87. Peripheral Memory Protection Clear Register 0 (PMPROTCLR0) Field Descriptions

Bit	Field	Value	Description
31-0	PCS[31-0]PROTCLR		Peripheral memory frame protection clear.
		0	*Read:* The peripheral memory frame*n* can be written to and read from in both user and privileged modes. *Write:* The bit is unchanged.
		1	*Read:* The peripheral memory frame*n* can be written to only in privileged mode, but it can be read in both user and privileged modes. *Write:* The corresponding bit in PMPROTSET0 and PMPROTCLR0 registers is cleared to 0.

2.5.3.4 Peripheral Memory Protection Clear Register 1 (PMPROTCLR1)

This register is shown in Figure 2-72 and described in Table 2-88.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-72. Peripheral Memory Protection Clear Register 1 (PMPROTCLR1) [offset = 14h]

31	0
PCS[63-32]PROTCLR	
R/WP-0	

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

Table 2-88. Peripheral Memory Protection Clear Register 1 (PMPROTCLR1) Field Descriptions

Bit	Field	Value	Description
31-0	PCS[63-32]PROTCLR		Peripheral memory frame protection clear.
		0	*Read:* The peripheral memory frame*n* can be written to and read from in both user and privileged modes. *Write:* The bit is unchanged.
		1	*Read:* The peripheral memory frame*n* can be written to only in privileged mode, but it can be read in both user and privileged modes. *Write:* The corresponding bit in PMPROTSET1 and PMPROTCLR1 registers is cleared to 0.

2.5.3.5 Peripheral Protection Set Register 0 (PPROTSET0)

There is one bit for each quadrant for PS0 to PS7.

The following are the ways in which quadrants are used within a PS frame-

a. The slave uses all the four quadrants.

   Only the bit corresponding to the quadrant 0 of PSn is implemented. It protects the whole 1K-byte frame. The remaining three bits are not implemented.

b. The slave uses two quadrants.

   Each quadrant has to be in one of these groups: (Quad 0 and Quad 1) or (Quad 2 and Quad 3).

   For the group Quad0/Quad1, the bit quadrant 0 protects both quadrants 0 and 1. The bit quadrant 1 is not implemented.

   For the group Quad2/Quad3, the bit quadrant 2 protects both quadrants 2 and 3. The bit quadrant 3 is not implemented

c. The slave uses only one quadrant.

   In this case, the bit, as specified in Table 2-89, protects the slave.

The above arrangement is true for all the peripheral select (PS0 to PS31), presented in Section 2.5.3.6 - Section 2.5.3.12. This register holds bits for PS0 to PS7 and is shown in Figure 2-73 and described in Table 2-89.

NOTE: Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-73. Peripheral Protection Set Register 0 (PPROTSET0) [offset = 20h]

31 ... 0
PS[7-0]QUAD[3-0]PROTSET
R/WP-0

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

Table 2-89. Peripheral Protection Set Register 0 (PPROTSET0) Field Descriptions

Bit	Field	Value	Description
31-0	PS[7-0]QUAD[3-0] PROTSET		Peripheral select quadrant protection set.
		0	*Read:* The peripheral select quadrant an be written to and read from in both user and privileged modes. *Write:* The bit is unchanged.
		1	*Read:* The peripheral select quadrant can be written to only in privileged mode, but it can be read in both user and privileged modes. *Write:* The corresponding bit in PPROTSET0 and PPROTCLR0 registers is set to 1.

2.5.3.6 Peripheral Protection Set Register 1 (PPROTSET1)

There is one bit for each quadrant for PS8 to PS15. The protection scheme is described in Section 2.5.3.5. This register is shown in Figure 2-74 and described in Table 2-90.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-74. Peripheral Protection Set Register 1 (PPROTSET1) [offset = 24h]

31	0
PS[15-8]QUAD[3-0]PROTSET	
R/WP-0	

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

Table 2-90. Peripheral Protection Set Register 1 (PPROTSET1) Field Descriptions

Bit	Field	Value	Description
31-0	PS[15-8]QUAD[3-0] PROTSET		Peripheral select quadrant protection set.
		0	*Read:* The peripheral select quadrant can be written to and read from in both user and privileged modes. *Write:* The bit is unchanged.
		1	*Read:* The peripheral select quadrant can be written to only in privileged mode, but it can be read in both user and privileged modes. *Write:* The corresponding bit in PPROTSET1 and PPROTCLR1 registers is set to 1.

2.5.3.7 Peripheral Protection Set Register 2 (PPROTSET2)

There is one bit for each quadrant for PS16 to PS23. The protection scheme is described in Section 2.5.3.5. This register is shown in Figure 2-75 and described in Table 2-91.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-75. Peripheral Protection Set Register 2 (PPROTSET2) [offset = 28h]

31	0
PS[23-16]QUAD[3-0]PROTSET	
R/WP-0	

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

Table 2-91. Peripheral Protection Set Register 2 (PPROTSET2) Field Descriptions

Bit	Field	Value	Description
31-0	PS[23-16]QUAD[3-0] PROTSET		Peripheral select quadrant protection set.
		0	*Read:* The peripheral select quadrant can be written to and read from in both user and privileged modes. *Write:* The bit is unchanged.
		1	*Read:* The peripheral select quadrant can be written to only in privileged mode, but it can be read in both user and privileged modes. *Write:* The corresponding bit in PPROTSET2 and PPROTCLR2 registers is set to 1.

2.5.3.8 Peripheral Protection Set Register 3 (PPROTSET3)

There is one bit for each quadrant for PS24 to PS31. The protection scheme is described in Section 2.5.3.5. This register is shown in Figure 2-76 and described in Table 2-92.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-76. Peripheral Protection Set Register 3 (PPROTSET3) [offset = 2Ch]

31	0
PS[31-24]QUAD[3-0]PROTSET	
R/WP-0	

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

Table 2-92. Peripheral Protection Set Register 3 (PPROTSET3) Field Descriptions

Bit	Field	Value	Description
31-0	PS[31-24]QUAD[3-0] PROTSET		Peripheral select quadrant protection set.
		0	*Read:* The peripheral select quadrant can be written to and read from in both user and privileged modes. *Write:* The bit is unchanged.
		1	*Read:* The peripheral select quadrant can be written to only in privileged mode, but it can be read in both user and privileged modes. *Write:* The corresponding bit in PPROTSET3 and PPROTCLR3 registers is set to 1.

2.5.3.9 Peripheral Protection Clear Register 0 (PPROTCLR0)

There is one bit for each quadrant for PS0 to PS7. The protection scheme is described in Section 2.5.3.5. This register is shown in Figure 2-77 and described in Table 2-93.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-77. Peripheral Protection Clear Register 0 (PPROTCLR0) [offset = 40h]

31	0
PS[7-0]QUAD[3-0]PROTCLR	
R/WP-0	

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

Table 2-93. Peripheral Protection Clear Register 0 (PPROTCLR0) Field Descriptions

Bit	Field	Value	Description
31-0	PS[7-0]QUAD[3-0] PROTCLR		Peripheral select quadrant protection clear.
		0	*Read:* The peripheral select quadrant can be written to and read from in both user and privileged modes. *Write:* The bit is unchanged.
		1	*Read:* The peripheral select quadrant can be written to only in privileged mode, but it can be read in both user and privileged modes. *Write:* The corresponding bit in PPROTSET0 and PPROTCLR0 registers is cleared to 0.

2.5.3.10 Peripheral Protection Clear Register 1 (PPROTCLR1)

There is one bit for each quadrant for PS8 to PS15. The protection scheme is described in Section 2.5.3.5. This register is shown in Figure 2-78 and described in Table 2-94.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-78. Peripheral Protection Clear Register 1 (PPROTCLR1) [offset = 44h]

31	0
PS[15-8]QUAD[3-0]PROTCLR	
R/WP-0	

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

Table 2-94. Peripheral Protection Clear Register 1 (PPROTCLR1) Field Descriptions

Bit	Field	Value	Description
31-0	PS[15-8]QUAD[3-0] PROTCLR		Peripheral select quadrant protection clear.
		0	*Read:* The peripheral select quadrant can be written to and read from in both user and privileged modes. *Write:* The bit is unchanged.
		1	*Read:* The peripheral select quadrant can be written to only in privileged mode, but it can be read in both user and privileged modes. *Write:* The corresponding bit in PPROTSET1 and PPROTCLR1 registers is cleared to 0.

2.5.3.11 Peripheral Protection Clear Register 2 (PPROTCLR2)

There is one bit for each quadrant for PS16 to PS23. The protection scheme is described in Section 2.5.3.5. This register is shown in Figure 2-79 and described in Table 2-95.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-79. Peripheral Protection Clear Register 2 (PPROTCLR2) [offset = 48h]

31	0
PS[23-16]QUAD[3-0]PROTCLR	
R/WP-0	

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

Table 2-95. Peripheral Protection Clear Register 2 (PPROTCLR2) Field Descriptions

Bit	Field	Value	Description
31-0	PS[23-16]QUAD[3-0] PROTCLR		Peripheral select quadrant protection clear.
		0	*Read:* The peripheral select quadrant can be written to and read from in both user and privileged modes. *Write:* The bit is unchanged.
		1	*Read:* The peripheral select quadrant can be written to only in privileged mode, but it can be read in both user and privileged modes. *Write:* The corresponding bit in PPROTSET2 and PPROTCLR2 registers is cleared to 0.

2.5.3.12 Peripheral Protection Clear Register 3 (PPROTCLR3)

There is one bit for each quadrant for PS24 to PS31. The protection scheme is described in Section 2.5.3.5. This register is shown in Figure 2-80 and described in Table 2-96.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-80. Peripheral Protection Clear Register 3 (PPROTCLR3) [offset = 4Ch]

31	0
PS[31-24]QUAD[3-0]PROTCLR	
R/WP-0	

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

Table 2-96. Peripheral Protection Clear Register 3 (PPROTCLR3) Field Descriptions

Bit	Field	Value	Description
31-0	PS[31-24]QUAD[3-0] PROTCLR		Peripheral select quadrant protection clear.
		0	*Read:* The peripheral select quadrant can be written to and read from in both user and privileged modes. *Write:* The bit is unchanged.
		1	*Read:* The peripheral select quadrant can be written to only in privileged mode, but it can be read in both user and privileged modes. *Write:* The corresponding bit in PPROTSET3 and PPROTCLR3 registers is cleared to 0.

2.5.3.13 Peripheral Memory Power-Down Set Register 0 (PCSPWRDWNSET0)

Each bit corresponds to a bit at the same index in the PMPROT register in that they both relate to the same peripheral. This register is shown in Figure 2-81 and described in Table 2-97.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-81. Peripheral Memory Power-Down Set Register 0 (PCSPWRDWNSET0) [offset = 60h]

31	0
PCS[31-0]PWRDNSET	
R/WP-1	

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

Table 2-97. Peripheral Memory Power-Down Set Register 0 (PCSPWRDWNSET0) Field Descriptions

Bit	Field	Value	Description
31-0	PCS[31-0]PWRDNSET		Peripheral memory clock power-down set.
		0	*Read:* The peripheral memory clock[31-0] is active. *Write:* The bit is unchanged.
		1	*Read:* The peripheral memory clock[31-0] is inactive. *Write:* The corresponding bit in the PCSPWRDWNSET0 and PCSPWRDWNCLR0 registers is set to 1.

2.5.3.14 Peripheral Memory Power-Down Set Register 1 (PCSPWRDWNSET1)

This register is shown in Figure 2-82 and described in Table 2-98.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-82. Peripheral Memory Power-Down Set Register 1 (PCSPWRDWNSET1) [offset = 64h]

31	0
PCS[63-32]PWRDNSET	
R/WP-1	

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

Table 2-98. Peripheral Memory Power-Down Set Register 1 (PCSPWRDWNSET1) Field Descriptions

Bit	Field	Value	Description
31-0	PCS[63-32]PWRDNSET		Peripheral memory clock power-down set.
		0	*Read:* The peripheral memory clock[63-32] is active. *Write:* The bit is unchanged.
		1	*Read:* The peripheral memory clock[63-32] is inactive. *Write:* The corresponding bit in the PCSPWRDWNSET1 and PCSPWRDWNCLR1 registers is set to 1.

2.5.3.15 Peripheral Memory Power-Down Clear Register 0 (PCSPWRDWNCLR0)

This register is shown in Figure 2-83 and described in Table 2-99.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-83. Peripheral Memory Power-Down Clear Register 0 (PCSPWRDWNCLR0) [offset = 70h]

31	0
PCS[31-0]PWRDNCLR	
R/WP-1	

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

Table 2-99. Peripheral Memory Power-Down Clear Register 0 (PCSPWRDWNCLR0) Field Descriptions

Bit	Field	Value	Description
31-0	PCS[31-0]PWRDNCLR		Peripheral memory clock power-down clear.
		0	*Read:* The peripheral memory clock[31-0] is active. *Write:* The bit is unchanged.
		1	*Read:* The peripheral memory clock[31-0] is inactive. *Write:* The corresponding bit in the PCSPWRDWNSET0 and PCSPWRDWNCLR0 registers is cleared to 0.

2.5.3.16 Peripheral Memory Power-Down Clear Register 1 (PCSPWRDWNCLR1)

This register is shown in Figure 2-84 and described in Table 2-100.

> **NOTE:** Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-84. Peripheral Memory Power-Down Clear Register 1 (PCSPWRDWNCLR1) [offset = 74h]

31 … 0
PCS[63-32]PWRDNCLR
R/WP-1

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

Table 2-100. Peripheral Memory Power-Down Set Register 1 (PCSPWRDWNCLR1) Field Descriptions

Bit	Field	Value	Description
31-0	PCS[63-32]PWRDNCLR		Peripheral memory clock power-down clear.
		0	*Read:* The peripheral memory clock[63-32] is active. *Write:* The bit is unchanged.
		1	*Read:* The peripheral memory clock[63-32] is inactive. *Write:* The corresponding bit in the PCSPWRDWNSET1 and PCSPWRDWNCLR1 registers is cleared to 0.

2.5.3.17 Peripheral Power-Down Set Register 0 (PSPWRDWNSET0)

There is one bit for each quadrant for PS0 to PS7. Each bit of this register corresponds to the bit at the same index in the corresponding PPROT register in that they relate to the same peripheral. These bits are used to power down/power up the clock to the corresponding peripheral.

For every bit implemented in the PPROT register, there is one bit in the PSnPWRDWN register, except when two peripherals (both in PS area) share buses. In that case, only one Power-Down bit is implemented, at the position corresponding to that peripheral whose quadrant comes first (the lower numbered).

The ways in which quadrants can be used within a frame are identical to what is described under PPROTSET0, Section 2.5.3.5.

This arrangement is the same for bits of PS8 to PS31, presented in Section 2.5.3.18 - Section 2.5.3.24. This register holds bits for PS0 to PS7. This register is shown in Figure 2-85 and described in Table 2-101.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-85. Peripheral Power-Down Set Register 0 (PSPWRDWNSET0) [offset = 80h]

31	0
PS[7-0]QUAD[3-0]PWRDWNSET	
R/WP-1	

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

Table 2-101. Peripheral Power-Down Set Register 0 (PSPWRDWNSET0) Field Descriptions

Bit	Field	Value	Description
31-0	PS[7-0]QUAD[3-0] PWRDWNSET		Peripheral select quadrant clock power-down set.
		0	*Read:* The clock to the peripheral select quadrant is active. *Write:* The bit is unchanged.
		1	*Read:* The clock to the peripheral select quadrant is inactive. *Write:* The corresponding bit in PSPWRDWNSET0 and PSPWRDWNCLR0 registers is set to 1.

2.5.3.18 Peripheral Power-Down Set Register 1 (PSPWRDWNSET1)

There is one bit for each quadrant for PS8 to PS15. The protection scheme is described in Section 2.5.3.17. This register is shown in Figure 2-86 and described in Table 2-102.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-86. Peripheral Power-Down Set Register 1 (PSPWRDWNSET1) [offset = 84h]

31	0
PS[15-8]QUAD[3-0]PWRDWNSET	
R/WP-1	

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

Table 2-102. Peripheral Power-Down Set Register 1 (PSPWRDWNSET1) Field Descriptions

Bit	Field	Value	Description
31-0	PS[15-8]QUAD[3-0] PWRDWNSET		Peripheral select quadrant clock power-down set.
		0	*Read:* The clock to the peripheral select quadrant is active. *Write:* The bit is unchanged.
		1	*Read:* The clock to the peripheral select quadrant is inactive. *Write:* The corresponding bit in PSPWRDWNSET1 and PSPWRDWNCLR1 registers is set to 1.

2.5.3.19 Peripheral Power-Down Set Register 2 (PSPWRDWNSET2)

There is one bit for each quadrant for PS16 to PS23. The protection scheme is described in Section 2.5.3.17. This register is shown in Figure 2-87 and described in Table 2-103.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-87. Peripheral Power-Down Set Register 2 (PSPWRDWNSET2) [offset = 88h]

31	0
PS[23-16]QUAD[3-0]PWRDWNSET	
R/WP-1	

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

Table 2-103. Peripheral Power-Down Set Register 2 (PSPWRDWNSET2) Field Descriptions

Bit	Field	Value	Description
31-0	PS[23-16]QUAD[3-0] PWRDWNSET		Peripheral select quadrant clock power-down set.
		0	*Read:* The clock to the peripheral select quadrant is active. *Write:* The bit is unchanged.
		1	*Read:* The clock to the peripheral select quadrant is inactive. *Write:* The corresponding bit in PSPWRDWNSET2 and PSPWRDWNCLR2 registers is set to 1.

2.5.3.20 Peripheral Power-Down Set Register 3 (PSPWRDWNSET3)

There is one bit for each quadrant for PS24 to PS31. The protection scheme is described in Section 2.5.3.17. This register is shown in Figure 2-88 and described in Table 2-104.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-88. Peripheral Power-Down Set Register 3 (PSPWRDWNSET3) [offset = 8Ch]

31 — 0
PS[31-24]QUAD[3-0]PWRDWNSET
R/WP-1

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

Table 2-104. Peripheral Power-Down Set Register 3 (PSPWRDWNSET3) Field Descriptions

Bit	Field	Value	Description
31-0	PS[31-24]QUAD[3-0]PWRDWNSET		Peripheral select quadrant clock power-down set.
		0	*Read:* The clock to the peripheral select quadrant is active. *Write:* The bit is unchanged.
		1	*Read:* The clock to the peripheral select quadrant is inactive. *Write:* The corresponding bit in PSPWRDWNSET3 and PSPWRDWNCLR3 registers is set to 1.

2.5.3.21 Peripheral Power-Down Clear Register 0 (PSPWRDWNCLR0)

There is one bit for each quadrant for PS0 to PS7. The protection scheme is described in Section 2.5.3.17. This register is shown in Figure 2-89 and described in Table 2-105.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-89. Peripheral Power-Down Clear Register 0 (PSPWRDWNCLR0) [offset = A0h]

31 — 0
PS[7-0]QUAD[3-0]PWRDWNCLR
R/WP-1

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

Table 2-105. Peripheral Power-Down Clear Register 0 (PSPWRDWNCLR0) Field Descriptions

Bit	Field	Value	Description
31-0	PS[7-0]QUAD[3-0]PWRDWNCLR		Peripheral select quadrant clock power-down clear.
		0	*Read:* The clock to the peripheral select quadrant is active. *Write:* The bit is unchanged.
		1	*Read:* The clock to the peripheral select quadrant is inactive. *Write:* The corresponding bit in PSPWRDWNSET0 and PSPWRDWNCLR0 registers is cleared to 0.

2.5.3.22 Peripheral Power-Down Clear Register 1 (PSPWRDWNCLR1)

There is one bit for each quadrant for PS8 to PS15. The protection scheme is described in Section 2.5.3.17. This register is shown in Figure 2-90 and described in Table 2-106.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-90. Peripheral Power-Down Clear Register 1 (PSPWRDWNCLR1) [offset = A4h]

31 … 0
PS[15-8]QUAD[3-0]PWRDWNCLR
R/WP-1

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

Table 2-106. Peripheral Power-Down Clear Register 1 (PSPWRDWNCLR1) Field Descriptions

Bit	Field	Value	Description
31-0	PS[15-8]QUAD[3-0] PWRDWNCLR		Peripheral select quadrant clock power-down clear.
		0	*Read:* The clock to the peripheral select quadrant is active. *Write:* The bit is unchanged.
		1	*Read:* The clock to the peripheral select quadrant is inactive. *Write:* The corresponding bit in PSPWRDWNSET1 and PSPWRDWNCLR1 registers is cleared to 0.

2.5.3.23 Peripheral Power-Down Clear Register 2 (PSPWRDWNCLR2)

There is one bit for each quadrant for PS16 to PS23. The protection scheme is described in Section 2.5.3.17. This register is shown in Figure 2-91 and described in Table 2-107.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-91. Peripheral Power-Down Clear Register 2 (PSPWRDWNCLR2) [offset = A8h]

31 … 0
PS[23-16]QUAD[3-0]PWRDWNCLR
R/WP-1

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

Table 2-107. Peripheral Power-Down Clear Register 2 (PSPWRDWNCLR2) Field Descriptions

Bit	Field	Value	Description
31-0	PS[23-16]QUAD[3-0] PWRDWNCLR		Peripheral select quadrant clock power-down clear.
		0	*Read:* The clock to the peripheral select quadrant is active. *Write:* The bit is unchanged.
		1	*Read:* The clock to the peripheral select quadrant is inactive. *Write:* The corresponding bit in PSPWRDWNSET2 and PSPWRDWNCLR2 registers is cleared to 0.

2.5.3.24 Peripheral Power-Down Clear Register 3 (PSPWRDWNCLR3)

There is one bit for each quadrant for PS24 to PS31. The protection scheme is described in Section 2.5.3.17. This register is shown in Figure 2-92 and described in Table 2-108.

NOTE: Only those bits that have a slave at the corresponding bit position are implemented. Writes to nonimplemented bits have no effect and reads are 0.

Figure 2-92. Peripheral Power-Down Clear Register 3 (PSPWRDWNCLR) [offset = ACh]

31	0
PS[31-24]QUAD[3-0]PWRDWNCLR	
R/WP-1	

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

Table 2-108. Peripheral Power-Down Clear Register 3 (PSPWRDWNCLR3) Field Descriptions

Bit	Field	Value	Description
31-0	PS[31-24]QUAD[3-0] PWRDWNCLR		Peripheral select quadrant clock power-down clear.
		0	*Read:* The clock to the peripheral select quadrant is active. *Write:* The bit is unchanged.
		1	*Read:* The clock to the peripheral select quadrant is inactive. *Write:* The corresponding bit in PSPWRDWNSET3 and PSPWRDWNCLR3 registers is cleared to 0.

Power Management Module (PMM)

This chapter describes the power management module (PMM).

Topic **Page**

3.1 Overview **210**
3.2 Power Domains **212**
3.3 PMM Operation **213**
3.4 PMM Registers **216**

3.1 Overview

This microcontroller is part of the Hercules family of microcontrollers from Texas Instruments for safety-critical applications. Several functions are implemented on this microcontroller targeted towards varied applications. The core logic is divided into several domains that can be independently turned on or off based on the application's requirements. This allows an application to reduce the leakage current for a core domain that has modules that are not being used by the application.

This chapter describes the Power Management Module (PMM). The PMM provides memory-mapped registers that control the states of the supported power domains. The PMM includes interfaces to the Power Mode Controller (PMC) and the Power State Controller (PSCON). The PMC and PSCON control the power up/down sequence of each power domain.

3.1.1 Main Features of the Power Management Module (PMM)

The main features of the PMM implemented on the microcontroller are:

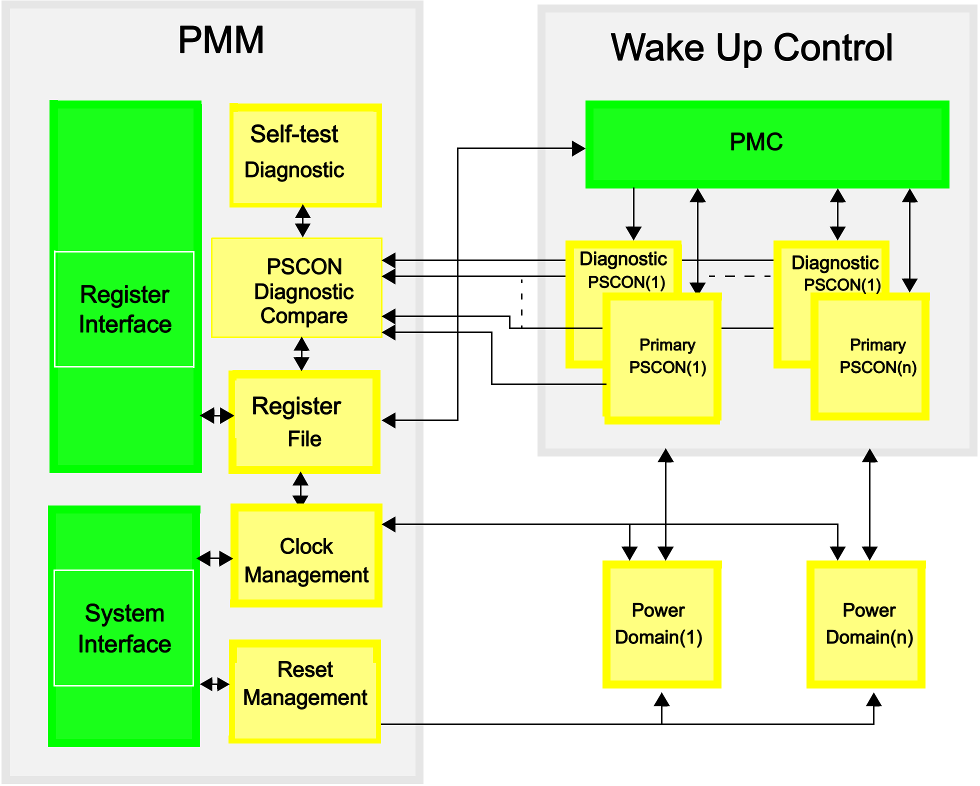
- Supports 5 logic power domains: PD1, PD2, PD3, PD4, and PD5
- Supports 2 memory-only power domains: RAM_PD1, RAM_PD2
- Manages the clocks for each power domain
- Manages the resets to each power domain
- Includes failsafe compare logic to continuously monitor the states of each power domain
- Supports diagnostic and self-test logic to validate failsafe compare logic

3.1.2 Block Diagram

Figure 3-1 is a block diagram of the PMM. The PMM consists of several key components:

- Register interface – the PMM control registers are mapped to the device memory space and start at address 0xFFFF0000.
- System Interface – the PMM receives the clocks, resets, errors and all other control signals through this interface.
- PSCON Diagnostic Compare – this block compares the outputs of each primary PSCON and the respective diagnostic PSCON implemented for failsafe safety.
- Self-Test Diagnostic – this block contains the logic to place the PSCON diagnostic compare block in a self-test mode in order to test the failsafe feature.
- Clock management – the PMM provides independent clock gating and handshaking controls for each power domain and also generates the clock domains for each power domain.
- Reset Management – the PMM provides independent reset signals for each power domain.
- Power State Controller (PSCON) – The PSCON is a finite state machine that controls the power sequence of a power domain from one state to another. Each power domain is controlled by one dedicated PSCON.
- Power Domain – A power domain is a group of logic and/or memories which is separated from the global power supply via power switches. These power switches are controlled by the PSCON and can be turned on or off.

Figure 3-1. PMM Block Diagram



3.2 Power Domains

Figure 3-2 shows the core and memory power domains implemented on the microcontroller.

This device has 7 separate core power domains:

- PD1 is an always-ON domain and is not controlled by PMM. It contains the CPU as well as other principal modules and the interconnect required for operation of the microcontroller. This domain also includes 64KB of the tightly-coupled RAM. The PD1 can operate on its own even when all the other core power domains are turned off by the PMM. Note that all I/Os are in this always-ON domain as well.

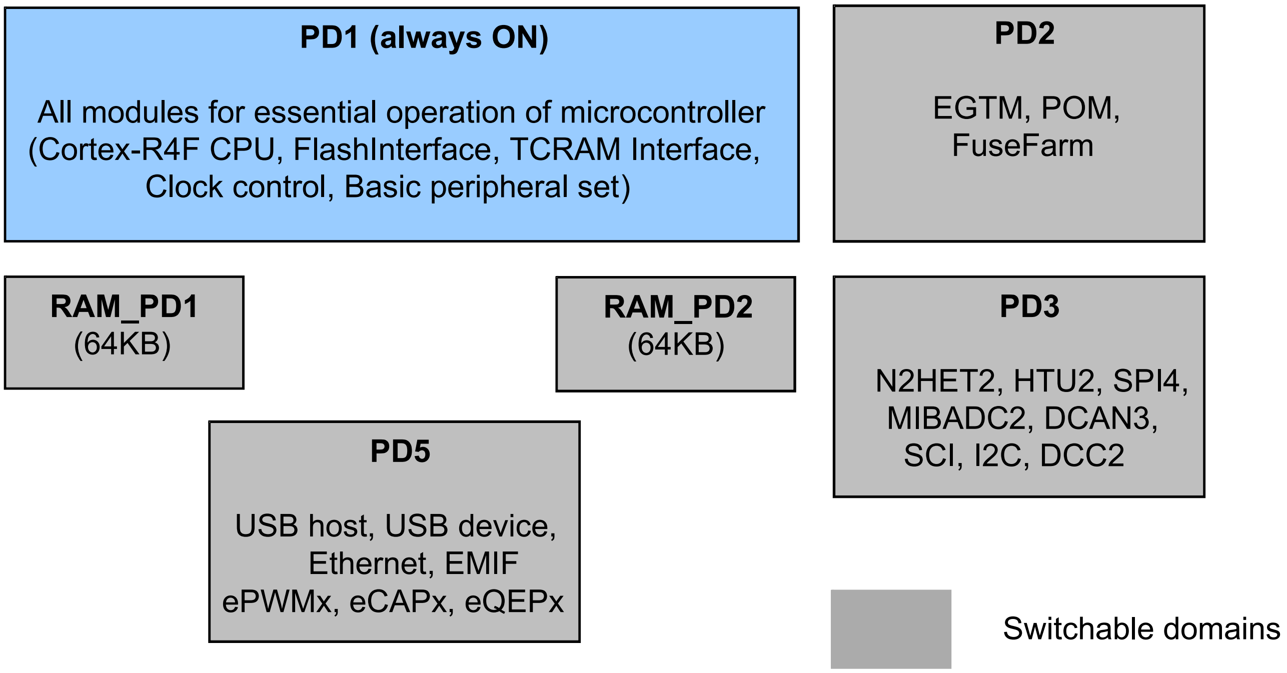
Core power domains PD2 through PD5 and RAM_PD1 through RAM_PD2 are controlled by the PMM.

- PD2 contains the Parameter Overlay Module (POM) and test logic.
- PD3 contains some additional peripheral modules as an enhanced configuration over and above the peripheral set available in PD1. These include a second High-End Timer (N2HET2) with its dedicated transfer unit (HTU2), a second Analog-to-Digital Converter (ADC2), a Serial Communication Interface (SCI), an Inter-Integrated Circuit controller (I2C), a third Controller Area Network controller (DCAN3), and a fourth Serial Peripheral Interface module (SPI4).
- PD4 is unavailable in this microcontroller.
- PD5 contains the Ethernet Controller (EMAC), External Memory Interface (EMIF), Enhanced Pulse Width Modulator (ePWMx), Enhanced Capture (eCAPx), and Enhanced Quadrature Encoder Pulse (eQEPx), as well as some components of the interconnect fabric required by these modules.
- RAM_PD1 and RAM_PD2 each contain 64KB of tightly-coupled RAM.

NOTE: **Switching of Power Domains**

The microcontrollers only support static switching of the power domains. That is, the power domains can be turned ON or OFF one time during device initialization. Once configured, it is not allowed to change the state of a power domain without first asserting a system reset.

Figure 3-2. Core Power Domains


PD1 (always ON)
All modules for essential operation of microcontroller (Cortex-R4F CPU, FlashInterface, TCRAM Interface, Clock control, Basic peripheral set)
PD2
EGTM, POM, FuseFarm
RAM_PD1 (64KB)
RAM_PD2 (64KB)
PD3
N2HET2, HTU2, SPI4, MIBADC2, DCAN3, SCI, I2C, DCC2
PD5
USB host, USB device, Ethernet, EMIF ePWMx, eCAPx, eQEPx
Switchable domains

3.3 PMM Operation

It is important to understand some fundamental concepts beforehand.

3.3.1 Power Switch

A power domain gets its power supply via a power switch. The power switch creates a link between the global core supply plane and the local switchable power domain supply. Each power domain uses multiple power switches, which are daisy-chained together.

3.3.2 Power Domain State

Each core power domain can be in one of three states: Active, Idle, or Off.

In the ***active*** state, a power domain is fully powered with normal supply voltage.

In the ***idle*** state, all clocks to a power domain are turned off (driven low). The supply voltage is still maintained at the normal level.

In the ***off*** state, a power domain is completely cut off from the power supply.

3.3.3 Default Power Domain State

The default state of each power domain, except for PD1, is controlled by TI during production testing via programmation of individual bits within the reset configuration word in the TI-OTP sector of flash bank 0. This allows each power domain to default to either the active state or the off state.

3.3.4 Disabling a Power Domain Permanently

TI can also permanently disable any power domain, except for PD1. This is also controlled by programmation of individual bits within the reset configuration word in the TI-OTP sector of flash bank 0.

3.3.5 Changing Power Domain State

A domain can only change state when commanded by the application. Each domain has an associated 4-bit key to define the intended power state. When the correct key is programmed, the PMM initiates the sequence to transition that domain to the commanded state.

A power state transition is considered complete only when every single power switch for that domain has switched over to the commanded state.

3.3.5.1 Turning a Power Domain Off

It is necessary to turn off all clocks going to a power domain before that domain can be powered down. PMM contains the hardware interlocks to handle this. Each power domain has an associated memory-mapped register which allows the application to turn off clocks to that power domain.

Steps to power down a domain with logic – PD2, PD3, PD4, PD5:

1. Write to the PDCLK_DISx register to disable all clocks to the power domain.
2. Write 0xA to the LOGICPDPWRCTRL0 register to power down each domain.
3. Poll for LOGICPDPWRSTATx.LOGICPDPWR_STATx.to become “00”. The power domain is now powered down.

A power domain with only SRAM macros does not have a clock input, so the sequence is shorter. This applies to RAM_PD1 and RAM_PD2 power domains shown in Figure 3-2:

1. Write the correct key to the MEMPDONx register to power down the domain.
2. Poll for MEMPDPWRSTATx.MEMPDPWR_STATx to become “00”. The power domain is now powered down.

3.3.5.2 Turning a Power Domain On

A power domain can be turned on by writing the correct key to the LOGICPDPWRCTRL0.LOGICPDONx or MEMPDPWRCTRL0.MEMPDONx. PMM will automatically restart the clocks to the power domain once the power is restored if the “automatic clock enable upon wake up” option is selected. If this option is not selected, the application can turn on clocks to the power domain by clearing the PDCLK_DIS register. The application must poll the LOGICPDPWRSTATx.DOMAIN_ONx to ensure that the power has been fully restored before enabling the clocks.

NOTE: If a power domain has been permanently disabled by TI, the application cannot turn this power domain ON. No error is generated if the application attempts to do so.

3.3.6 Reset Management

PMM handles the reset sequence for each power domain. When a power domain is turned on from an off state, the PMM will reset the power domain to ensure that all logic begins in its default reset state.

PMM generates nPORRST (power-on reset), nRST (system reset), nPRST (peripheral reset), and nTRST (test / debug logic reset) for each domain.

3.3.7 Diagnostic Power State Controller (PSCON)

Each power domain state is controlled by a primary PSCON. There is a second PSCON as well for each power domain. This is the diagnostic PSCON. All power management inputs to a power domain are controlled only by the primary PSCON. All power management outputs from the power domain are fed back to both the primary and the diagnostic PSCON.

The PMM commands both the PSCON identically so that they are always in a lock-step operating mode. A dedicated compare unit checks the outputs of the two PSCON modules on every cycle.

3.3.8 PSCON Compare Block

The diagnostic compare block can operate in one of four modes:

3.3.8.1 Lock-Step Mode

This is the default mode of operation of the PSCON compare block. The PSCON diagnostic compare block compares the outputs from the two PSCONs on every cycle. Any mismatch in the PSCON outputs is indicated as a PSCON compare error. This error signal is mapped to the Error Signaling Module’s (ESM) Group1 channel 38. The application can define the response to this error.

3.3.8.2 Self-Test Mode

A self-test mechanism is provided to check the PSCON compare logic for faults. The compare error signal output is disabled in self-test mode. The PSCON diagnostic compare block generates two types of patterns during self-test mode: compare match test followed by compare mismatch test. During the self-test, each test pattern is applied on both PSCON signal ports of the PSCON diagnostic compare block and then is clocked for one cycle. The duration of the self-test is 24 cycles. Any detected fault is indicated as a self-test error, mapped to ESM group1 channel 39. If no fault is detected, the self-test complete flag is set.

The application can poll for this flag to be set and then switch the mode of the PSCON compare block back to lock-step mode by writing to the mode key register.

NOTE: PSCON operation when compare block is in self-test mode

When the PSCON compare block is in its self-test mode, both PSCONs continue to function normally. However, there is no comparison done on the PSCON outputs.

Compare match test:

An identical vector is applied to both input ports at the same time, thereby expecting a compare match. If the compare unit produces a mismatch then the self-test error flag is set and the self-test error signal is generated. The compare match test is terminated if a compare mismatch is detected. The compare match test takes 4 cycles to complete when the test passes.

Compare mismatch test:

A vector with all 1's is applied to the PSCON diagnostic compare block’s primary input port and the same input is also applied to the secondary input port but with one bit flipped starting from bit position 0. The unequal vectors should cause the PSCON diagnostic compare block to generate a compare mismatch at bit position 0. In case a mismatch is not detected, a self-test error is indicated. This compare mismatch test algorithm is repeated until every single bit position is verified on both PSCON signal ports.

3.3.8.3 Error-Forcing Mode

This mode is designed specifically to ensure that the error signal output from the PSCON compare block is not stuck inactive. In this mode, a test pattern is applied to the PSCON related inputs of the compare logic to force an error. During error forcing mode, both the error signal and the self-test error signal will be asserted to the ESM. The application can clear flags for ESM group1 channel 38 and ESM group1 channel 39 once the error is flagged. If the two ESM flags do not get set, this indicates that the PSCON compare error signal is stuck inactive and cannot be relied upon to detect a PSCON mismatch.

3.3.8.4 Self-Test Error-Forcing Mode

In this mode, an error is forced so that the self-test error output from the PSCON compare block is activated. The application can clear the flag for ESM group1 channel 39 once the error is flagged. If the ESM group1 channel 39 flag does not get set, this indicates that the PSCON compare block self-test error signal is stuck inactive and there is no self-test mechanism available for the PSCON compare block.

3.3.8.5 PMM Operation During CPU Halt Debug Mode

No compare errors are generated when the CPU is halted in debug mode, regardless of the mode of the diagnostic compare block. No status flags are updated in this mode. Normal operation of the compare block is resumed once the CPU exits the debug mode.

3.4 PMM Registers

Table 3-1 lists the memory-mapped registers for the PMM. All register offset addresses not listed in Table 3-1 should be considered as reserved locations and the register contents should not be modified. The address offset is specified from the base address of FFFF 0000h. Any access to an unimplemented location within the PMM register frame will generate a bus error that results in an Abort exception.

Table 3-1. PMM Registers

Offset	Acronym	Register Name	Section
0h	LOGICPDPWRCTRL0	Logic Power Domain Control Register 0	Section 3.4.1
10h	MEMPDPWRCTRL0	Memory Power Domain Control Register 0	Section 3.4.2
20h	PDCLKDIS	Power Domain Clock Disable Register	Section 3.4.3
24h	PDCLKDISSET	Power Domain Clock Disable Set Register	Section 3.4.4
28h	PDCLKDISCLR	Power Domain Clock Disable Clear Register	Section 3.4.5
40h	LOGICPDPWRSTAT0	Logic Power Domain PD2 Power Status Register	Section 3.4.6
44h	LOGICPDPWRSTAT1	Logic Power Domain PD3 Power Status Register	Section 3.4.7
48h	LOGICPDPWRSTAT2	Logic Power Domain PD4 Power Status Register	Section 3.4.8
4Ch	LOGICPDPWRSTAT3	Logic Power Domain PD5 Power Status Register	Section 3.4.9
80h	MEMPDPWRSTAT0	Memory Power Domain RAM_PD1 Power Status Register	Section 3.4.10
84h	MEMPDPWRSTAT1	Memory Power Domain RAM_PD2 Power Status Register	Section 3.4.11
A0h	GLOBALCTRL1	Global Control Register 1	Section 3.4.12
A8h	GLOBALSTAT	Global Status Register	Section 3.4.13
ACh	PRCKEYREG	PSCON Diagnostic Compare Key Register	Section 3.4.14
B0h	LPDDCSTAT1	LogicPD PSCON Diagnostic Compare Status Register 1	Section 3.4.15
B4h	LPDDCSTAT2	LogicPD PSCON Diagnostic Compare Status Register 2	Section 3.4.16
B8h	MPDDCSTAT1	Memory PD PSCON Diagnostic Compare Status Register 1	Section 3.4.17
BCh	MPDDCSTAT2	Memory PD PSCON Diagnostic Compare Status Register 2	Section 3.4.18
C0h	ISODIAGSTAT	Isolation Diagnostic Status Register	Section 3.4.19

3.4.1 LOGICPDPWRCTRL0 Register (Offset = 0h) [reset = X]

LOGICPDPWRCTRL0 is shown in Figure 3-3 and described in Table 3-2.

The default values of the control fields are determined by the device reset configuration word stored in the TI-OTP region of flash bank 0.

Figure 3-3. LOGICPDPWRCTRL0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED				LOGICPDON0				RESERVED				LOGICPDON1			
R/W-0h				R/WP-X				R-0h				R/WP-X			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RESERVED				LOGICPDON2				RESERVED				LOGICPDON3			
R-0h				R/WP-X				R-0h				R/WP-X			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 3-2. LOGICPDPWRCTRL0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	RESERVED	R/W	0h	Reads return 0. Writes have no effect.
27-24	LOGICPDON0	R/WP	X	Read in User and Privileged Mode. Write in Privileged Mode only. Read any other value: Power domain PD2 is in Active state. Write any other value: Power domain PD2 is commanded to switch to Active state. 9h = Reserved. Ah (W) = Power domain PD2 is commanded to switch to OFF state. Ah (R) = Power domain PD2 is in OFF state.
23-20	RESERVED	R	0h	Reads return 0. Writes have no effect.
19-16	LOGICPDON1	R/WP	X	Read in User and Privileged Mode. Write in Privileged Mode only. Read any other value: Power domain PD3 is in Active state. Write any other value: Power domain PD3 is commanded to switch to Active state. 9h = Reserved. Ah (W) = Power domain PD3 is commanded to switch to OFF state. Ah (R) = Power domain PD3 is in OFF state.
15-12	RESERVED	R	0h	Reads return 0. Writes have no effect.
11-8	LOGICPDON2	R/WP	X	Read in User and Privileged Mode. Write in Privileged Mode only. Read any other value: Power domain PD4 is in Active state. Write any other value: Power domain PD4 is commanded to switch to Active state. 9h = Reserved. Ah (W) = Power domain PD4 is commanded to switch to OFF state. Ah (R) = Power domain PD4 is in OFF state.
7-4	RESERVED	R	0h	Reads return 0. Writes have no effect.

Table 3-2. LOGICPDPWRCTRL0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3-0	LOGICPDON3	R/WP	X	Read in User and Privileged Mode. Write in Privileged Mode only. Read any other value: Power domain PD5 is in Active state. Write any other value: Power domain PD5 is commanded to switch to Active state. 9h = Reserved. Ah (W) = Power domain PD5 is commanded to switch to OFF state. Ah (R) = Power domain PD5 is in OFF state.

3.4.2 MEMPDPWRCTRL0 Register (Offset = 10h) [reset = X]

MEMPDPWRCTRL0 is shown in Figure 3-4 and described in Table 3-3.

The default values of the control fields are determined by the device reset configuration word stored in the TI-OTP region of flash bank 0.

Figure 3-4. MEMPDPWRCTRL0 Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="4">RESERVED</td><td colspan="4">MEMPDON0</td><td colspan="4">RESERVED</td><td colspan="4">MEMPDON1</td></tr>
<tr><td colspan="4">R-0h</td><td colspan="4">R/WP-X</td><td colspan="4">R-0h</td><td colspan="4">R/WP-X</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">RESERVED</td></tr>
<tr><td colspan="16">R-0h</td></tr>
</table>

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 3-3. MEMPDPWRCTRL0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	Reads return 0. Writes have no effect.
27-24	MEMPDON0	R/WP	X	Read in User and Privileged Mode. Write in Privileged Mode only. Read any other value: Power domain RAM_PD1 is in Active state. Write any other value: Power domain RAM_PD1 is commanded to switch to Active state. 9h = Reserved. Ah (W) = Power domain RAM_PD1 is commanded to switch to OFF state. Ah (R) = Power domain RAM_PD1 is in OFF state.
23-20	RESERVED	R	0h	Reads return 0. Writes have no effect.
19-16	MEMPDON1	R/WP	X	Read in User and Privileged Mode. Write in Privileged Mode only. Read any other value: Power domain RAM_PD2 is in Active state. Write any other value: Power domain RAM_PD2 is commanded to switch to Active state. 9h = Reserved. Ah (W) = Power domain RAM_PD2 is commanded to switch to OFF state. Ah (R) = Power domain RAM_PD2 is in OFF state.
15-0	RESERVED	R	0h	Reads return 0. Writes have no effect.

3.4.3 PDCLKDIS Register (Offset = 20h) [reset = X]

PDCLKDIS is shown in Figure 3-5 and described in Table 3-4.

The default values of the control fields are determined by the device reset configuration word stored in the TI-OTP region of flash bank 0.

Figure 3-5. PDCLKDIS Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				PDCLK_ DIS_3	PDCLK_ DIS_2	PDCLK_ DIS_1	PDCLK_ DIS_0
R-0h				R/WP-X	R/WP-X	R/WP-X	R/WP-X

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 3-4. PDCLKDIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reads return 0. Writes have no effect.
3	PDCLK_DIS_3	R/WP	X	Read in User and Privileged Mode returns the current value of PDCLK_DIS[3]. Write in Privileged Mode only. 0h = Enable clocks to logic power domain PD5. 1h = Disable clocks to logic power domain PD5.
2	PDCLK_DIS_2	R/WP	X	Read in User and Privileged Mode returns the current value of PDCLK_DIS[2]. Write in Privileged Mode only. 0h = Enable clocks to logic power domain PD4. 1h = Disable clocks to logic power domain PD4.
1	PDCLK_DIS_1	R/WP	X	Read in User and Privileged Mode returns the current value of PDCLK_DIS[1]. Write in Privileged Mode only. 0h = Enable clocks to logic power domain PD3. 1h = Disable clocks to logic power domain PD3.
0	PDCLK_DIS_0	R/WP	X	Read in User and Privileged Mode returns the current value of PDCLK_DIS[0]. Write in Privileged Mode only. 0h = Enable clocks to logic power domain PD2. 1h = Disable clocks to logic power domain PD2.

3.4.4 PDCLKDISSET Register (Offset = 24h) [reset = X]

PDCLKDISSET is shown in Figure 3-6 and described in Table 3-5.

The default values of the control fields are determined by the device reset configuration word stored in the TI-OTP region of flash bank 0.

Figure 3-6. PDCLKDISSET Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				PDCLK_ DISSET_3	PDCLK_ DISSET_2	PDCLK_ DISSET_1	PDCLK_ DISSET_0
R-0h				R/WP-X	R/WP-X	R/WP-X	R/WP-X

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 3-5. PDCLKDISSET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reads return 0. Writes have no effect.
3	PDCLK_DISSET_3	R/WP	X	Read in User and Privileged Mode returns the current value of PDCLK_DISSET[3]. Write in Privileged Mode only. 0h = No effect to state of clocks to power domain PD5. 1h = Disable clocks to logic power domain PD5.
2	PDCLK_DISSET_2	R/WP	X	Read in User and Privileged Mode returns the current value of PDCLK_DISSET[2]. Write in Privileged Mode only. 0h = No effect to state of clocks to power domain PD4. 1h = Disable clocks to logic power domain PD4.
1	PDCLK_DISSET_1	R/WP	X	Read in User and Privileged Mode returns the current value of PDCLK_DISSET[1]. Write in Privileged Mode only. 0h = No effect to state of clocks to power domain PD3. 1h = Disable clocks to logic power domain PD3.
0	PDCLK_DISSET_0	R/WP	X	Read in User and Privileged Mode returns the current value of PDCLK_DISSET[0]. Write in Privileged Mode only. 0h = No effect to state of clocks to power domain PD2. 1h = Disable clocks to logic power domain PD2.

3.4.5 PDCLKDISCLR Register (Offset = 28h) [reset = X]

PDCLKDISCLR is shown in Figure 3-7 and described in Table 3-6.

The default values of the control fields are determined by the device reset configuration word stored in the TI-OTP region of flash bank 0.

Figure 3-7. PDCLKDISCLR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				PDCLK_ DISCLR_3	PDCLK_ DISCLR_2	PDCLK_ DISCLR_1	PDCLK_ DISCLR_0
R-0h				R/WP-X	R/WP-X	R/WP-X	R/WP-X

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 3-6. PDCLKDISCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reads return 0. Writes have no effect.
3	PDCLK_DISCLR_3	R/WP	X	Read in User and Privileged Mode returns the current value of PDCLK_DIS[3]. Write in Privileged Mode only. 0h = No effect to state of clocks to power domain PD5. 1h = Enable clocks to logic power domain PD5.
2	PDCLK_DISCLR_2	R/WP	X	Read in User and Privileged Mode returns the current value of PDCLK_DIS[2]. Write in Privileged Mode only. 0h = No effect to state of clocks to power domain PD4. 1h = Enable clocks to logic power domain PD4.
1	PDCLK_DISCLR_1	R/WP	X	Read in User and Privileged Mode returns the current value of PDCLK_DIS[1]. Write in Privileged Mode only. 0h = No effect to state of clocks to power domain PD3. 1h = Enable clocks to logic power domain PD3.
0	PDCLK_DISCLR_0	R/WP	X	Read in User and Privileged Mode returns the current value of PDCLK_DIS[0]. Write in Privileged Mode only. 0h = No effect to state of clocks to power domain PD2. 1h = Enable clocks to logic power domain PD2.

3.4.6 LOGICPDPWRSTAT0 Register (Offset = 40h) [reset = X]

LOGICPDPWRSTAT0 is shown in Figure 3-8 and described in Table 3-7.

This is a read-only register. All writes are ignored.

The default values of the control fields are determined by the device reset configuration word stored in the TI-OTP region of flash bank 0.

Figure 3-8. LOGICPDPWRSTAT0 Register

31	30	29	28	27	26	25	24
RESERVED							LOGIC_IN_TRANS0
R-0h							R-X
23	22	21	20	19	18	17	16
RESERVED							MEM_IN_TRANS0
R-0h							R-X
15	14	13	12	11	10	9	8
RESERVED							DOMAIN_ON0
R-0h							R-X
7	6	5	4	3	2	1	0
RESERVED						LOGICPDPWR_STAT0	
R-0h						R-X	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 3-7. LOGICPDPWRSTAT0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reads return 0. Writes have no effect.
24	LOGIC_IN_TRANS0	R	X	Logic in transition status for power domain PD2. Read in User and Privileged Mode. 0h = Logic in power domain PD2 is in the steady Active or OFF state. 1h = Logic in power domain PD2 is in the process of power-down/up.
23-17	RESERVED	R	0h	Reads return 0. Writes have no effect.
16	MEM_IN_TRANS0	R	X	Memory in transition status for power domain PD2. Read in User and Privileged Mode. 0h = Memory in power domain PD2 is in the steady Active or OFF state. 1h = Memory in power domain PD2 is in the process of power-down/up.
15-9	RESERVED	R	0h	Reads return 0. Writes have no effect.
8	DOMAIN_ON0	R	X	Current state of power domain PD2. The default value of this field is controlled by the device reset configuration word in the TI-OTP region of flash bank 0. Read in User and Privileged Mode. 0h = Power domain PD2 is in the OFF state. 1h = Power domain PD2 is in the Active state.
7-2	RESERVED	R	0h	Reads return 0. Writes have no effect.
1-0	LOGICPDPWR_STAT0	R	X	Logic power domain PD2 power state. The default value of this field is controlled by the device reset configuration word in the TI-OTP region of flash bank 0. Read in User and Privileged Mode. 0h = Logic power domain PD2 is switched OFF. 1h = Logic power domain PD2 is in Idle state. 2h = Reserved 3h = Logic power domain PD2 is in Active state.

3.4.7 LOGICPDPWRSTAT1 Register (Offset = 44h) [reset = X]

LOGICPDPWRSTAT1 is shown in Figure 3-9 and described in Table 3-8.

This is a read-only register. All writes are ignored.

The default values of the control fields are determined by the device reset configuration word stored in the TI-OTP region of flash bank 0.

Figure 3-9. LOGICPDPWRSTAT1 Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td></tr>
<tr><td colspan="7">RESERVED</td><td>LOGIC_IN_ TRANS1</td></tr>
<tr><td colspan="7">R-0h</td><td>R-X</td></tr>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="7">RESERVED</td><td>MEM_IN_ TRANS1</td></tr>
<tr><td colspan="7">R-0h</td><td>R-X</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="7">RESERVED</td><td>DOMAIN_ON1</td></tr>
<tr><td colspan="7">R-0h</td><td>R-X</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="6">RESERVED</td><td colspan="2">LOGICPDPWR_STAT1</td></tr>
<tr><td colspan="6">R-0h</td><td colspan="2">R-X</td></tr>
</table>

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 3-8. LOGICPDPWRSTAT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reads return 0. Writes have no effect.
24	LOGIC_IN_TRANS1	R	X	Logic in transition status for power domain PD3. Read in User and Privileged Mode. 0h = Logic in power domain PD3 is in the steady Active or OFF state. 1h = Logic in power domain PD3 is in the process of power-down/up.
23-17	RESERVED	R	0h	Reads return 0. Writes have no effect.
16	MEM_IN_TRANS1	R	X	Memory in transition status for power domain PD3. Read in User and Privileged Mode. 0h = Memory in power domain PD3 is in the steady Active or OFF state. 1h = Memory in power domain PD3 is in the process of power-down/up.
15-9	RESERVED	R	0h	Reads return 0. Writes have no effect.
8	DOMAIN_ON1	R	X	Current state of power domain PD3. The default value of this field is controlled by the device reset configuration word in the TI-OTP region of flash bank 0. Read in User and Privileged Mode. 0h = Power domain PD3 is in the OFF state. 1h = Power domain PD3 is in the Active state.
7-2	RESERVED	R	0h	Reads return 0. Writes have no effect.
1-0	LOGICPDPWR_STAT1	R	X	Logic power domain PD3 power state. The default value of this field is controlled by the device reset configuration word in the TI-OTP region of flash bank 0. Read in User and Privileged Mode. 0h = Logic power domain PD3 is switched OFF. 1h = Logic power domain PD3 is in Idle state. 2h = Reserved 3h = Logic power domain PD3 is in Active state.

3.4.8 LOGICPDPWRSTAT2 Register (Offset = 48h) [reset = X]

LOGICPDPWRSTAT2 is shown in Figure 3-10 and described in Table 3-9.

This is a read-only register. All writes are ignored.

The default values of the control fields are determined by the device reset configuration word stored in the TI-OTP region of flash bank 0.

Figure 3-10. LOGICPDPWRSTAT2 Register

31	30	29	28	27	26	25	24
RESERVED							LOGIC_IN_TRANS2
R-0h							R-X
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RESERVED							MEM_IN_TRANS2
R-0h							R-X
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RESERVED							DOMAIN_ON2
R-0h							R-X
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RESERVED						LOGICPDPWR_STAT2	
R-0h						R-X	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 3-9. LOGICPDPWRSTAT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reads return 0. Writes have no effect.
24	LOGIC_IN_TRANS2	R	X	Logic in transition status for power domain PD4. Read in User and Privileged Mode. 0h = Logic in power domain PD4 is in the steady Active or OFF state. 1h = Logic in power domain PD4 is in the process of power-down/up.
23-17	RESERVED	R	0h	Reads return 0. Writes have no effect.
16	MEM_IN_TRANS2	R	X	Memory in transition status for power domain PD4. Read in User and Privileged Mode. 0h = Memory in power domain PD4 is in the steady Active or OFF state. 1h = Memory in power domain PD4 is in the process of power-down/up.
15-9	RESERVED	R	0h	Reads return 0. Writes have no effect.
8	DOMAIN_ON2	R	X	Current state of power domain PD4. The default value of this field is controlled by the device reset configuration word in the TI-OTP region of flash bank 0. Read in User and Privileged Mode. 0h = Power domain PD4 is in the OFF state. 1h = Power domain PD4 is in the Active state.
7-2	RESERVED	R	0h	Reads return 0. Writes have no effect.
1-0	LOGICPDPWR_STAT2	R	X	Logic power domain PD4 power state. The default value of this field is controlled by the device reset configuration word in the TI-OTP region of flash bank 0. Read in User and Privileged Mode. 0h = Logic power domain PD4 is switched OFF. 1h = Logic power domain PD4 is in Idle state. 2h = Reserved 3h = Logic power domain PD4 is in Active state.

3.4.9 LOGICPDPWRSTAT3 Register (Offset = 4Ch) [reset = X]

LOGICPDPWRSTAT3 is shown in Figure 3-11 and described in Table 3-10.

This is a read-only register. All writes are ignored.

The default values of the control fields are determined by the device reset configuration word stored in the TI-OTP region of flash bank 0.

Figure 3-11. LOGICPDPWRSTAT3 Register

31	30	29	28	27	26	25	24
RESERVED							LOGIC_IN_ TRANS3
R-0h							R-X
23	22	21	20	19	18	17	16
RESERVED							MEM_IN_ TRANS3
R-0h							R-X
15	14	13	12	11	10	9	8
RESERVED							DOMAIN_ON3
R-0h							R-X
7	6	5	4	3	2	1	0
RESERVED						LOGICPDPWR_STAT3	
R-0h						R-X	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 3-10. LOGICPDPWRSTAT3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reads return 0. Writes have no effect.
24	LOGIC_IN_TRANS3	R	X	Logic in transition status for power domain PD5. Read in User and Privileged Mode. 0h = Logic in power domain PD5 is in the steady Active or OFF state. 1h = Logic in power domain PD5 is in the process of power-down/up.
23-17	RESERVED	R	0h	Reads return 0. Writes have no effect.
16	MEM_IN_TRANS3	R	X	Memory in transition status for power domain PD5. Read in User and Privileged Mode. 0h = Memory in power domain PD5 is in the steady Active or OFF state. 1h = Memory in power domain PD5 is in the process of power-down/up.
15-9	RESERVED	R	0h	Reads return 0. Writes have no effect.
8	DOMAIN_ON3	R	X	Current state of power domain PD5. The default value of this field is controlled by the device reset configuration word in the TI-OTP region of flash bank 0. Read in User and Privileged Mode. 0h = Power domain PD5 is in the OFF state. 1h = Power domain PD5 is in the Active state.
7-2	RESERVED	R	0h	Reads return 0. Writes have no effect.
1-0	LOGICPDPWR_STAT3	R	X	Logic power domain PD5 power state. The default value of this field is controlled by the device reset configuration word in the TI-OTP region of flash bank 0. Read in User and Privileged Mode. 0h = Logic power domain PD5 is switched OFF. 1h = Logic power domain PD5 is in Idle state. 2h = Reserved 3h = Logic power domain PD5 is in Active state.

3.4.10 MEMPDPWRSTAT0 Register (Offset = 80h) [reset = X]

MEMPDPWRSTAT0 is shown in Figure 3-12 and described in Table 3-11.

This is a read-only register. All writes are ignored.

The default values of the control fields are determined by the device reset configuration word stored in the TI-OTP region of flash bank 0.

Figure 3-12. MEMPDPWRSTAT0 Register

31	30	29	28	27	26	25	24
RESERVED							LOGIC_IN_TRANS0
R-0h							R-X
23	22	21	20	19	18	17	16
RESERVED							MEM_IN_TRANS0
R-0h							R-X
15	14	13	12	11	10	9	8
RESERVED							DOMAIN_ON0
R-0h							R-X
7	6	5	4	3	2	1	0
RESERVED						MEMPDPWR_STAT0	
R-0h						R-X	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 3-11. MEMPDPWRSTAT0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reads return 0. Writes have no effect.
24	LOGIC_IN_TRANS0	R	X	Logic in transition status for power domain RAM_PD1. This power domain only contains SRAM macros. However, an SRAM macro also has some digital logic controlled by the PSCON. Therefore, a memory power domain also contains a logic status indicator. Read in User and Privileged Mode. 0h = Logic in power domain RAM_PD1 is in the steady Active or OFF state. 1h = Logic in power domain RAM_PD1 is in the process of power-down/up.
23-17	RESERVED	R	0h	Reads return 0. Writes have no effect.
16	MEM_IN_TRANS0	R	X	Memory in transition status for power domain RAM_PD1. Read in User and Privileged Mode. 0h = Memory in power domain RAM_PD1 is in the steady Active or OFF state. 1h = Memory in power domain RAM_PD1 is in the process of power-down/up.
15-9	RESERVED	R	0h	Reads return 0. Writes have no effect.
8	DOMAIN_ON0	R	X	Current state of power domain RAM_PD1. The default value of this field is controlled by the device reset configuration word in the TI-OTP region of flash bank 0. Read in User and Privileged Mode. 0h = Power domain RAM_PD1 is in the OFF state. 1h = Power domain RAM_PD1 is in the Active state.
7-2	RESERVED	R	0h	Reads return 0. Writes have no effect.

Table 3-11. MEMPDPWRSTAT0 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1-0	MEMPDPWR_STAT0	R	X	Memory power domain RAM_PD1 power state. The default value of this field is controlled by the device reset configuration word in the TI-OTP region of flash bank 0. Read in User and Privileged Mode. 0h = Memory power domain RAM_PD1 is switched OFF. 1h = Reserved 2h = Reserved 3h = Logic power domain RAM_PD1 is in Active state.

3.4.11 MEMPDPWRSTAT1 Register (Offset = 84h) [reset = X]

MEMPDPWRSTAT1 is shown in Figure 3-13 and described in Table 3-12.

This is a read-only register. All writes are ignored.

The default values of the control fields are determined by the device reset configuration word stored in the TI-OTP region of flash bank 0.

Figure 3-13. MEMPDPWRSTAT1 Register

31	30	29	28	27	26	25	24
RESERVED							LOGIC_IN_TRANS1
R-0h							R-X

23	22	21	20	19	18	17	16
RESERVED							MEM_IN_TRANS1
R-0h							R-X

15	14	13	12	11	10	9	8
RESERVED							DOMAIN_ON1
R-0h							R-X

7	6	5	4	3	2	1	0
RESERVED						MEMPDPWR_STAT1	
R-0h						R-X	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 3-12. MEMPDPWRSTAT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reads return 0. Writes have no effect.
24	LOGIC_IN_TRANS1	R	X	Logic in transition status for power domain RAM_PD2. This power domain only contains SRAM macros. However, an SRAM macro also has some digital logic controlled by the PSCON. Therefore, a memory power domain also contains a logic status indicator. Read in User and Privileged Mode. 0h = Logic in power domain RAM_PD2 is in the steady Active or OFF state. 1h = Logic in power domain RAM_PD2 is in the process of power-down/up.
23-17	RESERVED	R	0h	Reads return 0. Writes have no effect.
16	MEM_IN_TRANS1	R	X	Memory in transition status for power domain RAM_PD2. Read in User and Privileged Mode. 0h = Memory in power domain RAM_PD2 is in the steady Active or OFF state. 1h = Memory in power domain RAM_PD2 is in the process of power-down/up.
15-9	RESERVED	R	0h	Reads return 0. Writes have no effect.
8	DOMAIN_ON1	R	X	Current state of power domain RAM_PD2. The default value of this field is controlled by the device reset configuration word in the TI-OTP region of flash bank 0. Read in User and Privileged Mode. 0h = Power domain RAM_PD2 is in the OFF state. 1h = Power domain RAM_PD2 is in the Active state.
7-2	RESERVED	R	0h	Reads return 0. Writes have no effect.

Table 3-12. MEMPDPWRSTAT1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1-0	MEMPDPWR_STAT1	R	X	Memory power domain RAM_PD2power state. The default value of this field is controlled by the device reset configuration word in the TI-OTP region of flash bank 0. Read in User and Privileged Mode. 0h = Memory power domain RAM_PD2 is switched OFF. 1h = Reserved 2h = Reserved 3h = Logic power domain RAM_PD2 is in Active state.

3.4.12 GLOBALCTRL1 Register (Offset = A0h) [reset = 0h]

GLOBALCTRL1 is shown in Figure 3-14 and described in Table 3-13.

Figure 3-14. GLOBALCTRL1 Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							PMCTRL_ PWRDN
R-0h							R/WP-0h
7	6	5	4	3	2	1	0
RESERVED							AUTO_CLK_ WAKE_ENA
R-0h							R/WP-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 3-13. GLOBALCTRL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Reads return 0. Writes have no effect.
8	PMCTRL_PWRDN	R/WP	0h	PMC/PSCON Power Down. Read in User and Privileged Mode returns current value of PMCTRL PWRDN. Write in Privileged mode only. 0h = Enable the clock to pmctrl_wakeup block. 1h = Disable the clock to pmctrl_wakeup block, which contains PMC and all PSCONs.
7-1	RESERVED	R	0h	Reads return 0. Writes have no effect.
0	AUTO_CLK_WAKE_ENA	R/WP	0h	Automatic Clock Enable on Wake Up. Read in User and Privileged Mode returns current value of AUTO CLK WAKE ENA. Write in Privileged mode only. 0h = Disable automatic clock wake up. The application must enable clocks by clearing the correct bit in the PDCLK_DIS register. 1h = Enable automatic clock wake up when a power domain transitions to Active state.

3.4.13 GLOBALSTAT Register (Offset = A8h) [reset = 1h]

GLOBALSTAT is shown in Figure 3-15 and described in Table 3-14.

Figure 3-15. GLOBALSTAT Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="7">RESERVED</td><td>PMCTRL_IDLE</td></tr>
<tr><td colspan="7">R-0h</td><td>R-1h</td></tr>
</table>

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 3-14. GLOBALSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reads return 0. Writes have no effect.
0	PMCTRL_IDLE	R	1h	State of PMC and all PSCONs. The PMM captures the status of PMC and PSCONs as they do not have a register interface to the host CPU. 0h = PMC and PSCONs for all power domains are in the process of generating power state transition control sequence for logic and/or SRAM. 1h = PMC and PSCONs for all power domains have completed generating power state transition control sequence triggered by PMC input control signals.

3.4.14 PRCKEYREG Register (Offset = ACh) [reset = 0h]

PRCKEYREG is shown in Figure 3-16 and described in Table 3-15.

Figure 3-16. PRCKEYREG Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																												MKEY			
R-0h																												R/WP-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 3-15. PRCKEYREG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reads return 0. Writes have no effect.
3-0	MKEY	R/WP	0h	Diagnostic PSCON Mode Key. The mode key is applied to all individual PSCON compare units. Read in User and Privileged mode returns the current value of MKEY. Write in Privileged mode only. 0h = Lock Step mode 6h = Self-test mode 9h = Error Forcing mode Fh = Self-test Error Forcing Mode

3.4.15 LPDDCSTAT1 Register (Offset = B0h) [reset = 0h]

LPDDCSTAT1 is shown in Figure 3-17 and described in Table 3-16.

Figure 3-17. LPDDCSTAT1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED												LCMPE_3_0			
R-0h												R/W1CP-0h			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												LSTC_3_0			
R-0h												R-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 3-16. LPDDCSTAT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0h	Reads return 0. Writes have no effect.
19-16	LCMPE_3_0	R/W1CP	0h	Logic Power Domain Compare Error. Each of these bits corresponds to a logic power domain: Bit 3 for PD5 Bit 2 for PD4 Bit 1 for PD3 Bit 0 for PD2 Read in User and Privileged Mode. Write in Privileged mode only. 0h (W) = Writing 0 has no effect. 0h (R) = PSCON signals are identical. 1h (W) = Clears the corresponding LCMPE[n] bit, if set. 1h (R) = PSCON signal compare mismatch identified.
15-4	RESERVED	R	0h	Reads return 0. Writes have no effect.
3-0	LSTC_3_0	R	0h	Logic Power Domain Self-test Complete. Each of these bits corresponds to a logic power domain: Bit 3 for PD5 Bit 2 for PD4 Bit 1 for PD3 Bit 0 for PD2 Read in User and Privileged Mode. Writes have no effect. 0h = Self-test is ongoing if self-test mode is entered. 1h = Self-test is complete.

3.4.16 *LPDDCSTAT2 Register (Offset = B4h) [reset = 0h]*

LPDDCSTAT2 is shown in Figure 3-18 and described in Table 3-17.

Figure 3-18. LPDDCSTAT2 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED												LSTET_3_0			
R-0h												R-0h			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RESERVED												LSTE_3_0			
R-0h												R-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 3-17. LPDDCSTAT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	RESERVED	R	0h	Reads return 0. Writes have no effect.
19-16	LSTET_3_0	R	0h	Logic Power Domain Self-test Error Type. Each of these bits corresponds to a logic power domain: Bit 3 for PD5 Bit 2 for PD4 Bit 1 for PD3 Bit 0 for PD2 Read in User and Privileged Mode. Writes have no effect. 0h = Self-test failed during compare match test. 1h = Self-test failed during compare mismatch test.
15-4	RESERVED	R	0h	Reads return 0. Writes have no effect.
3-0	LSTE_3_0	R	0h	Logic Power Domain Self-test Error. Each of these bits corresponds to a logic power domain: Bit 3 for PD5 Bit 2 for PD4 Bit 1 for PD3 Bit 0 for PD2 Read in User and Privileged Mode. Writes have no effect. 0h = Self-test passed. 1h = Self-test failed.

3.4.17 MPDDCSTAT1 Register (Offset = B8h) [reset = 0h]

MPDDCSTAT1 is shown in Figure 3-19 and described in Table 3-18.

This register shows the interrupt status (before enabling) and allows setting of the interrupt status.

Figure 3-19. MPDDCSTAT1 Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED						MCMPE_1_0	
R-0h						R/W1CP-0h	
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED						MSTC_1_0	
R-0h						R-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 3-18. MPDDCSTAT1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R	0h	Reads return 0. Writes have no effect.
17-16	MCMPE_1_0	R/W1CP	0h	Memory Power Domain Compare Error. Each of these bits corresponds to a memory power domain: Bit 1 for RAM_PD2 Bit 0 for RAM_PD1 Read in User and Privileged Mode. Write in Privileged mode only. 0h (W) = Writing 0 has no effect. 0h (R) = PSCON signals are identical. 1h (W) = Clears the corresponding MCMPE[n] bit, if set. 1h (R) = PSCON signal compare mismatch identified.
15-2	RESERVED	R	0h	Reads return 0. Writes have no effect.
1-0	MSTC_1_0	R	0h	Memory Power Domain Self-test Complete. Each of these bits corresponds to a memory power domain: Bit 1 for RAM_PD2 Bit 0 for RAM_PD1 Read in User and Privileged Mode. Writes have no effect. 0h = Self-test is ongoing if self-test mode is entered. 1h = Self-test is complete.

3.4.18 MPDDCSTAT2 Register (Offset = BCh) [reset = 0h]

MPDDCSTAT2 is shown in Figure 3-20 and described in Table 3-19.

Figure 3-20. MPDDCSTAT2 Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED						MSTET_1_0	
R-0h						R-0h	
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED						MSTE_1_0	
R-0h						R-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 3-19. MPDDCSTAT2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R	0h	Reads return 0. Writes have no effect.
17-16	MSTET_1_0	R	0h	Memory Power Domain Self-test Error Type. Each of these bits corresponds to a memory power domain: Bit 1 for RAM_PD2 Bit 0 for RAM_PD1 Read in User and Privileged Mode. Writes have no effect. 0h = Self-test failed during compare match test. 1h = Self-test failed during compare mismatch test.
15-2	RESERVED	R	0h	Reads return 0. Writes have no effect.
1-0	MSTE_1_0	R	0h	Memory Power Domain Self-test Error. Each of these bits corresponds to a memory power domain: Bit 1 for RAM_PD2 Bit 0 for RAM_PD1 Read in User and Privileged Mode. Writes have no effect. 0h = Self-test passed. 1h = Self-test failed.

3.4.19 ISODIAGSTAT Register (Offset = C0h) [reset = 0h]

ISODIAGSTAT is shown in Figure 3-21 and described in Table 3-20.

There is an ISO_DIAG cell implemented separately for each logic power domain. This is a special tie-off cell that reads a value of 1 when the logic power domain is powered up. This cell has an isolation value of 0. That is, when this logic power domain is turned off, this cell will read 0. The ISO_DIAG statuses for each logic power domain is reflected in the ISODIAGSTAT register. The application can poll this diagnostic register to make sure that a domain that has been commanded to turn off has actually received the command.

Figure 3-21. ISODIAGSTAT Register

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												ISO_DIAG_3_0			
R-0h												R-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 3-20. ISODIAGSTAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reads return 0. Writes have no effect.
3-0	ISO_DIAG_3_0	R	0h	Isolation Diagnostic. Each of these bits corresponds to a logic power domain: Bit 3 for PD5 Bit 2 for PD4 Bit 1 for PD3 Bit 0 for PD2 Read in User and Privileged Mode. Writes have no effect. 0h = Isolation is enabled for corresponding power domain. 1h = Isolation is disabled for corresponding power domain.

I/O Multiplexing and Control Module (IOMM)

This chapter describes the I/O Multiplexing and Control Module (IOMM).

Topic **Page**

4.1 Overview 240
4.2 Main Features of I/O Multiplexing Module (IOMM) 240
4.3 Control of Multiplexed Outputs 240
4.4 Control of Multiplexed Inputs 241
4.5 Control of Special Multiplexed Options 242
4.6 Safety Features 251
4.7 IOMM Registers 252
4.8 Output Multiplexing and Control 265

4.1 Overview

This chapter describes the overall features of the module that controls the I/O multiplexing on the device. The mapping of control registers to multiplexing options is specified in Section 4.8.

4.2 Main Features of I/O Multiplexing Module (IOMM)

The IOMM contains memory-mapped registers (MMR) that control device-specific multiplexed functions. The safety and diagnostic features of the IOMM are:

- Kicker mechanism to protect the MMRs from accidental writes
- Error indication for access violations

4.3 Control of Multiplexed Outputs

The signal multiplexing controlled by each memory-mapped control register (PINMMRn) is described in Table 4-21. Each byte in the PINMMRs control the functionality output on a single terminal. Consider the following example for the PINMMR10 control register.

Figure 4-1. PINMMR10 Control Register, Address = FFFF EB38h

31		26	25	24
Reserved			EMIF DATA[2]	EMIF DATA[2]
RWP-0			R/WP-0	R/WP-1
23		18	17	16
Reserved		N2HET2[7]	Reserved	EMIF nCS[0]
RWP-0		R/WP-0	R/WP-0	R/WP-1
15		10	9	8
Reserved			EMIF DATA[3]	EMIF DATA[3]
RWP-0			R/WP-0	R/WP-1
7	3	2	1	0
Reserved		RMII_RX_ER	MII_RX_ER	AD1 EVT
RWP-0		R/WP-0	R/WP-0	R/WP-1

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

- Consider the multiplexing controlled by PINMMR10[23–16]. These bits control the multiplexing between the EMIF_nCS[0] and N2HET2[7] on the ball N17 of the 337BGA package for this device. The default function on the N17 ball is EMIF_nCS[0]. This is dictated by bit 16 of the PINMMR10 register being set.
- If the application wants to use N17 as an N2HET2[7] signal, then bit 16 of PINMMR10 must be cleared and bit 18 must be set.
- Each feature of the output function is determined by the function selected to be output on a terminal.

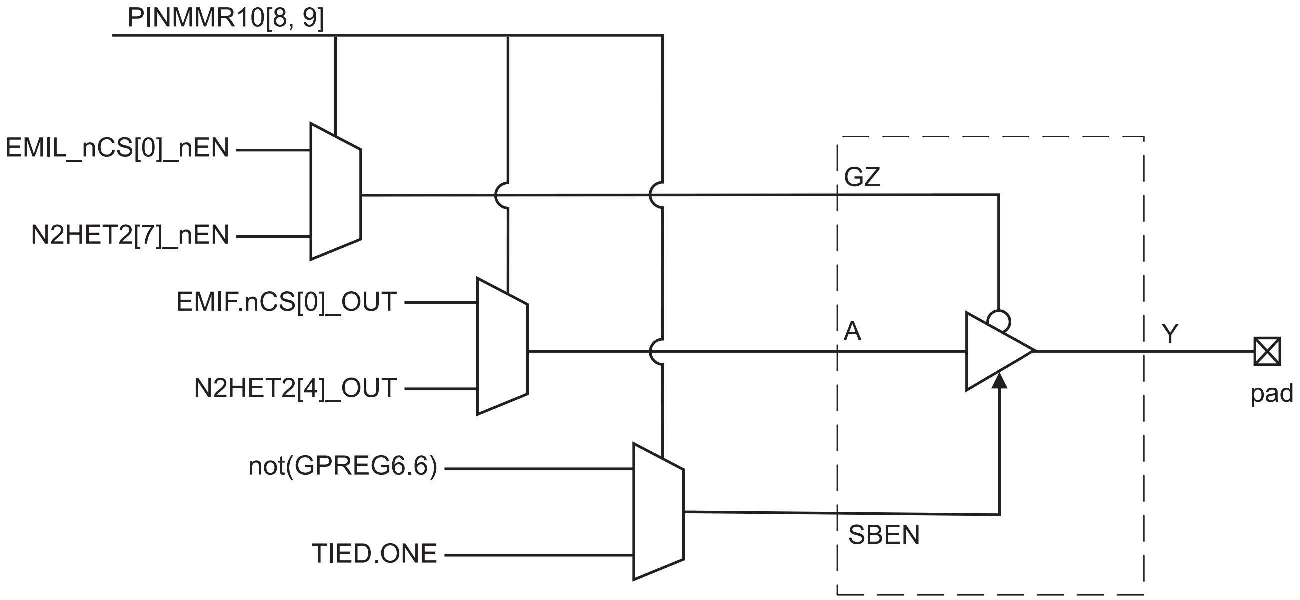
For example, the ball N17 on the 337BGA package is driven by an output buffer with an 8mA drive strength and which supports the adaptive impedance control mode for reduced electromagnetic emissions. This output buffer has the following signals: A (signal to be output), GZ (output enable), SBEN (Standard Buffer Enable) and LPM (Low Power Mode). Each of these signals is an output of a multiplexor which allows the selected function to control all available features of the output buffer.

- The PINMMR control registers are used to implement a one-hot encoding scheme for selecting the multiplexed function.
  - For example, for the N17 ball on the 337BGA package for this device at least one out of bit 16 or bit 18 must be set.
  - If the application clears both bits 16 and 18, then the default function, EMIF_nCS[0], will be selected for output on N17.
  - If the application sets both bits 16 and 18, then the default function will be selected for output on N17.
  - If the application sets one or more reserved bit(s) within the byte 23–16, then the default function

will be selected for output on N17.

Figure 4-2 shows the multiplexing between the output functions for the N17 ball. This terminal uses an 8mA low-EMI output buffer.

Figure 4-2. Output Multiplexing Example



4.4 Control of Multiplexed Inputs

The microcontrollers are available in two package options – 337-ball grid array (BGA) and 144-pin Quad Flat Pack (QFP). Input multiplexing is not required for the 337BGA package. For the 144QFP package, some signals have a multiplexor implemented on their input side. The selection between the two input paths is done by a combination of two control registers. These registers are described in Table 4-1.

Table 4-1. Input Multiplexing on 144QFP Parts

Signal	144QFP Dedicated Input Pin #	144QFP Multiplexed Input Pin #	A, Output Control for Muxed Pin	B, Input Control for Muxed Pin
SPI4SIMO	–	30	PINMMR5[9]	PINMMR23[16]
SPI4SOMI	–	31	PINMMR5[17]	PINMMR23[24]
SPI4CLK	–	25	PINMMR5[1]	PINMMR23[8]
SPI4NENA	–	23	PINMMR4[17]	PINMMR24[0]
SPI4NCS[0]	–	24	PINMMR4[25]	PINMMR24[8]
N2HET1[17]	–	130	PINMMR20[17]	PINMMR24[16]
N2HET1[19]	–	40	PINMMR8[9]	PINMMR24[24]
N2HET1[23]	–	96	PINMMR12[17]	PINMMR25[8]
N2HET1[25]	–	37	PINMMR7[9]	PINMMR25[16]
N2HET1[27]	–	4	PINMMR0[26]	PINMMR25[24]
N2HET1[29]	–	3	PINMMR0[18]	PINMMR26[0]
N2HET1[31]	–	54	PINMMR9[10]	PINMMR26[8]
GIOB[2]	142	55	PINMMR29[16]	PINMMR29[16]

For signals with a “–” in the column for dedicated input pin #, these signals do have a dedicated pad on the die. The default input for these signals comes from the dedicated pad. To be able to drive in the input from the multiplexed pins on the 144QFP package, the IOMM registers need to be configured.

Table 4-1 shows two controls for each of these signals: A and B.

A indicates the control register bit to be set to enable the corresponding signal to be output on the multiplexed pin.

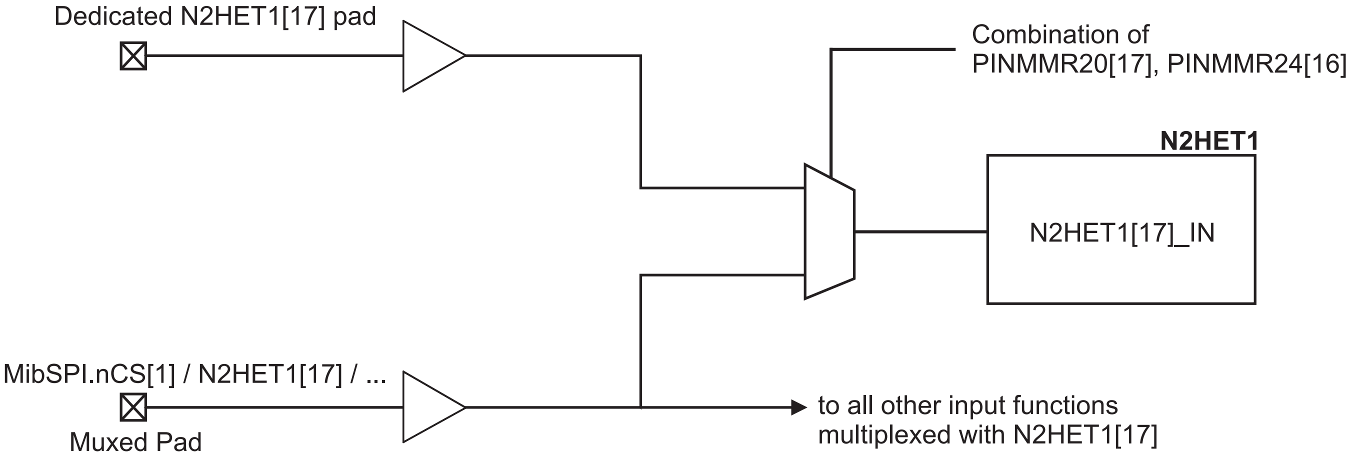
B is another control register bit that is used to select between the dedicated input terminal and the multiplexed pin.

- The input to a module comes from the **dedicated** pad when the condition **[not(A) or (A AND B)] = TRUE**.
- The input to a module comes from the **multiplexed** pin when the condition **[A and not(B)] = TRUE**.

NOTE: Inputs are broadcast to all multiplexed functions

The input signals are broadcast to all modules hooked up to a terminal. The application must ensure that modules that are not being used in the application do not react to a change on their input functions. For example, a GIO signal toggle can trigger an interrupt request, when the application actually is using the function multiplexed with this GIO signal.

Figure 4-3. Input Multiplexing Example


Dedicated N2HET1[17] pad
Combination of
PINMMR20[17], PINMMR24[16]
N2HET1
N2HET1[17]_IN
MibSPI.nCS[1] / N2HET1[17] / ...
Muxed Pad
to all other input functions
multiplexed with N2HET1[17]

4.5 Control of Special Multiplexed Options

Several of the PINMMR registers are used to control specific functions on this microcontroller.

4.5.1 Control of SDRAM clock (EMIF_CLK)

PINMMR29[8] is set by default. This blocks the EMIF SDRAM clock signal (EMIF_CLK) from being output from the microcontroller. If the EMIF is used to connect to an external SDRAM module, then the application must enable the SDRAM clock output by clearing the PINMMR29[8] bit.

4.5.2 Control for other EMIF Outputs

There are some EMIF address and control signals that are multiplexed with N2HET2 signals. For applications that require the use of these N2HET2 signals, it is inconvenient if the EMIF starts driving these address and control signals as output after reset is released and before the application can configure the I/O Multiplexing Module registers. Therefore, these EMIF signals are blocked from being output by default. This is done by using the bit 31 of the system module General Purpose Control Register (GPREG1). This bit is cleared (0) by default. In this condition, these EMIF/N2HET2 terminals are configured as inputs and pulled down. An application that requires the EMIF functionality must set the GPREG1[31] bit. This causes the EMIF address and control signals to then be output on the EMIF/N2HET2 terminals when the EMIF functionality is selected via the IOMM output multiplexing control registers.

4.5.3 Control for Fast Mode for Synchronous and Asynchronous EMIF

The EMIF on this microcontroller implements a "fast mode" for both the synchronous and asynchronous interfaces. There are some critical timing paths on read accesses to external asynchronous as well as synchronous memories. The fast mode allows the clock to the capture flops inside the EMIF module to be pushed in order to capture the incoming data correctly. The fast mode for asynchronous interface is enabled by setting PINMMR29[18] and the fast mode for the synchronous interface is enabled by setting PINMMR29[17].

4.5.4 Control of Ethernet Controller Mode

PINMMR29[24] is set by default. This bit is used to enable the Reduced Media Independent Interface of the Ethernet controller. If the application desires to use the Media Independent Interface of the Ethernet controller, then the PINMMR29[24] must be cleared.

4.5.5 Control of ADC Trigger Events

The microcontrollers contain two Analog-to-Digital Converter modules: ADC1 and ADC2. The ADC conversions can be started using a rising or falling or both edges as the trigger event. Both the ADC modules support up to eight event trigger inputs. There are two sets of these 8 inputs for each ADC. The option for each of these 8 inputs are controlled by registers in the I/O multiplexing module as shown in Table 4-2 and Table 4-3.

Table 4-2. ADC1 Trigger Event Selection

Group Source Select, G1SRC, G2SRC or EVSRC	Event #	Trigger Event Signal				
		PINMMR30[0] = 1 (default)	PINMMR30[0] = 0 and PINMMR30[1] = 1			
			Option A	Control for Option A	Option B	Control for Option B
000	1	AD1EVT	AD1EVT	—	AD1EVT	—
001	2	N2HET1[8]	N2HET2[5]	PINMMR30[8] = 1	ePWM_B	PINMMR30[8] = 0 and PINMMR30[9] = 1
010	3	N2HET1[10]	N2HET1[27]	—	N2HET1[27]	—
011	4	RTI Compare 0 Interrupt	RTI Compare 0 Interrupt	PINMMR30[16] = 1	ePWM_A1	PINMMR30[16] = 0 and PINMMR30[17] = 1
100	5	N2HET1[12]	N2HET1[17]	—	N2HET1[17]	—
101	6	N2HET1[14]	N2HET1[19]	PINMMR30[24] = 1	N2HET2[1]	PINMMR30[24] = 0 and PINMMR30[25] = 1
110	7	GIOB[0]	N2HET1[11]	PINMMR31[0] = 1	ePWM_A2	PINMMR31[0] = 0 and PINMMR31[1] = 1
111	8	GIOB[1]	N2HET2[13]	PINMMR31[8] = 1	ePWM_AB	PINMMR31[8] = 0 and PINMMR31[9] = 1

Table 4-3. ADC2 Trigger Event Selection

Group Source Select, G1SRC, G2SRC or EVSRC	Event #	Trigger Event Signal				
		PINMMR30[0] = 1 (default)	PINMMR30[0] = 0 and PINMMR30[1] = 1			
			Option A	Control for Option A	Option B	Control for Option B
000	1	AD2EVT	AD1EVT	—	AD1EVT	—

Table 4-3. ADC2 Trigger Event Selection (continued)

001	2	N2HET1[8]	N2HET2[5]	PINMMR31[16] = 1	ePWM_B	PINMMR31[16] = 0 and PINMMR31[17] = 1
010	3	N2HET1[10]	N2HET1[27]	—	N2HET1[27]	—
011	4	RTI Compare 0 Interrupt	RTI Compare 0 Interrupt	PINMMR31[24] = 1	ePWM_A1	PINMMR31[24] = 0 and PINMMR31[25] = 1
100	5	N2HET1[12]	N2HET1[17]	—	N2HET1[17]	—
101	6	N2HET1[14]	N2HET1[19]	PINMMR32[0] = 1	N2HET2[1]	PINMMR32[0] = 0 and PINMMR32[1] = 1
110	7	GIOB[0]	N2HET1[11]	PINMMR32[8] = 1	ePWM_A2	PINMMR32[8] = 0 and PINMMR32[9] = 1
111	8	GIOB[1]	N2HET2[13]	PINMMR32[16] = 1	ePWM_AB	PINMMR32[16] = 0 and PINMMR32[17] = 1

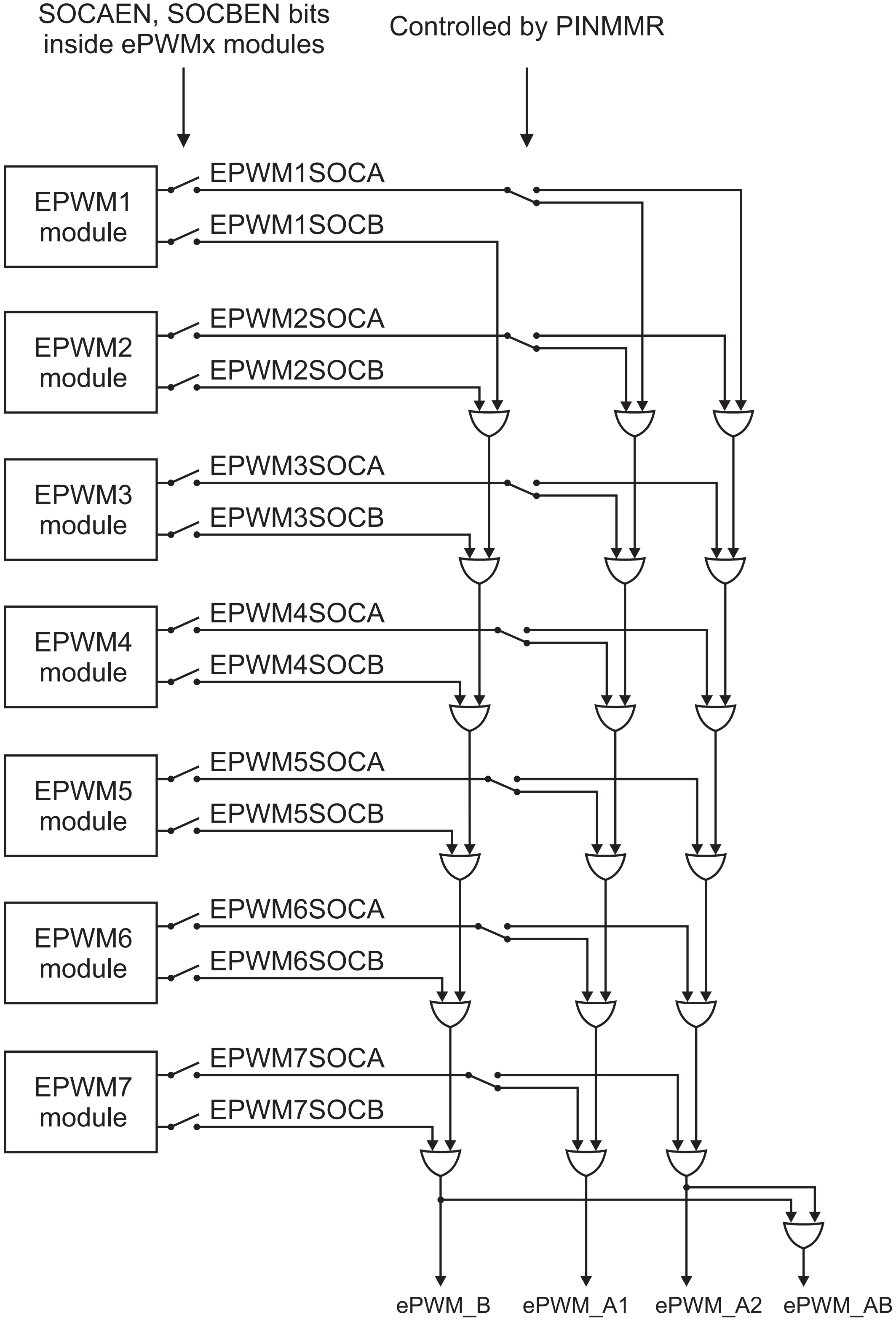
4.5.6 Control for ADC Event Trigger Signal Generation from ePWMx Modules

This microcontroller implements 7 ePWM modules. Each of these modules generate two outputs, SOCA (Start Of Conversion) and SOCB, for use in triggering the on-chip ADC modules. Registers from the I/O multiplexing module are used to control the logic for generation of the ePWM_A1, ePWM_A2, ePWM_AB, and ePWM_B signals from these ePWMx_SOCA and ePWMx_SOCB signals.

The logic equations for the 4 outputs from the combinational logic shown in Figure 4-4 are:

- B = SOC1B or SOC2B or SOC3B or SOC4B or SOC5B or SOC6B or SOC7B
- A1 = [SOC1A and not(SOC1A_SEL)]
  [SOC2A and not(SOC2A_SEL)]
  [SOC3A and not(SOC3A_SEL)]
  [SOC4A and not(SOC4A_SEL)]
  [SOC5A and not(SOC5A_SEL)]
  [SOC6A and not(SOC6A_SEL)]
  [SOC7A and not(SOC7A_SEL)]
- A2 = [SOC1A and SOC1A_SEL]
  [SOC2A and SOC2A_SEL]
  [SOC3A and SOC3A_SEL]
  [SOC4A and SOC4A_SEL]
  [SOC5A and SOC5A_SEL]
  [SOC6A and SOC6A_SEL]
  [SOC7A and SOC7A_SEL]
- AB = B or A2

Figure 4-4. ADC Trigger Event Signal Generation from ePWMx



The SOCxA_SEL signals used in the above logic equations are generated using registers in the I/O multiplexing module.

- PINMMR35[0] defines the value of SOC1A_SEL. This bit is set by default and can be cleared by the application.
- PINMMR35[8] defines the value of SOC2A_SEL. This bit is set by default and can be cleared by the application.
- PINMMR35[16] defines the value of SOC3A_SEL. This bit is set by default and can be cleared by the application.
- PINMMR35[24] defines the value of SOC4A_SEL. This bit is set by default and can be cleared by the application.
- PINMMR36[0] defines the value of SOC5A_SEL. This bit is set by default and can be cleared by the application.
- PINMMR36[8] defines the value of SOC6A_SEL. This bit is set by default and can be cleared by the application.
- PINMMR36[16] defines the value of SOC7A_SEL. This bit is set by default and can be cleared by the application.

4.5.7 Control for Generating Interrupt Upon External Fault Indication to N2HETx

The N2HET module on this microcontroller allows the application to selectively disable any PWM output form the N2HET module whenever a fault condition is indicated to the N2HET. This fault condition is input to the N2HET module via the N2HETx_PIN_nDIS input signal. It is important for the CPU to be notified with an interrupt whenever this fault condition is indicated to the N2HET module.

The PIN_nDIS signal for the N2HET1 module comes from the GIOA[5] terminal, and the application can enable the interrupt generation for whenever the GIOA[5] terminal is driven low.

The PIN_nDIS signal for the N2HET2 module comes from the MibSPI3_nCS[0] / AD2EVT terminal. These signals do not offer the capability of generating an interrupt when they are driven low. Therefore, the input from this terminal can optionally be connected to the GIOB[2] input. This connection is enabled by a register in the I/O multiplexing module. This is the PINMMR29[16].

4.5.8 Control for Enabling Clocks to ePWMx Modules

The VCLK4 domain is used as the clock for the ePWMx modules. This clock can be individually gated off when one or more of the ePWMx modules are not being used.

- PINMMR37[8] is set by default and enables the clock to the ePWM1. The application can clear this bit to disable the clock to the ePWM1.
- PINMMR37[16] is set by default and enables the clock to the ePWM2. The application can clear this bit to disable the clock to the ePWM2.
- PINMMR37[24] is set by default and enables the clock to the ePWM3. The application can clear this bit to disable the clock to the ePWM3.
- PINMMR38[0] is set by default and enables the clock to the ePWM4. The application can clear this bit to disable the clock to the ePWM4.
- PINMMR38[8] is set by default and enables the clock to the ePWM5. The application can clear this bit to disable the clock to the ePWM5.
- PINMMR38[16] is set by default and enables the clock to the ePWM6. The application can clear this bit to disable the clock to the ePWM6.
- PINMMR38[24] is set by default and enables the clock to the ePWM7. The application can clear this bit to disable the clock to the ePWM7.

4.5.9 Control for Enabling Clocks to eCAPx Modules

The VCLK4 domain is used as the clock for the eCAPx modules. This clock can be individually gated off when one or more of the eCAPx modules are not being used.

- PINMMR39[0] is set by default and enables the clock to the eCAP1. The application can clear this bit to disable the clock to the eCAP1.
- PINMMR39[8] is set by default and enables the clock to the eCAP2. The application can clear this bit to disable the clock to the eCAP2.
- PINMMR39[16] is set by default and enables the clock to the eCAP3. The application can clear this bit to disable the clock to the eCAP3.
- PINMMR39[24] is set by default and enables the clock to the eCAP4. The application can clear this bit to disable the clock to the eCAP4.
- PINMMR40[0] is set by default and enables the clock to the eCAP5. The application can clear this bit to disable the clock to the eCAP5.
- PINMMR40[8] is set by default and enables the clock to the eCAP6. The application can clear this bit to disable the clock to the eCAP6.

4.5.10 Control for Enabling Clocks to eQEPx Modules

The VCLK4 domain is used as the clock for the eQEPx modules. This clock can be individually gated off when one or more of the eQEPx modules are not being used.

- PINMMR40[16] is set by default and enables the clock to the eQEP1. The application can clear this bit to disable the clock to the eQEP1.
- PINMMR40[24] is set by default and enables the clock to the eQEP2. The application can clear this bit to disable the clock to the eQEP2.

4.5.11 Control for Synchronizing Time Bases for All ePWMx Modules

The ePWMx modules implement a mechanism that allows their time bases to be synchronized. This is done by using a signal called TBCLKSYNC, which is a common input to all the ePWMx modules. This TBCLKSYNC is generated by a register bit in the I/O multiplexing module. PINMMR37[1] is the TBCLKSYNC signal. This bit is clear (0) by default.

When TBCLKSYNC = 0, the time-base clock of all ePWMx modules is stopped. This is the default condition.

When TBCLKSYNC = 1, the time-base clocks of all ePWMx modules are started aligned to the rising edge of the TBCLKSYNC signal.

The correct procedure for enabling and synchronizing the time-base clocks of all the ePWMx modules is:

1. Enable the clocks to the desired individual ePWMx modules if they have been disabled
2. Set TBCLKSYNC = 0. This will stop the time-base clocks of any enabled ePWMx module.
3. Configure the time-base clock prescaler values and desired ePWM modes.
4. Set TBCLKSYNC = 1.

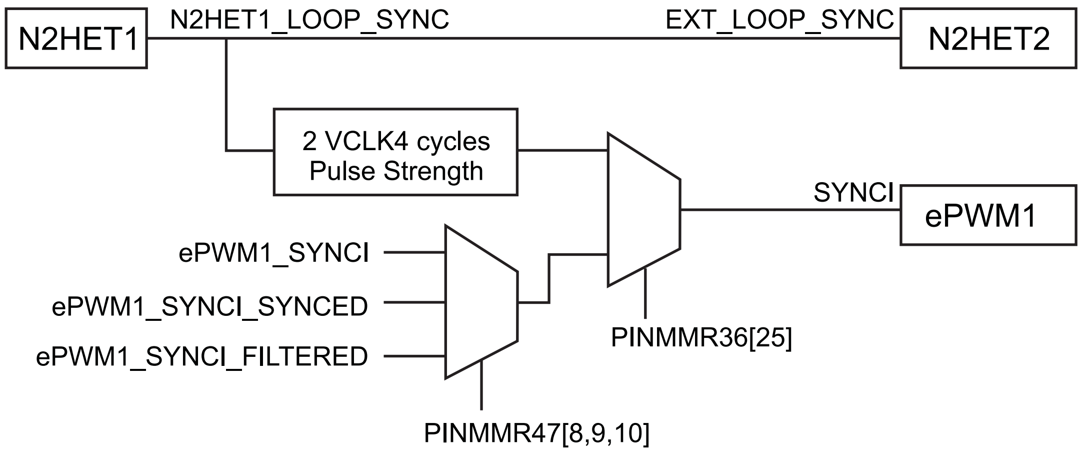
4.5.12 Control for Synchronizing all ePWMx Modules to N2HET1 Module Time-Base

Some applications require a synchronized time base for all PWM signals generated by the microcontroller. The N2HET1 module uses a time base that is created by configuring the high-resolution and loop-resolution prescalers in the N2HET1 module control registers. The N2HET1 module outputs the loop-resolution clock signal (N2HET1_LOOP_SYNC) so that other timer modules on the microcontroller can use it to synchronize their time bases to the N2HET1 loop-resolution clock.

There is a dedicated connection between the N2HET1 and N2HET2 modules, which allows the N2HET2 to use the N2HET1_LOOP_SYNC signal to synchronize its own time base to that of N2HET1.

The seven ePWMx modules can also optionally use the N2HET1_LOOP_SYNC for their time-base synchronization using a specially designed scheme.

Figure 4-5. Sychronizing ePWMx Modules to N2HET1 Time-Base



PINMMR36[25] is used to select between the ePWM1_SYNCI and the stretched N2HET1_LOOP_SYNC signals.

If PINMMR36[25] = 0, the SYNCI input to the ePWM1 comes from the ePWM1_SYNCI device input terminal. This is the default connection.

If PINMMR36[25] = 1, the SYNCI input to the ePWM1 comes from the pulse-stretched N2HET1_LOO_SYNC signal.

4.5.13 Control for Input Connections to ePWMx Modules

The ePWMx modules take the following signals as input:

- ePWM1_SYNCI: external time-base input to the ePWMx
- nTZ1 through nTZ6: trip-zone inputs to the ePWMx

Of the six trip-zone inputs, three are input from device terminals while the other three are connected to internal error events. Registers from the I/O multiplexing module are used to control various aspects of these input connections to the ePWMx modules.

Table 4-4. Controls for ePWMx Inputs

Input Signal	Control for Asynchronous Input (default connection)	Control for Double-VCLK4-Synchronized Input	Control for Double-VCLK4-Synchronized and 6-VCLK4-Filtered Input
nTZ1	PINMMR46[16] = 1	PINMMR46[16] = 0 and PINMMR46[17] = 1	PINMMR46[17:16] = 00 and PINMMR46[18] = 1
nTZ2	PINMMR46[24] = 1	PINMMR46[24] = 0 and PINMMR46[25] = 1	PINMMR46[25:24] = 00 and PINMMR46[26] = 1
nTZ3	PINMMR47[0] = 1	PINMMR47[0] = 0 and PINMMR47[1] = 1	PINMMR47[1:0] = 00 and PINMMR47[2] = 1
ePWM1_SYNCI	PINMMR47[8] = 1	PINMMR47[8] = 0 and PINMMR47[9] = 1	PINMMR47[9:8] = 00 and PINMMR47[10] = 1

Of the three internal error events for the trip-zone inputs, nTZ4 is connected to the eQEPx error output signal. There are two eQEP modules on this microcontroller, and registers in the I/O multiplexing module are used to allow a flexible scheme for the connection between the eQEPx error signal and the nTZ4 inputs of the ePWMx modules.

Table 4-5. Controls for eQEPx_ERROR Connection to ePWMx nTZ4 Inputs

ePWMx Module	Control for nTZ4 = not(eQEP1ERR or eQEP2ERR) (default connection)	Control for nTZ4 = not(eQEP1ERR)	Control for nTZ4 = not(eQEP2ERR)
ePWM1	PINMMR41[0] = 1	PINMMR41[0] = 0 and PINMMR41[1] = 1	PINMMR41[1:0] = 00 and PINMMR41[2] = 1
ePWM2	PINMMR41[8] = 1	PINMMR41[8] = 0 and PINMMR41[9] = 1	PINMMR41[9:8] = 00 and PINMMR41[10] = 1
ePWM3	PINMMR41[16] = 1	PINMMR41[16] = 0 and PINMMR41[17] = 1	PINMMR41[17:16] = 00 and PINMMR41[18] = 1
ePWM4	PINMMR41[24] = 1	PINMMR41[24] = 0 and PINMMR41[25] = 1	PINMMR41[25:24] = 00 and PINMMR41[26] = 1
ePWM5	PINMMR42[0] = 1	PINMMR42[0] = 0 and PINMMR42[1] = 1	PINMMR42[1:0] = 00 and PINMMR42[2] = 1
ePWM6	PINMMR42[8] = 1	PINMMR42[8] = 0 and PINMMR42[9] = 1	PINMMR42[9:8] = 00 and PINMMR42[10] = 1
ePWM7	PINMMR42[16] = 1	PINMMR42[16] = 0 and PINMMR42[17] = 1	PINMMR42[17:16] = 00 and PINMMR42[18] = 1

4.5.14 Control for Input Connections to eCAPx Modules

Each eCAPx module has a single input from the device terminals. This input can be connected to the eCAPx module in two ways:

1. Double-synchronized using VCLK4, or
2. Double-synchronized using VCLK4 and then filtered through a 6-VCLK4-cycle counter

Registers in the I/O multiplexing module are used to control these input connections for each eCAPx module.

Table 4-6. Controls for eCAPx Inputs

eCAPx Input	Control for Double-VCLK4-Synchronized Input (default connection)	Control for Double-VCLK4-Synchronized and 6-VCLK4-Filtered Input
eCAP1	PINMMR43[0] = 0	PINMMR43[0] = 0 and PINMMR43[1] = 1
eCAP2	PINMMR43[8] = 0	PINMMR43[8] = 0 and PINMMR43[9] = 1
eCAP3	PINMMR43[16] = 0	PINMMR43[16] = 0 and PINMMR43[17] = 1
eCAP4	PINMMR43[24] = 0	PINMMR43[24] = 0 and PINMMR43[25] = 1
eCAP5	PINMMR44[0] = 0	PINMMR44[0] = 0 and PINMMR44[1] = 1
eCAP6	PINMMR44[8] = 0	PINMMR44[8] = 0 and PINMMR44[9] = 1

4.5.15 Control for Input Connections to eQEPx Modules

Each eQEPx module has four inputs from the device terminals. These inputs can be connected to the eQEPx module in two ways:

1. Double-synchronized using VCLK4, or
2. Double-synchronized using VCLK4 and then filtered through a 6-VCLK4-cycle counter

Registers in the I/O multiplexing module are used to control these input connections for each eQEPx module.

Table 4-7. Controls for eQEPx Inputs

eQEPx Input	Control for Double-VCLK4-Synchronized Input (default connection)	Control for Double-VCLK4-Synchronized and 6-VCLK4-Filtered Input
eQEP1A	PINMMR44[16] = 1	PINMMR44[16] = 0 and PINMMR44[17] = 1
eQEP1B	PINMMR44[24] = 1	PINMMR44[24] = 0 and PINMMR44[25] = 1
eQEP1I	PINMMR45[0] = 1	PINMMR45[0] = 0 and PINMMR45[1] = 1
eQEP1S	PINMMR45[8] = 1	PINMMR45[8] = 0 and PINMMR45[9] = 1
eQEP2A	PINMMR45[16] = 1	PINMMR45[16] = 0 and PINMMR45[17] = 1
eQEP2B	PINMMR45[24] = 1	PINMMR45[24] = 0 and PINMMR45[25] = 1
eQEP2I	PINMMR46[0] = 1	PINMMR46[0] = 0 and PINMMR46[1] = 1
eQEP2S	PINMMR46[8] = 1	PINMMR46[8] = 0 and PINMMR46[9] = 1

4.6 Safety Features

The IOMM supports certain safety functions that are designed to prevent unintentional changes to the I/O multiplexing configuration. These are described in the following sections.

4.6.1 Locking Mechanism for Memory-Mapped Registers

The IOMM contains a mechanism to prevent any spurious writes from changing any of the PINMMR values. The PINMMRs are locked by default and after any system reset. None of the PINMMRs can be written under this condition. The application can read any of the IOMM registers regardless of the state of the locking mechanism.

- ***Enabling Write Access to the PINMMRs***

To enable write access to the PINMMRs, the CPU must write 0x83e70b13 to the kick0 register followed by a write of 0x95a4f1e0 to the kick1 register.

- ***Disabling Write Access to the PINMMRs***

It is recommended to disable write access to the PINMMRs once the I/O multiplexing configuration is completed. This can be done by:

- writing any other data value to either of the kick registers, or
- restarting the unlock sequence by writing 0x83e70b13 to the kick0 register

NOTE: No Error On Write to Locked PINMMRs

There is no error response on any write accesses to the PINMMRs when write access is disabled. None of the PINMMRs change state due to this write.

4.6.2 Error Conditions

The IOMM generates one error signal that is mapped to the Error Signaling Module’s Group 1, channel 37. This error signal is generated under either of the following two conditions:

- Address Error – occurs when there is a read or a write access to an un-implemented memory location within the IOMM register frame.
- Protection Error – occurs when the CPU writes to an IOMM register while not in a privileged mode of operation.

4.7 IOMM Registers

Table 4-8 lists the memory-mapped registers for the IOMM. All register offset addresses not listed in Table 4-8 should be considered as reserved locations and the register contents should not be modified. The address offset is specified from the base address of FFFF EA00h.

Table 4-8. IOMM Registers

Offset	Acronym	Register Name	Section
0h	REVISION_REG	Revision Register	Section 4.7.1
20h	ENDIAN_REG	Device Endianness Register	Section 4.7.2
38h	KICK_REG0	Kicker Register 0	Section 4.7.3
3Ch	KICK_REG1	Kicker Register 1	Section 4.7.4
E0h	ERR_RAW_STATUS_REG	Error Raw Status / Set Register	Section 4.7.5
E4h	ERR_ENABLED_STATUS_REG	Error Enabled Status / Clear Register	Section 4.7.6
E8h	ERR_ENABLE_REG	Error Signaling Enable Register	Section 4.7.7
ECh	ERR_ENABLE_CLR_REG	Error Signaling Enable Clear Register	Section 4.7.8
F4h	FAULT_ADDRESS_REG	Fault Address Register	Section 4.7.9
F8h	FAULT_STATUS_REG	Fault Status Register	Section 4.7.10
FCh	FAULT_CLEAR_REG	Fault Clear Register	Section 4.7.11
B10h to BCCh	PINMMR_0 to PINMMR_47	Pin Multiplexing Control Registers	Section 4.7.12

4.7.1 REVISION_REG Register (Offset = 0h) [reset = 4E840102h]

REVISION_REG is shown in Figure 4-6 and described in Table 4-9.

This is a read-only register that provides the revision information about the I/O Multiplexing Module (IOMM).

Figure 4-6. REVISION_REG Register

31	30	29	28	27	26	25	24
REV_SCHEME		RESERVED		REV_MODULE			
R-1h		R-0h		R-E84h			
23	22	21	20	19	18	17	16
REV_MODULE							
R-E84h							
15	14	13	12	11	10	9	8
REV_RTL					REV_MAJOR		
R-0h					R-1h		
7	6	5	4	3	2	1	0
REV_CUSTOM		REV_MINOR					
R-0h		R-2h					

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 4-9. REVISION_REG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	REV_SCHEME	R	1h	Revision Scheme, value = 01.
29-28	RESERVED	R	0h	
27-16	REV_MODULE	R	E84h	Module Id, value = E84h.
15-11	REV_RTL	R	0h	RTL Revision, value = 0.
10-8	REV_MAJOR	R	1h	Major Revision, value = 001.
7-6	REV_CUSTOM	R	0h	Custom Revision, value = 0.
5-0	REV_MINOR	R	2h	Minor Revision, value = 2h.

4.7.2 ENDIAN_REG Register (Offset = 20h) [reset = 0h]

ENDIAN_REG is shown in Figure 4-7 and described in Table 4-10.

This is a read-only register that reflects the state of the device endianness.

Figure 4-7. ENDIAN_REG Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							ENDIAN
R-0h							R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 4-10. ENDIAN_REG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	
0	ENDIAN	R	0h	Device endianness 0h = Device is configured in little-endian mode. 1h = Device is configured in big-endian mode.

4.7.3 KICK_REG0 Register (Offset = 38h) [reset = 0h]

KICK_REG0 is shown in Figure 4-8 and described in Table 4-11.

This register forms the first part of the unlock sequence for being able to update the I/O multiplexing control registers (PINMMRn).

Figure 4-8. KICK_REG0 Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="32">KICK0</td></tr>
<tr><td colspan="32">R/W-0h</td></tr>
</table>

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-11. KICK_REG0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	KICK0	R/W	0h	Kicker 0 Register. The value 83E7 0B13h must be written to KICK0 as part of the process to unlock the CPU write access to the PINMMRn registers.

4.7.4 KICK_REG1 Register (Offset = 3Ch) [reset = 0h]

KICK_REG1 is shown in Figure 4-9 and described in Table 4-12.

This register forms the second part of the unlock sequence for being able to update the I/O multiplexing control registers (PINMMRn).

Figure 4-9. KICK_REG1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KICK1																															
R/W-0h																															

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-12. KICK_REG1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	KICK1	R/W	0h	Kicker 1 Register. The value 95A4 F1E0h must be written to the KICK1 as part of the process to unlock the CPU write access to the PINMMRn registers.

4.7.5 ERR_RAW_STATUS_REG Register (Offset = E0h) [reset = 0h]

ERR_RAW_STATUS_REG is shown in Figure 4-10 and described in Table 4-13.

This register shows the status of the error conditions (before enabling) and allows setting the status. The IOMM module error signal is connected to the device's Error Signaling Module (ESM) group1 channel 37. The application can choose to generate an interrupt whenever this ESM channel flag gets set. This interrupt service routine can then read this Error Raw Status Register to determine the actual cause of the error condition. The Error Raw Status register is also writable by the application in order to test the ESM signaling and interrupt generation mechanism.

Figure 4-10. ERR_RAW_STATUS_REG Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-13. ERR_RAW_STATUS_REG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	
1	ADDR_ERR	R/W	0h	Addressing Error Status. An Addressing Error occurs when an unimplemented location inside the IOMM register frame is accessed. 0h (W) = Writing 0 has no effect. 0h (R) = Addressing Error has not occurred. 1h (W) = Addressing Error status is set. 1h (R) = Addressing Error has been detected.
0	PROT_ERR	R/W	0h	Protection Error Status. A Protection Error occurs when any control register inside the IOMM is written in the CPU's user mode of operation. 0h (W) = Writing 0 has no effect. 0h (R) = Protection Error has not occurred. 1h (W) = Protection Error status is set. 1h (R) = Protection Error has been detected.

4.7.6 ERR_ENABLED_STATUS_REG Register (Offset = E4h) [reset = 0h]

ERR_ENABLED_STATUS_REG is shown in Figure 4-11 and described in Table 4-14.

This register shows the status of the error conditions and allows clearing of the error status.

Figure 4-11. ERR_ENABLED_STATUS_REG Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 4-14. ERR_ENABLED_STATUS_REG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	
1	ENABLED_ADDR_ERR	R/W	0h	Addressing Error Signaling Enable and Status Clear 0h (W) = Writing 0 has no effect. 0h (R) = Addressing Error Signaling is disabled. 1h (W) = Addressing Error status is cleared. 1h (R) = Addressing Error Signaling is enabled.
0	ENABLED_PROT_ERR	R/W	0h	Protection Error Signaling Enable and Status Clear 0h (W) = Writing 0 has no effect. 0h (R) = Protection Error Signaling is disabled. 1h (W) = Protection Error status is cleared. 1h (R) = Protection Error Signaling is enabled.

4.7.7 ERR_ENABLE_REG Register (Offset = E8h) [reset = 0h]

ERR_ENABLE_REG is shown in Figure 4-12 and described in Table 4-15.

This register shows the interrupt enable status and allows enabling of the interrupts.

Figure 4-12. ERR_ENABLE_REG Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 4-15. ERR_ENABLE_REG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	
1	ADDR_ERR_EN	R/W	0h	Addressing Error Signaling Enable 0h (W) = Writing 0 has no effect. 0h (R) = Addressing Error signaling is disabled. 1h (W) = Addressing Error signaling is enabled. 1h (R) = Addressing Error signaling is enabled.
0	PROT_ERR_EN	R/W	0h	Protection Error Signaling Enable 0h (W) = Writing 0 has no effect. 0h (R) = Protection Error signaling is disabled. 1h (W) = Protection Error signaling is enabled. 1h (R) = Protection Error signaling is enabled.

4.7.8 ERR_ENABLE_CLR_REG Register (Offset = ECh) [reset = 0h]

ERR_ENABLE_CLR_REG is shown in Figure 4-13 and described in Table 4-16.

This register shows the error signaling enable status and allows disabling of the error signaling.

Figure 4-13. ERR_ENABLE_CLR_REG Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="6">RESERVED</td><td>ADDR_ERR_
EN_CLR</td><td>PROT_ERR_
EN_CLR</td></tr>
<tr><td colspan="6">R-0h</td><td>R/W-0h</td><td>R/W-0h</td></tr>
</table>

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 4-16. ERR_ENABLE_CLR_REG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-2	RESERVED	R	0h	
1	ADDR_ERR_EN_CLR	R/W	0h	Addressing Error Signaling Enable Clear 0h (W) = Writing 0 has no effect. 0h (R) = Addressing Error signaling is disabled. 1h (W) = Addressing Error signaling is disabled. 1h (R) = Addressing Error signaling is enabled.
0	PROT_ERR_EN_CLR	R/W	0h	Protection Error Signaling Enable Clear 0h (W) = Writing 0 has no effect. 0h (R) = Protection Error signaling is disabled. 1h (W) = Protection Error signaling is disabled. 1h (R) = Protection Error signaling is enabled.

4.7.9 FAULT_ADDRESS_REG Register (Offset = F4h) [reset = 0h]

FAULT_ADDRESS_REG is shown in Figure 4-14 and described in Table 4-17.

This register holds the address of the first fault transfer.

Figure 4-14. FAULT_ADDRESS_REG Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">RESERVED</td></tr>
<tr><td colspan="16">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="7">RESERVED</td><td colspan="9">FAULT_ADDR</td></tr>
<tr><td colspan="7">R-0h</td><td colspan="9">R/W-0h</td></tr>
</table>

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 4-17. FAULT_ADDRESS_REG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	
8-0	FAULT_ADDR	R/W	0h	Fault Address. The fault address offset in case of an address error or a protection error condition.

4.7.10 FAULT_STATUS_REG Register (Offset = F8h) [reset = 0h]

FAULT_STATUS_REG is shown in Figure 4-15 and described in Table 4-18.

This register holds the status and attributes of the first fault transfer.

Figure 4-15. FAULT_STATUS_REG Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED				FAULT_ID				FAULT_MSTID							
R-0h				R-0h				R-0h							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			FAULT_PRIVID				RESERVED			FAULT_TYPE					
R-0h			R-0h				R-0h			R-0h					

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-18. FAULT_STATUS_REG Register Field DTable 4-18escriptions

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	
27-24	FAULT_ID	R	0h	Faulting Transaction ID
23-16	FAULT_MSTID	R	0h	ID of Master that initiated the faulting transaction
15-13	RESERVED	R	0h	
12-9	FAULT_PRIVID	R	0h	Faulting Privilege ID
8-6	RESERVED	R	0h	
5-0	FAULT_TYPE	R	0h	Type of fault detected. 0h = No fault 1h = User execute fault 2h = User write fault 4h = User read fault 8h = Supervisor execute fault 10h = Supervisor write fault 20h = Supervisor read fault

4.7.11 FAULT_CLEAR_REG Register (Offset = FCh) [reset = 0h]

FAULT_CLEAR_REG is shown in Figure 4-16 and described in Table 4-19.

This register allows the application to clear the current fault so that another can be captured when 1 is written to this register.

Figure 4-16. FAULT_CLEAR_REG Register

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

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 4-19. FAULT_CLEAR_REG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	
0	FAULT_CLEAR	R/W	0h	Fault Clear 0h (W) = Writing 0 has no effect. 0h (R) = Current value of the FAULT_CLEAR bit is 0. 1h (W) = Writing a 1 clears the current fault. 1h (R) = Current value of the FAULT_CLEAR bit is 1.

4.7.12 PINMMR_0 to PINMMR_47 Register (Offset = B10h to BCCh) [reset = 1010101h]

PINMMR_0 to PINMMR_47 is shown in Figure 4-17 and described in Table 4-20.

These registers control the multiplexing of the functionality available on each pad as well as some special multiplexing schemes for specific functionalities on the microcontroller. There are 48 such registers PINMMR0 through PINMMR47. Each 8-bit field of a PINMMR register controls the functionality of a single ball/pin. The mapping between the PINMMRx control registers and the functionality selected on a given terminal is defined in .

Figure 4-17. PINMMR_0 to PINMMR_47 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PINMMRx_31_24								PINMMRx_23_16							
R/W-1h								R/W-1h							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PINMMRx_15_8								PINMMRx_7_0							
R/W-1h								R/W-1h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 4-20. PINMMR_0 to PINMMR_47 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	PINMMRx_31_24	R/W	1h	Each of these byte-fields control the functionality on a given ball/pin. Please refer to Output Multiplexing and Control table for a list of multiplexed signals. This list is sorted by the control register used.
23-16	PINMMRx_23_16	R/W	1h	Each of these byte-fields control the functionality on a given ball/pin. Please refer to Output Multiplexing and Control table for a list of multiplexed signals. This list is sorted by the control register used.
15-8	PINMMRx_15_8	R/W	1h	Each of these byte-fields control the functionality on a given ball/pin. Please refer to Output Multiplexing and Control table for a list of multiplexed signals. This list is sorted by the control register used.
7-0	PINMMRx_7_0	R/W	1h	Each of these byte-fields control the functionality on a given ball/pin. Please refer to Output Multiplexing and Control table for a list of multiplexed signals. This list is sorted by the control register used.

4.8 Output Multiplexing and Control

The device output terminals are shared between several different functions. The controls for these outputs are listed in Table 4-21. PINMMR35-PINMMR47 are used to control ePWMx, eCAPx, and eQEPx modules, see Section 4.5 and the device-specific data manual.

Table 4-21. Output Multiplexing and Control

Default Function	Selection Bit	Alternate Function 1	Selection Bit	Alternate Function 2	Selection Bit	Alternate Function 3	Selection Bit	Alternate Function 4	Selection Bit	Alternate Function 5	Selection Bit
GIOB[3]	PINMMR0[0]	USB2.RCV	PINMMR0[1]	USB_FUNC.RXDI	PINMMR0[2]	RESERVED	PINMMR0[3]	RESERVED	PINMMR0[4]		
GIOA[0]	PINMMR0[8]	USB2.VP	PINMMR0[9]	USB_FUNC.RXDPI	PINMMR0[10]	RESERVED	PINMMR0[11]	RESERVED	PINMMR0[12]	RESERVED	–
MIBSPI3NCS[3]	PINMMR0[16]	I2C_SCL	PINMMR0[17]	N2HET1[29]	PINMMR0[18]	nTZ1	PINMMR0[19]	RESERVED	PINMMR0[20]	RESERVED	–
MIBSPI3NCS[2]	PINMMR0[24]	I2C_SDA	PINMMR0[25]	N2HET1[27]	PINMMR0[26]	nTZ2	PINMMR0[27]	RESERVED	PINMMR0[28]	RESERVED	–
GIOA[1]	PINMMR1[0]	USB2.VM	PINMMR1[1]	USB_FUNC.RXDMI	PINMMR1[2]	RESERVED	PINMMR1[3]	RESERVED	PINMMR1[4]	RESERVED	–
N2HET1[11]	PINMMR1[8]	MIBSPI3NCS[4]	PINMMR1[9]	N2HET2[18]	PINMMR1[10]	USB2. OVERCURRENT	PINMMR1[11]	USB_FUNC. VBUSI	PINMMR1[12]	EPWM1SYNCO	PINMMR1[13]
GIOA[2]	PINMMR2[0]	USB2.TXDAT	PINMMR2[1]	USB_FUNC.TXDO	PINMMR2[2]	N2HET2[0]	PINMMR2[3]	EQEP2I	PINMMR2[4]	RESERVED	–
GIOA[3]	PINMMR2[16]	N2HET2[2]	PINMMR2[17]	RESERVED	PINMMR2[18]	RESERVED	PINMMR2[19]	RESERVED	PINMMR2[20]	RESERVED	–
GIOA[5]	PINMMR2[24]	EXTCLKIN	PINMMR2[25]	EPWM1A	PINMMR2[26]	RESERVED	PINMMR2[27]	RESERVED	PINMMR2[28]	RESERVED	–
N2HET1[22]	PINMMR3[8]	USB2.TXSE0	PINMMR3[9]	USB_FUNC.SE0O	PINMMR3[10]	RESERVED	PINMMR3[11]	RESERVED	PINMMR3[12]	RESERVED	–
GIOA[6]	PINMMR3[16]	N2HET2[4]	PINMMR3[17]	EPWM1B	PINMMR3[18]	RESERVED	PINMMR3[19]	RESERVED	PINMMR3[20]	RESERVED	–
GIOA[7]	PINMMR4[0]	N2HET2[6]	PINMMR4[1]	EPWM2A	PINMMR4[2]	RESERVED	PINMMR4[3]	RESERVED	PINMMR4[4]	RESERVED	–
N2HET1[01]	PINMMR4[16]	SPI4NENA	PINMMR4[17]	USB2.TXEN	PINMMR4[18]	USB_FUNC. PUENO	PINMMR4[19]	N2HET2[8]	PINMMR4[20]	EQEP2A	PINMMR4[21]
N2HET1[03]	PINMMR4[24]	SPI4NCS[0]	PINMMR4[25]	USB2.SPEED	PINMMR4[26]	USB_FUNC. PUENON	PINMMR4[27]	N2HET2[10]	PINMMR4[28]	EQEP2B	PINMMR4[29]
N2HET1[0]	PINMMR5[0]	SPI4CLK	PINMMR5[1]	EPWM2B	PINMMR5[2]	RESERVED	PINMMR5[3]	RESERVED	PINMMR5[4]	RESERVED	–
N2HET1[02]	PINMMR5[8]	SPI4SIMO	PINMMR5[9]	EPWM3A	PINMMR5[10]	RESERVED	PINMMR5[11]	RESERVED	PINMMR5[12]	RESERVED	–
N2HET1[05]	PINMMR5[16]	SPI4SOMI	PINMMR5[17]	N2HET2[12]	PINMMR5[18]	EPWM3B	PINMMR5[19]	RESERVED	PINMMR5[20]	RESERVED	–
N2HET1[07]	PINMMR6[0]	USB2.PORTPOWER	PINMMR6[1]	USB_FUNC.GZO	PINMMR6[2]	N2HET2[14]	PINMMR6[3]	EPWM7B	PINMMR6[4]	RESERVED	–
EMIF_DATA[11]	PINMMR6[8]	EMIF_DATA[11]	PINMMR6[9]	RESERVED	PINMMR6[10]	RESERVED	PINMMR6[11]	RESERVED	PINMMR6[12]	RESERVED	–
N2HET1[09]	PINMMR6[16]	N2HET2[16]	PINMMR6[17]	USB2.SUSPEND	PINMMR6[18]	USB_FUNC. SUSPENDO	PINMMR6[19]	EPWM7A	PINMMR6[20]	RESERVED	–
MIBSPI3NCS[1]	PINMMR7[8]	N2HET1[25]	PINMMR7[9]	MDCLK	PINMMR7[10]	RESERVED	PINMMR7[11]	RESERVED	PINMMR7[12]	RESERVED	–
N2HET1[06]	PINMMR7[16]	SCIRX	PINMMR7[17]	EPWM5A	PINMMR7[18]	RESERVED	PINMMR7[19]	RESERVED	PINMMR7[20]	RESERVED	–
N2HET1[13]	PINMMR8[0]	SCITX	PINMMR8[1]	EPWM5B	PINMMR8[2]	RESERVED	PINMMR8[3]	RESERVED	PINMMR8[4]	RESERVED	–
MIBSPI1NCS[2]	PINMMR8[8]	N2HET1[19]	PINMMR8[9]	MDIO	PINMMR8[10]	RESERVED	PINMMR8[11]	RESERVED	PINMMR8[12]	RESERVED	–
N2HET1[15]	PINMMR8[16]	MIBSPI1NCS[4]	PINMMR8[17]	ECAP1	PINMMR8[18]	RESERVED	PINMMR8[19]	RESERVED	PINMMR8[20]	RESERVED	–
MIBSPI3NENA	PINMMR9[8]	MIBSPI3NCS[5]	PINMMR9[9]	N2HET1[31]	PINMMR9[10]	EQEP1B	PINMMR9[11]	RESERVED	PINMMR9[12]	RESERVED	–
MIBSPI3NCS[0]	PINMMR9[16]	AD2EVT	PINMMR9[17]	GIOB[2]	PINMMR9[18]	EQEP1I	PINMMR9[19]	RESERVED	PINMMR9[20]	RESERVED	–
MIBSPI1NCS[3]	PINMMR9[24]	N2HET1[21]	PINMMR9[25]	RESERVED	PINMMR9[26]	RESERVED	PINMMR9[27]	RESERVED	PINMMR9[28]	RESERVED	–
AD1EVT	PINMMR10[0]	MII_RX_ER	PINMMR10[1]	RMII_RX_ER	PINMMR10[2]	RESERVED	PINMMR10[3]	RESERVED	PINMMR10[4]	RESERVED	–
EMIF_nCS[0]	PINMMR10[16]	RESERVED	PINMMR10[17]	N2HET2[7]	PINMMR10[18]	RESERVED	PINMMR10[19]	RESERVED	PINMMR10[20]	RESERVED	–
EMIF_nCS[3]	PINMMR11[0]	RESERVED	PINMMR11[1]	N2HET2[9]	PINMMR11[2]	RESERVED	PINMMR11[3]	RESERVED	PINMMR11[4]	RESERVED	–
N2HET1[24]	PINMMR11[24]	MIBSPI1NCS[5]	PINMMR11[25]	MII_RXD[0]	PINMMR11[26]	RMII_RXD[0]	PINMMR11[27]	RESERVED	PINMMR11[28]	RESERVED	–
N2HET1[26]	PINMMR12[0]	MII_RXD[1]	PINMMR12[1]	RMII_RXD[1]	PINMMR12[2]	RESERVED	PINMMR12[3]	RESERVED	PINMMR12[4]	RESERVED	–

Table 4-21. Output Multiplexing and Control (continued)

Default Function	Selection Bit	Alternate Function 1	Selection Bit	Alternate Function 2	Selection Bit	Alternate Function 3	Selection Bit	Alternate Function 4	Selection Bit	Alternate Function 5	Selection Bit
MIBSPI1NENA	PINMMR12[16]	N2HET1[23]	PINMMR12[17]	MII_RXD[2]	PINMMR12[18]	USB1.VP	PINMMR12[19]	ECAP4	PINMMR12[20]	RESERVED	–
MIBSPI5NENA	PINMMR12[24]	RESERVED	PINMMR12[25]	MII_RXD[3]	PINMMR12[26]	USB1.VM	PINMMR12[27]	MIBSPI5SOMI[1]	PINMMR12[28]	ECAP5	PINMMR12[29]
MIBSPI5SOMI[0]	PINMMR13[0]	RESERVED	PINMMR13[1]	MII_TXD[0]	PINMMR13[2]	RMII_TXD[0]	PINMMR13[3]	RESERVED	PINMMR13[4]	RESERVED	–
MIBSPI5SIMO[0]	PINMMR13[8]	RESERVED	PINMMR13[9]	MII_TXD[1]	PINMMR13[10]	RMII_TXD[1]	PINMMR13[11]	MIBSPI5SOMI[2]	PINMMR13[12]	RESERVED	–
MIBSPI5CLK	PINMMR13[16]	RESERVED	PINMMR13[17]	MII_TXEN	PINMMR13[18]	RMII_TXEN	PINMMR13[19]	RESERVED	PINMMR13[20]	RESERVED	–
MIBSPI1NCS[0]	PINMMR13[24]	MIBSPI1SOMI[1]	PINMMR13[25]	MII_TXD[2]	PINMMR13[26]	USB1.RCV	PINMMR13[27]	ECAP6	PINMMR13[28]	RESERVED	–
N2HET1[08]	PINMMR14[0]	MIBSPI1SIMO[1]	PINMMR14[1]	MII_TXD[3]	PINMMR14[2]	USB1. OVERCURRENT	PINMMR14[3]	RESERVED	PINMMR14[4]	RESERVED	–
N2HET1[28]	PINMMR14[8]	MII_RXCLK	PINMMR14[9]	RMII_REFCLK	PINMMR14[10]	MII_RX_AVCLK4	PINMMR14[11]	RESERVED	PINMMR14[12]	RESERVED	–
EMIF_nWE	PINMMR14[16]	EMIF_RNW	PINMMR14[17]	RESERVED	PINMMR14[18]	RESERVED	PINMMR14[19]	RESERVED	PINMMR14[20]	RESERVED	–
EMIF_BA[1]	PINMMR14[24]	N2HET2[5]	PINMMR14[25]	RESERVED	PINMMR14[26]	RESERVED	PINMMR14[27]	RESERVED	PINMMR14[28]	RESERVED	–
N2HET1[10]	PINMMR17[0]	MII_TX_CLK	PINMMR17[1]	USB1.TXEN	PINMMR17[2]	MII_TX_AVCLK4	PINMMR17[3]	nTZ3	PINMMR17[4]	RESERVED	–
N2HET1[12]	PINMMR17[16]	MII_CRS	PINMMR17[17]	RMII_CRS_DV	PINMMR17[18]	RESERVED	PINMMR17[19]	RESERVED	PINMMR17[20]	RESERVED	–
N2HET1[14]	PINMMR18[8]	USB1.TXSE0	PINMMR18[9]	RESERVED	PINMMR18[10]	RESERVED	PINMMR18[11]	RESERVED	PINMMR18[12]	RESERVED	–
GIOB[0]	PINMMR18[24]	USB1.TXDAT	PINMMR18[25]	RESERVED	PINMMR18[26]	RESERVED	PINMMR18[27]	RESERVED	PINMMR18[28]	RESERVED	–
N2HET1[30]	PINMMR19[8]	MII_RX_DV	PINMMR19[9]	USB1.SPEED	PINMMR19[10]	EQEP2S	PINMMR19[11]	RESERVED	PINMMR19[12]	RESERVED	–
MIBSPI1NCS[1]	PINMMR20[16]	N2HET1[17]	PINMMR20[17]	MII_COL	PINMMR20[18]	USB1.SUSPEND	PINMMR20[19]	EQEP1S	PINMMR20[20]	RESERVED	–
EMIF_ADDR[1]	PINMMR21[0]	N2HET2[3]	PINMMR21[1]	RESERVED	PINMMR21[2]	RESERVED	PINMMR21[3]	RESERVED	PINMMR21[4]	RESERVED	–
GIOB[1]	PINMMR21[8]	USB1.PORTPOWER	PINMMR21[9]	RESERVED	PINMMR21[10]	RESERVED	PINMMR21[11]	RESERVED	PINMMR21[12]	RESERVED	–
EMIF_ADDR[0]	PINMMR22[0]	N2HET2[1]	PINMMR22[1]	RESERVED	PINMMR22[2]	RESERVED	PINMMR22[3]	RESERVED	PINMMR22[4]	RESERVED	–
EMIF_ADDR[7]	PINMMR22[8]	RESERVED	PINMMR22[9]	N2HET2[13]	PINMMR22[10]	RESERVED	PINMMR22[11]	RESERVED	PINMMR22[12]	RESERVED	–
EMIF_ADDR[6]	PINMMR22[16]	RESERVED	PINMMR22[17]	N2HET2[11]	PINMMR22[18]	RESERVED	PINMMR22[19]	RESERVED	PINMMR22[20]	RESERVED	–
EMIF_ADDR[8]	PINMMR22[24]	RESERVED	PINMMR22[25]	N2HET2[15]	PINMMR22[26]	RESERVED	PINMMR22[27]	RESERVED	PINMMR22[28]	RESERVED	–
EMIF_ADDR[8]	PINMMR23[0]	RESERVED	PINMMR23[1]	N2HET2[15]	PINMMR23[2]	RESERVED	PINMMR23[3]	RESERVED	PINMMR23[4]	RESERVED	–
MIBSPI5NCS[0]	PINMMR27[0]	RESERVED	PINMMR27[1]	EPWM4A	PINMMR27[2]	RESERVED	PINMMR27[3]	RESERVED	PINMMR27[4]	RESERVED	–
SPI2NENA	PINMMR29[0]	SPI2NCS[1]	PINMMR29[1]	RESERVED	PINMMR29[2]	RESERVED	PINMMR29[3]	RESERVED	PINMMR29[4]	RESERVED	–
N2HET1[04]	PINMMR33[0]	EPWM4B	PINMMR33[1]	RESERVED	PINMMR33[2]	RESERVED	PINMMR33[3]	RESERVED	PINMMR33[4]	RESERVED	–
MIBSPI3SOMI[0]	PINMMR33[8]	AWM_EXT_ENA	PINMMR33[9]	ECAP2	PINMMR33[10]	RESERVED	PINMMR33[11]	RESERVED	PINMMR33[12]	RESERVED	–
MIBSPI3SIMO[0]	PINMMR33[16]	AWM_EXT_SEL[0]	PINMMR33[17]	ECAP3	PINMMR33[18]	RESERVED	PINMMR33[19]	RESERVED	PINMMR33[20]	RESERVED	–
MIBSPI3CLK	PINMMR33[24]	AWM_EXT_SEL[1]	PINMMR33[25]	EQEP1A	PINMMR33[26]	RESERVED	PINMMR33[27]	RESERVED	PINMMR33[28]	RESERVED	–
N2HET1[16]	PINMMR34[0]	EPWM1SYNCI	PINMMR34[1]	EPWM1SYNCO	PINMMR34[2]	RESERVED	PINMMR34[3]	RESERVED	PINMMR34[4]	RESERVED	–
N2HET1[18]	PINMMR34[8]	EPWM6A	PINMMR34[9]	RESERVED	PINMMR34[10]	RESERVED	PINMMR34[11]	RESERVED	PINMMR34[12]	RESERVED	–
N2HET1[20]	PINMMR34[16]	EPWM6B	PINMMR34[17]	RESERVED	PINMMR34[18]	RESERVED	PINMMR34[19]	RESERVED	PINMMR34[20]	RESERVED	–

Chapter 5

SPNU514C – March 2018

F021 Flash Module Controller (FMC)

The Flash electrically-erasable programmable read-only memory module is a type of nonvolatile memory that has fast read access times and is able to be reprogrammed in the field or in the application. This chapter describes the F021 Flash module controller (FMC).

Topic **Page**

5.1 Overview 268
5.2 Default Flash Configuration 269
5.3 SECDED 270
5.4 Memory Map 274
5.5 Power On, Power Off, and Reset Considerations 277
5.6 Emulation and SIL3 Diagnostic Modes 278
5.7 Control Registers 284

5.1 Overview

The F021 Flash is used to provide non-volatile memory for instruction execution or data storage. The Flash can be electrically programmed and erased many times to ease code development.

Refer to the following documents for support in how to initialize and use the on-chip Flash and its API:

- *Initialization of Hercules ARM Cortex-R4F Microcontrollers Application Report* (SPNA106)
- *F021 (Texas Instruments 65nm Flash) Flash API Reference Guide* (SPNU501)

5.1.1 Features

- Read, program and erase with a single 3.3 V supply voltage
- Supports error detection and correction
  - Single Error Correction and Double Error Detection (SECDED)
  - Error Correction Code (ECC) is evaluated in the CPU for the main Flash bank arrays and in the Flash Wrapper for the EEPROM emulation Flash banks
  - Address bits included in ECC calculation
- Provides different read modes to optimize performance and verify the integrity of Flash contents
- Provides built-in power mode control logic
- Integrated program/erase state machine
  - Simplifies software algorithms
  - Supports simultaneous read access on a bank while performing a write or erase operation on any one of the remaining banks
  - Suspend command allows read access to a sector being programmed/erased
  - Fast erase and program times (for details, see the device-specific data sheet)

For the actual size of the Flash memory for the device, see the device-specific data sheet.

5.1.2 Definition of Terms

Terms used in this document have the following meaning:

- ATCM: Port A tightly coupled memory
- BAGP (Bank Active Grace Period): Time (in HCLK cycles) from the most recent Flash access of a particular bank until that bank enters fallback power mode. This reduces power consumption by the Flash. However, it can also increase access time.
- bw: Normal data space bank data width of a Flash bank. The bw is 128 bits (144 bits including the error correction bits).
- bwe: EEPROM emulation bank is 128-bits wide (144 bits including the error correction bits).
- Charge pump: Voltage generators and associated control (logic, oscillator, and bandgap, for example).
- CSM: Program/erase command state machine
- Fallback power mode: The power mode (active, standby or sleep, depending on which mode is selected) into which a bank or the charge pump falls back each time the active grace period expires.
- Flash bank: A group of Flash sectors that share input/output buffers, data paths, sense amplifiers, and control logic.
- FEE - Flash EEPROM Emulation. Features on the FMC to support using a Flash type memory in place of an EEPROM Flash memory. EEPROM is erasable by the word while this Flash memory is only erasable by the sector. The FEE bank is accessible only through Bus 2 in a special address range and always resides in bank 7.
- Flash module: Flash banks, charge pump, and Flash wrapper.
- Flash wrapper: Power and mode control logic, data path, wait logic, and write/erase state machines.
- FMC: Flash Module Controller.
- Command: A sequence of coded instructions to Flash module to execute a certain task.

- FSM (Flash State Machine): State machine that parses and decodes FSM commands. It executes embedded algorithms and generates control signals to both Flash bank and charge pump during the actual program/erase operation.
- OTP (one-time programmable): A program-only-once Flash sector (cannot be erased)
- PAGP (Pump Active Grace Period): Time (in HCLK cycles) from when the last of the banks have entered fallback power mode until the pump enters a fallback power mode. This can reduce power consumption by the Flash; however, it can also increase access time.
- Pipeline mode: The mode in which Flash is read 128 bits (+ 16 bit ECC) at a time, providing higher throughput.
- Sector: A contiguous region of Flash memory which must be erased simultaneously.
- Wide_Word: The width of the data output from the Flash bank. This is 144-bits wide for main Flash and for the FEE bank.
- Standard read mode: The mode assumed when the pipeline mode is disabled. Physically, 128 (+ 16 bit ECC) is read at a time. However, only 32 bits of data is used while the other bits of data are discarded.
- Read Margin 1 mode: More stringent read mode designed for early detection of marginally erased bits.
- Read Margin 0 mode: More stringent read mode designed for early detection of marginally programmed bits.

5.1.3 F021 Flash Tools

Texas Instruments provides the following tools for F021 Flash:

- nowECC Generation Tool - to generate the Flash ECC from the Flash data.
- nowFLASH Programming Tool - to erase/program/verify the device Flash content through JTAG.
- Code Composer Studio - the development environment with integrated Flash programming capabilities.
- F021 Flash API Library - a set of software peripheral functions to program/erase the Flash module. Refer to *F021 Flash API Reference Guide* (SPNU501) for more information.

5.2 Default Flash Configuration

At power up, the Flash module state exhibits the following properties:

- Wait states are set to 1 data wait state and 0 address wait states
- Pipeline mode is disabled
- The Flash content is protected from modification
- Power modes are set to *Active* (no power savings)
- The boot code must initialize the wait states (including data wait states and address wait states) and the desired pipeline mode by initializing the FRDCNTL register to achieve the optimum system performance. This needs to be done before switching to the final device operating frequency. Refer to *Initialization of Hercules ARM Cortex-R4F Microcontrollers Application Report* (SPNA106) for more information.

5.3 SECDED

The Flash memory can be protected by Single Error Correction Double Error Detection (SECDED). The main program memory is protected by the SECDED circuit inside of the Cortex-R4 CPU. All OTP and the FEE memory (bank 7) is protected by SECDED logic in the Flash wrapper.

5.3.1 SECDED Initialization

Flash error detection and correction is not enabled at reset. To enable SECDED, error correction detection must be enabled in the Flash wrapper, the CPU event bus must be enabled and SECDED must be enabled within the CPU. Refer to *Initialization of Hercules ARM Cortex-R4F Microcontrollers Application Report* (SPNA106) for information on these steps.

The ECC values for all of the ATCM program memory space (Flash banks 0 through 6) must be programmed into the Flash before SECDED is enabled. This can be done by generating the correct values of the ECC with an external tool such as nowECC or may be generated by the programming tool. The Cortex-R4 CPU may generate speculative fetches to any location within the ATCM memory space. A speculative fetch to a location with invalid ECC, which is subsequently not used, will not create an abort, but will set the ESM flags for a correctable or uncorrectable error. An uncorrectable error will unconditionally cause the nERROR pin to toggle low. Therefore care must be taken to generate the correct ECC for the entire ATCM space including the holes between sections and any unused or blank Flash areas.

The Cortex-R4 CPU does not generate speculative fetches into the address space of bank 7, the EEPROM Emulation Flash. It is only necessary to initialize the ECC values of the locations which will be intentionally read by the CPU or other bus masters.

5.3.2 ECC Encoding

Nineteen address lines are also included in the ECC calculation. A failure of a single address line inside of the bank will be treated as an uncorrectable error. The ECC encoding is shown in Table 5-1. Bits 31:0 come from the word at the address ending in 0x0 or 0x8, Bits 63:31 come from the word at the address ending in 0x4 or 0xC.

Table 5-1. ECC Encoding for LE Devices

		82	81	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
		Participating Address Bits																		
ADDR_MSW_LSW	**ECC Bit**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**
0007F_00FFFF00_FF0000FF	7													x	x	x	x	x	x	x
7FF80_FF0000FF_FF0000FF	6	x	x	x	x	x	x	x	x	x	x	x	x							
07F80_FF00FF00_FF00FF00	5					x	x	x	x	x	x	x	x							
19F83_C0FCC0FC_C0FCC0FC	4			x	x			x	x	x	x	x	x						x	x
6A78D_38E338E3_38E338E3	3	x	x		x		x			x	x	x	x				x	x		x
2A9B5_A699A699_A699A699	2		x		x		x		x			x	x		x	x		x		x
0BAD1_15571557_15571557	1				x		x	x	x		x		x	x		x				x
554EA_B4D1B4D1_4B2E4B2E	0	x		x		x		x		x			x	x	x		x		x	

Participating Data Bits																																				
63	**62**	**61**	**60**	**59**	**58**	**57**	**56**	**55**	**54**	**53**	**52**	**51**	**50**	**49**	**48**	**47**	**46**	**45**	**44**	**43**	**42**	**41**	**40**	**39**	**38**	**37**	**36**	**35**	**34**	**33**	**32**	**31**	**30**	**29**	**28**	**27**
								x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x									x	x	x	x	x
x	x	x	x	x	x	x	x																	x	x	x	x	x	x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x									x	x	x	x	x	x	x	x									x	x	x	x	x
x	x							x	x	x	x	x	x			x	x							x	x	x	x	x	x			x	x			
		x	x	x				x	x	x				x	x			x	x	x				x	x	x				x	x			x	x	x
x		x			x	x		x			x	x			x	x		x			x	x		x			x	x			x	x		x		
			x		x		x		x		x		x	x	x				x		x		x		x		x		x	x	x				x	
x		x	x		x			x	x		x				x	x		x	x		x			x	x		x				x		x			x

Participating Data Bits																											Parity[(1)]	Check Bits[(2)]
26	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**		
x	x	x																	x	x	x	x	x	x	x	x	Even	ECC[7]
x	x	x																	x	x	x	x	x	x	x	x	Even	ECC[6]
x	x	x									x	x	x	x	x	x	x	x									Even	ECC[5]
			x	x	x	x	x	x			x	x							x	x	x	x	x	x			Even	ECC[4]
			x	x	x				x	x			x	x	x				x	x	x				x	x	Odd	ECC[3]
x	x		x			x	x			x	x		x			x	x		x			x	x			x	Odd	ECC[2]
x		x		x		x		x	x	x				x		x		x		x		x		x	x	x	Even	ECC[1]
	x	x			x		x	x	x			x			x		x	x			x		x	x	x		Even	ECC[0]

[(1)] For Odd parity, XOR a 1 to the row's XOR result. For even Parity, use the row's XOR result directly.

[(2)] Each ECC[x] bit represents the XOR of all the address and data bits marked with x in the same row.

5.3.3 Syndrome Table: Decode to Bit in Error

The syndrome is an 8-bit value that decodes to the bit in error. The bit in error can be a bit among the 64 data bits, the 19 address bits, or a bit among the 8 ECC check bits. A syndrome value of 00000000 indicates there is no error. Any other syndrome combinations not shown in the table are uncorrectable multi-bit error. Errors of three of more bits may escape detection. The syndrome decoding is shown in Table 5-2.

Table 5-2. Syndrome Table, Decode to Bit in Error

	Address Bit Error Position																		
	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03
	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
	0	0	1	1	0	0	1	1	1	1	1	1	0	0	0	0	0	1	1
	1	1	0	1	0	1	0	0	1	1	1	1	0	0	0	1	1	0	1
	0	1	0	1	0	1	0	1	0	0	1	1	0	1	1	0	1	0	1
	0	0	0	1	0	1	1	1	0	1	0	1	1	0	1	0	0	0	1
	1	0	1	0	1	0	1	0	1	0	0	1	1	1	0	1	0	1	0

Data Bit Error Position																																				
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1
1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	0	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	0	1	1	0	0	0
0	0	1	1	1	0	0	0	1	1	1	0	0	0	1	1	0	0	1	1	1	0	0	0	1	1	1	0	0	0	1	1	0	0	1	1	1
1	0	1	0	0	1	1	0	1	0	0	1	1	0	0	1	1	0	1	0	0	1	1	0	1	0	0	1	1	0	0	1	1	0	1	0	0
0	0	0	1	0	1	0	1	0	1	0	1	0	1	1	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	1	1	0	0	0	1	0
1	0	1	1	0	1	0	0	1	1	0	1	0	0	0	1	1	0	1	1	0	1	0	0	1	1	0	1	0	0	0	1	0	1	0	0	1

Data Bit Error Position																											ECC Error Bit								
26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	07	06	05	04	03	02	01	00	
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	Bit[7]
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	Bit[6]
1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	Bit[5]
0	0	0	1	1	1	1	1	1	0	0	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	1	0	0	0	0	Bit[4]
0	0	0	1	1	1	0	0	0	1	1	0	0	1	1	1	0	0	0	1	1	1	0	0	0	1	1	0	0	0	0	1	0	0	0	Bit[3]
1	1	0	1	0	0	1	1	0	0	1	1	0	1	0	0	1	1	0	1	0	0	1	1	0	0	1	0	0	0	0	0	1	0	0	Bit[2]
1	0	1	0	1	0	1	0	1	1	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1	1	1	0	0	0	0	0	0	1	0	Bit1]
0	1	1	0	0	1	0	1	1	1	0	0	1	0	0	1	0	1	1	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	1	Bit[0]

5.3.4 Syndrome Table: An Alternate Method

Table 5-3. Alternate Syndrome Table

Syndrome lsb: 3:0	Syndrome msb 7:4															
	0x	**1x**	**2x**	**3x**	**4x**	**5x**	**6x**	**7x**	**8x**	**9x**	**Ax**	**Bx**	**Cx**	**Dx**	**Ex**	**Fx**
x0	good	E04	E05	D	E06	D	D	D62	E07	D	D	D46	D	M	M	D
x1	E00	D	D	D14	D	A19	A17	D	D	A04	M	D	M	D	D	D30
x2	E01	D	D	M	D	D34	D56	D	D	D50	D40	D	M	D	D	M
x3	D	D18	D08	D	M	D	D	A15	A09	D	D	M	D	D02	D24	D
x4	E02	D	D	D15	D	D35	D57	D	D	D51	D41	D	M	D	D	D31
x5	D	D19	D09	D	M	D	D	D63	A08	D	D	D47	D	D03	D25	D
x6	D	D20	D10	D	M	D	D	A14	A07	D	D	M	D	D04	D26	D
x7	M	D	D	M	D	D36	D58	D	D	D52	D42	D	M	D	D	M
x8	E03	D	D	M	D	D37	D59	D	D	D53	D43	D	M	D	D	M
x9	D	D21	D11	D	A21	D	D	A13	A06	D	D	M	D	D05	D27	D
xA	D	D22	D12	D	D33	D	D	A12	D49	D	D	M	D	D06	D28	D
xB	D17	D	D	M	D	D38	D60	D	D	D54	D44	D	D01	D	D	M
xC	D	D23	D13	D	A20	D	D	A11	A05	D	D	M	D	D07	D29	D
xD	M	D	D	M	D	D39	D61	D	D	D55	D45	D	M	D	D	M
xE	D16	D	D	M	D	A18	A16	D	D	A03	M	D	D00	D	D	M
xF	D	M	M	D	D32	D	D	A10	D48	D	D	M	D	M	M	D

- E0x - Single-bit ECC error, correctable
- Dxx - Single-bit data error, correctable
- Axx - Single-bit address error, uncorrectable
- D - Double-bit error, uncorrectable
- M - Multi-bit errors, uncorrectable

5.4 Memory Map

The Flash module contains the program memory, which is mapped starting at location 0, and one Customer OTP sector and one TI OTP sector per bank. The Customer OTP sectors may be programmed by the customer, but cannot be erased. They are typically blank in new parts. The TI OTP sectors are used to contain manufacturing information. They may be read by the customer but can not be programmed or erased. The TI OTP sectors contain settings used by the Flash API to setup the Flash state machine for erase and program operations.

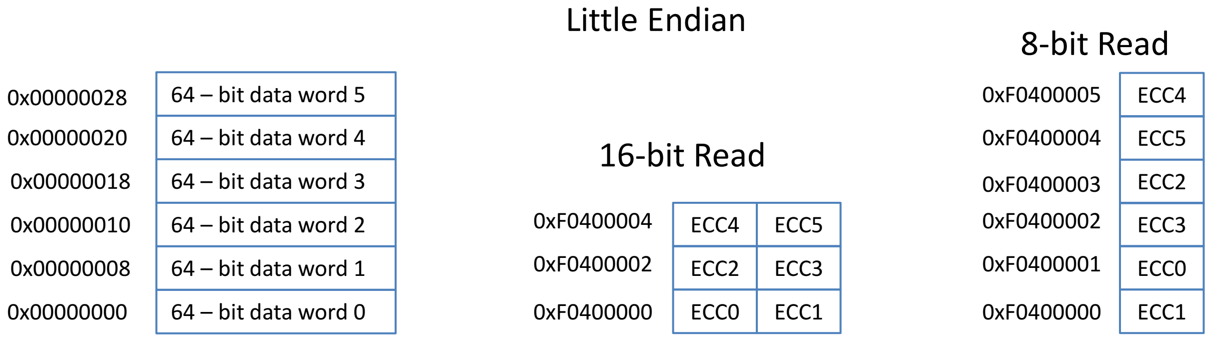
All of these OTP regions are memory-mapped to facilitate ease of access by the CPU. They are memory mapped to an offset starting at F000 0000h in the CPU’s memory map.

The RWAIT value is used to define the number of wait states for the program-memory Flash. The EWAIT value is used to define the number of wait states for the data Flash in bank 7. Bank 7 starting at offset F020 0000h is dedicated for data storages such as EEPROM Emulation.

5.4.1 Location of Flash ECC Bits

The Flash ECC bits can be read starting at address 0xF0400000. The ECC bits are packed in their memory space as shown in Figure 5-1. The ECC bytes must be read as bytes or halfwords. Reading a single ECC byte with ECC enabled will actually cause 144 bits to be read from the Flash, and the ECC bits will be corrected if necessary. Any errors in either of the two double-words accessed will be recorded in the FEDACSTATUS register (main memory) or the EE_STATUS register (bank 7).

Figure 5-1. ECC Organization for Program Flash (144-Bits Wide)



5.4.2 OTP Memory

5.4.2.1 Flash Bank and Sector Sizes

Flash Bank/Sectoring information can be determined from the device-specific datasheet or can be computed by reading locations in the TI OTP and FMC registers.

The number of banks, which banks are available, and the number of sectors for bank 0 can be read from TI OTP location F008 0158h as shown in Figure 5-2 and described in Table 5-4.

Figure 5-2. TI OTP Bank 0 Sector Information

31-24	23-16
Reserved	BX_NUM_Sectors
R	R

15	14	13	12	11	10	9	8	7-0
B7	B6	B5	B4	B3	B2	B1	B0	NUM_Banks
R-1	R-0	R-0	R-0	R-0	R-0	R-1	R-1	R

LEGEND: R = Read only

Table 5-4. TI OTP Bank 0 Sector Information Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reserved. All bits will be read as 0.
23-16	BX_NUM_Sectors	1-32	Number of sectors in this bank.
15	B7	1	1 = Bank 7 is present
14	B6	0	0 = Bank 6 is not present
13	B5	0	0 = Bank 5 is not present
12	B4	0	0 = Bank 4 is not present
11	B3	0	0 = Bank 3 is not present
10	B2	0	0 = Bank 2 is not present
9	B1	1	1 = Bank 1 is present
8	B0	1	1 = Bank 0 is present
7-0	NUM_Banks	2 or 3	Number of banks on this part.

The bank sector information is repeated once for each bank in the device. The number of sectors is unique for each bank. The number of banks and which banks are implemented is repeated in each location. Use the TI OTP information for bank 0 to determine which banks are in the device, and then read the number of sectors for each bank using the TI OTP locations shown in Table 5-5.

Table 5-5. TI OTP Sector Information Address

Bank	TI OTP Address
0	F008 0158h
1	F008 2158h
2	F008 4158h
3	F008 6158h
4	F008 8158h
5	F008 A158h
6	F008 C158h
7	F008 E158h

5.4.2.2 Package and Memory Size

Package and memory size information can be determined from the device-specific datasheet, or can be computed by reading locations in the TI OTP Bank 0 registers.

The package and memory size can be read from TI OTP location F008 015Ch as shown in Figure 5-3 and described in Table 5-6.

Figure 5-3. TI OTP Bank 0 Package and Memory Size Information (F008 015Ch)

31 28	27 16
Reserved	PACKAGE
R	R

15 0
MEMORY_SIZE
R

LEGEND: R = Read only

Table 5-6. TI OTP Bank 0 Package and Memory Size Information Field Descriptions

Bit	Field	Description
31-28	Reserved	Reserved
27-16	PACKAGE	Count of pins in the package
15-0	MEMORY_SIZE	Flash memory size in Kbytes

5.4.2.3 LPO Trim and Max HCLK

The HF LPO trim solution, LF LPO trim solution and maximum HCLK frequency can be read from TI OTP location F008 01B4h as shown in Figure 5-4 and described in Table 5-7.

Figure 5-4. TI OTP Bank 0 LPO Trim and Max HCLK Information (F008 01B4h)

31 24	23 16
HFLPO_TRIM	LFLPO_TRIM
R	R

15 0
MAX_HCLK
R

LEGEND: R = Read only

Table 5-7. TI OTP Bank 0 LPO Trim and Max HCLK Information Field Descriptions

Bit	Field	Description
31-24	HFLPO_TRIM	HF LPO Trim Solution
23-16	LFLPO_TRIM	LF LPO Trim Solution
15-0	MAX_HCLK	Maximum HCLK Speed

5.4.2.4 Part Number Symbolization

Device part number symbolization information can be determined from the device-specific datasheet or can be computed by reading locations in the TI OTP bank 0 registers.

For example the device part number symbolization "TMS570LS3137CPGEQQ1" can be read from TI OTP bank 0 location F008 01E0h through F008 01FFh as shown in Figure 5-5.

Figure 5-5. TI OTP Bank 0 Symbolization Information (F008 01E0h-F008 01FFh)

0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
0x54	0x4D	0x53	0x35	0x37	0x30	0x4C	0x53	0x33	0x31	0x33	0x37	0x43	0x50	0x47	0x45
R															

0x10	0x11	0x12	0x13	0x14	0x15	0x16	0x17	0x18	0x19	0x1A	0x1B	0x1C	0x1D	0x1E	0x1F
0x51	0x51	0x31	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
R															

LEGEND: R = Read only

5.4.2.5 Deliberate ECC Errors for FMC ECC Checking

Deliberate single-bit and double-bit errors have been placed in the OTP for checking the FMC ECC functionality. Any portion of the 64 bits in TI OTP bank 0 location F008 03F0h through F008 03F7h as shown in Figure 5-6 will generate a single-bit error. Any portion of the 64 bits in TI OTP bank 0 location F008 03F8h through F008 03FFh as shown in Figure 5-6 will generate a double-bit error.

Figure 5-6. TI OTP Bank 0 Deliberate ECC Error Information (F008 03F0h-F008 03FFh)

0x00	0x04	0x08	0x0C
0x12345678	0x9ABCDEF1	0x12345678	0x9ABCDEF3
R	R	R	R

LEGEND: R = Read only, ECC is calculated for the value 0x123456789ABCDEF0

5.5 Power On, Power Off, and Reset Considerations

5.5.1 Error Checking at Power On

As the device is coming out of the device reset sequence, the Flash wrapper reads two configuration words from the TI OTP section of bank 0, the hardware configuration word at address 0xF0080140, and then the AJSM visible password at address 0xF0000000. During these reads ECC is enabled. Single-bit errors are corrected and generate an ESM group 1 channel 6 error event. The first failing address will be latched in the FCOR_ERR_ADD register along with the bit position in FCOR_ERR_POS register and the FEDACSTATUS register flags will be updated to indicate the type of error. Uncorrectable errors will generate an ESM group 3 channel 7 error event, the $\overline{\text{ERROR}}$ pin will be activated, the first failing address will be latched in the FUNC_ERR_ADD register and the FEDACSTATUS register flags will be updated to indicate the type of error.

5.5.2 Flash Integrity when Reset while Programming or Erasing

If a device is reset while programming, then the bits being programmed when reset is asserted are indeterminate; however, the other bits in the Flash are not disturbed. Likewise, If the device is reset while being erased, the sector or sectors being erased will have indeterminate bits; however, the other sectors in the same bank and the other banks will not be disturbed.

5.5.3 Flash Integrity at Power Off

If power is lost during a programming or erase operation, a power-on reset must be asserted before the core supply voltage drops below specification. The $\overline{\text{PORRST}}$ pin has a glitch filter which means that the $\overline{\text{PORRST}}$ pin must be asserted low $t_{f(nPORRST)}$ (2µs) before the core supply drops below Vcc_{MIN} (1.14V). If this requirement is met, then the bits being programmed when $\overline{\text{PORRST}}$ goes low are indeterminate; however, the other bits in the Flash are not disturbed. Likewise, if this requirement is met, and $\overline{\text{PORRST}}$ is asserted while erasing, the sector or sectors being erased will have indeterminate bits; however, the other sectors in the same bank and the other banks will not be disturbed.

5.6 Emulation and SIL3 Diagnostic Modes

5.6.1 System Emulation

During emulation when the SUSPEND signal is high, the data read from memory is still passed to SECDED for correction if ECC_ENABLE is active. If a correctable error is detected, then it is corrected but error event is not generated and error occurrence counter is not incremented if in profiling mode. If a double error is detected, then the raw data is returned without generating a double-error signal.

The SUSPEND signal can be disable by using the SUSP_IGNR bit in the Flash error detection and correction control register 1 (FEDACCTRL1). The SUSPEND signal should not be confused with the suspend_now operation for the FSM.

5.6.2 Diagnostic Mode

The Flash wrapper can be put in diagnostic mode to verify various logic. There are multiple diagnostic modes supported by the wrapper. A specific diagnostic mode is selected via the DIAG_MODE control bits in the diagnostic control register (FDIAGCTRL), as listed in Table 5-8.

The diagnostic mode is only enabled by a 4-bit key stored in the DIAG_EN_KEY bits in FDIAGCTRL register. Only DIAG_EN_KEY = 0101 enables any diagnostic mode and all diagnostic modes use the DIAG_TRIG bit in FDIAGCTRL register to initiate the action.

All tests run from any pipeline mode. Some of the diagnostic modes can corrupt the Flash data access, and generate errors as part of the test. Running in non-pipeline may minimize some of these conditions.

For all modes it is best to follow this sequence:

1. Write 0101 to the DIAG_EN_KEY bits and set the desired DIAG_MODE control bits. This blocks many UERR sources.
2. Set any data registers needed for this mode.
3. Write 1 to the DIAG_TRIG bit to initiate the action and allow UERRs to happen for one cycle.
4. Write 1010 to the DIAG_EN_KEY bits to disable the diagnostic modes.

When the CONF_TYPE is 5, the ECC logic is in the CPU and most diagnostic modes are removed. Some because the logic they test no longer exists and the rest because the path is validated via the ECC path to the CPU.

Table 5-8. DIAG_MODE Encoding

Mode	DIAG_MODE Bits			Description
0	0	0	0	Diagnostic mode is disabled. Same as DIAG_EN_KEY not equal to 5h.
1	0	0	1	ECC Data Correction test mode
2	0	1	0	ECC Syndrome Reporting test mode
3	0	1	1	ECC Malfunction test mode 1 (same data)
4	1	0	0	ECC Malfunction test mode 2 (inverted data)
5	1	0	1	Address Tag Register test mode
6	1	1	0	Reserved
7	1	1	1	ECC Data Correction Diagnostic test mode

5.6.2.1 ECC Data Correction Test Mode: DIAG_MODE = 1

This diagnostic mode can be enabled while ECC logic is also enabled for normal bank read. The Flash wrapper will arbitrate the usage of the ECC logic if a conflict occurs between a normal bank read and diagnostic checking.

When in diagnostic data correction mode, FEMU_xxxx registers contain the 64-bit EEPROM emulation data register, the 19-bit emulation address register and the 8-bit emulation check-bit register. These values are used to enter diagnostic data to exercise the SECDED logic. The user can apply a value with an error in any bit location. When the DIAG_TRIG is set, the SECDED calculation is done and the corrected values are saved back into the same FEMU_xxxx registers. The error position register is also updated to indicate the bit position in error. Either ERR_ONE_FLG or ERR_ZERO_FLG bit is set when a correctable error is detected. The D_COR_ERR bit will also be set in FEDACSTATUS register. For uncorrectable error, the error status bit ERR_PRF_FLG is set as well as the D_UNC_ERR bit in the same register. Status bits should be cleared by the user before applying a new diagnostic data.

It takes multiple CPU transactions to preload the registers with diagnostic values. During this time, the result of the diagnostic logic such as comparator can change. User should apply a trigger by setting DIAG_TRIG bit to 1 as a qualifier after all registers are loaded with intended values. The DIAG_TRIG serves to validate the diagnostic result. Only when DIAG_TRIG is high and a failing result in the diagnostic logic will update the corresponding status flag and the position register.

5.6.2.2 ECC Syndrome Reporting Test Mode: DIAG_MODE = 2

When in diagnostic syndrome reporting mode, the resulting syndrome calculated by SECDED is captured into the ECC check-bit register FEMU_ECC. The syndrome can be read by the user and compare with a known syndrome value. Diagnostic data in FEMU_DxSW and FEMU_ADDR is not corrected and the error position register is not updated. The FEDACSTATUS register error bits are not updated during this mode. See Figure 5-7.

For devices with ECC_IN_CPU (CONF_TYPE = 5), the resulting FEMU_ECC value represents the 32-bit byte swapped values. Here, bytes 7654_3210 are rearranged to 4567_0123. For instance, if the syndrome shows an error in data bit 33, it would really be an error in EMU_DMW bit 57. You can also XOR the data bit position with “011000”. (21h XOR 18h => 39h)

NOTE: The user should pre-load the registers with the test values with DIAG_TRIG = 0. After all test values are written, the DIAG_TRIG should then be set high to validate the diagnostic result.

5.6.2.3 ECC Malfunction Test Mode 1: DIAG_MODE = 3

There are three inputs to the malfunction detection logic: the resulting syndrome, the original uncorrected data, and the final corrected data. See Figure 5-7.

In normal function, the malfunction detection logic will detect an error if the syndrome is 0 and if the data before the correction and the data after the correction is not equal; or if the syndrome is not 0 and if the data before the correction and data after the correction is equal to each other. During diagnostic mode 3 or 4, user supplied values are sent to the malfunction logic. No functional checking is done by the ecc_malfunction logic while the mode is 3 or 4.

Diagnostic mode 3 is also known as “same data” mode. A diagnostic value can be stored in the ECC checkbit register. The value stored in the 64-bit Raw data register will be supplied to the two inputs of the malfunction comparator logic. If a non-zero value is stored in the Raw ECC checkbit register (FRAW_ECC), then the malfunction logic should detect it as an error and set the ECC_B2_MAL_ERR bit (ECC malfunction error). There is one ECC_B2_MAL_ERR bit for each SECDED block and they are called ECC1_MAL_ERR and ECC0_MAL_ERR and are selectable with the DIAG_ECC_SEL bit. The DIAG_TRIG is set to initiate this mode.

The FRAW_DATAx, FRAW_ECC, and FUNC_ERR_ADD will load on an ECC_MAL ERR but will not contain useful information during Diag mode 3.

5.6.2.4 ECC Malfunction Test Mode 2: DIAG_MODE = 4

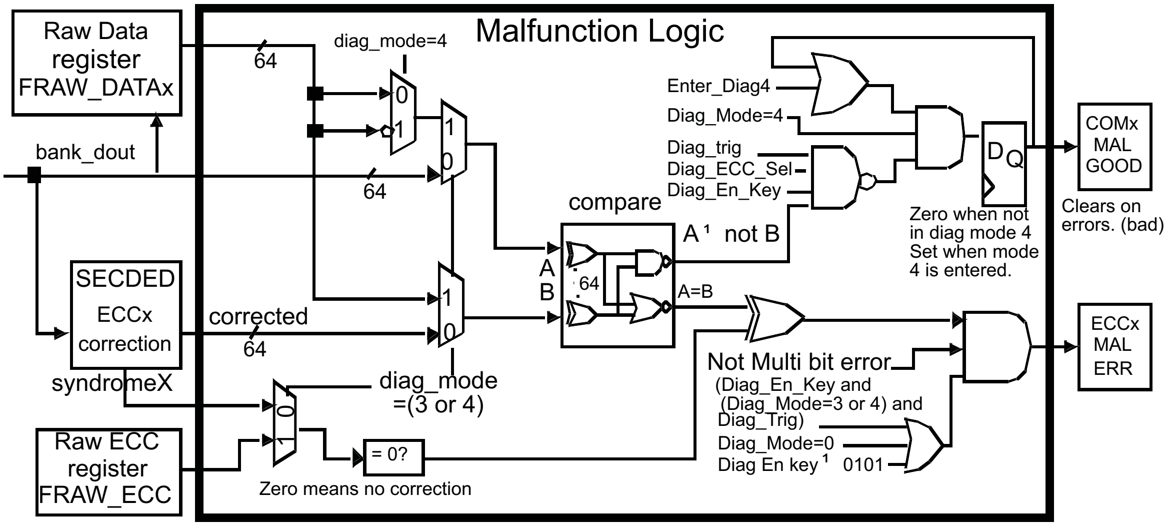
Diagnostic mode 4 is also known as “inverted data” mode. A diagnostic value can be stored in the ECC checkbit register. A value stored in the 64-bit Raw data register and its bit-wise inverted counterpart will be supplied to the two inputs of the malfunction comparator logic. If a zero value is stored in the Raw ECC checkbit register (FRAW_ECC), then the malfunction logic should detect it as an error and set the ECC_B2_MAL_ERR bit (diagnostic ECC malfunction). See Figure 5-7.

In this mode only, the EE_CMG bit (Compare Malfunction Good) is cleared if any one of the 64 XOR gates is malfunctioning. There is one EE_CMG bit for each SECDED block and they are called COM1_MAL_GOOD and COM0_MAL_GOOD. The EE_CMG bits go high when entering diagnostic mode 4 and go low and stay low if an error is detected. These bits are only valid in diagnostic mode 4 and outside of this mode the bits are 0.

Set the DIAG_ECC_SEL bits before entering mode 4 or enter mode 4 from a non-mode 4 and set the DIAG_ECC_SEL bits at the same time. Do not just change the DIAG_ECC_SEL bits or the corresponding MAL_GOOD will not get set because it did not enter mode 4 correctly.

The FRAW_DATAx, FRAW_ECC, and FUNC_ERR_ADD will load on an ECC_MAL ERR but will not contain useful information during Diag mode 4.

Figure 5-7. ECC Malfunction Test Logic



5.6.2.5 Address Tag Register Test Mode: DIAG_MODE = 5

NOTE: The test code for Diag mode 5 needs to be executed from RAM or when executed from Flash, the Flash Address Wait State (ASWSTEN) bit in the FRDCNTL register should be set to 1. This is due to conflicting accesses by the CPU and Flash wrapper during the test execution.

There are four sets of address tag registers. Each set consists of a primary and a duplicate address tag registers. Normally, these registers store the recently issued CPU addresses during pipeline mode. To detect errors in these registers, the primary and duplicate address tag registers are continuously compared to each other if the buffer is valid. If they are different, then an address tag register error event is generated.

These registers are memory-mapped. All primary address tag registers are memory-mapped to one address and, likewise, all duplicate tag registers are mapped to another single address. During diagnostic mode, each individual set can be selected by the DIAG_BUF_SEL (Diagnostic Buffer Select) bit in the FDIAGCTRL register. User-supplied values can be written into the selected set during a diagnostic mode. If different values are written into the primary and the duplicate address tag registers, then the ADD_TAG_ERR (Address Tag Error) flag in the FEDACSTATUS register will be set. This diagnostic mode uses the FRAW_DATAL register to supply the alternate address when DIAG_TRIG is set. The FUNC_ERR_ADD register will not contain useful information during Diag mode 5. It will also trigger the normal uncorrectable register freeze.

All address tags and buffer valid bits will be cleared to 0 when leaving Diag mode 5. Going to mode 5 and back out clears the pipeline buffers and is useful for other test modes also. No functional checking is done by the address tag logic while the mode is 5.

NOTE: The user should pre-load the registers with the test values with DIAG_TRIG = 0. After all test values are written, the DIAG_TRIG should then be set high to validate the diagnostic result.

5.6.2.6 ECC Data Correction Diagnostic Test Mode: DIAG_MODE = 7

Testing the error correction and ECC logic in the CPU involves corrupting the ECC value returned to the CPU. By inverting one or more bits of the ECC, the CPU will detect errors in a selected data or ECC bit, or in any possible value returned by the ECC.

To set an error for a particular bit use the syndrome, see Section 5.3.3. For example, if you want to corrupt data bit 62 then put the value 70h into the test register.

The method uses the DATA_INV_PAR value in the FPAR_OVR register to alter the ECC during a slave access cycle. The value in the DATA_INV_PAR register will be XORed with the current ECC to give a bad ECC value to the CPU. This only will occur when the DIAG_MODE is 7, the PAR_OVR_KEY is 5, the DIAG_EN_KEY in the FDIAGCTRL register is 5 and the access is a slave cycle.

This mode can set the FEDACSTATUS register status error bits B1_UNC_ERR or ERR_ZERO_FLG, but it will not set the D_UNC_ERR nor D_COR_ERR bits. Also, the logic to support the ECCx_MAL_ERR and COMx_MAL_GOOD bits is not implemented for the CPU path so these bits are not set.

The sequence to do this test is:

1. Make sure the true DMA module is off.
2. Put 5h in BUS_PAR_DIS and 5h in PAR_OVR_KEY fields (00005Axxh) of the FPAR_OVR register.
3. Put the desired value in DAT_INV_PAR field of the FPAR_OVR register.
4. Put 7h in DIAG_MODE and 5h in DIAG_EN_KEY fields of the FDIAGCTRL register.
5. Read desired address from the mirrored Flash location. Mirrored Flash starts at address 0x20000000.
6. Put 0 in DIAG_MODE or Ah in one of the key fields to turn off this test.
7. Check error registers (FCOR_ERR_ADD, FEDACSTATUS, and FUNC_ERR_ADD) for ECC errors.
8. Repeat as necessary to test out the ECC.
9. Put 0 in DIAG_MODE field of the FDIAGCTRL register and Ah in both of the key fields to completely disable this test at the end of the test.
10. Put 2h in PAR_OVR_KEY field (00005400h) of the FPAR_OVR register to clear DAT_INV_PAR field.

5.6.3 *Diagnostic Mode Summary*

The following tables give a summary of the input registers needed for each mode, the possible registers that can change, and the possible error bits in FEDACSTATUS register that may set.

Table 5-9. Bus 1 Diagnostic Mode Summary

DIAG MODE	Name	Inputs	Possible Outputs	Possible Error Bits Set	Notes
1	ECC Data Correction test mode				Not Applicable
2	ECC Syndrome Reporting test mode				Not Applicable
3	ECC Malfunction test mode 1				Not Applicable
4	ECC Malfunction test mode 2				Not Applicable
5	Address Tag Register test mode	FPRIM_ADD_TAG FDUP_ADD_TAG FRAW_DATAL	FUNC_ERR_ADD [1]	ADD_TAG_ERR	
6	Reserved				
7	ECC Data Correction Diagnostic test mode	DAT_INV_PAR	FUNC_ERR_ADD FCOR_ERR_ADD	B1_UNC_ERR ERR_ZERO_FLG	Slave access only

[1] Register output value will change, but will not contain useful information.

Table 5-10. Bus 2 and ECC Diagnostic Mode Summary

DIAG MODE	Name	Inputs	Possible Outputs	Possible Error Bits Set	Notes
1	ECC Data Correction test mode	FEUM_DMSW FEMU_DLSW FEMU_ECC FEMU_ADDR	FEMU_ECC FUNC_ERR_ADD FCOR_ERR_ADD FCOR_ERR_POS	D_UNC_ERR D_COR_ERR ERR_ONE_FLG ERR_ZERO_FLG ERR_PRF_FLG	
			FEMU_ECC EE_UNC_ERR_ADD EE_COR_ERR_ADD EE_COR_ERR_POS	EE_D_UNC_ERR EE_D_COR_ERR EE_ERR_ONE_FLG EE_ERR_ZERO_FLG EE_ERR_PRF_FLG	
2	ECC Syndrome Reporting test mode	FEMU_DMSW FEMU_DLSW FEMU_ECC FEMU_ADDR	FEMU_ECC	NA	
3	ECC Malfunction test mode 1	FRAW_DATAH FRAW_DATAL FRAW_ECC	FRAW_DATAH [1] FRAW_DATAL [1] FRAW_ECC [1] FUNC_ERR_ADD [1]	ECC_B2_MAL_ERR D_UNC_ERR	
			FRAW_DATAH [1] FRAW_DATAL [1] FRAW_ECC [1] EE_UNC_ERR_ADD [1]	EE_CME EE_D_UNC_ERR	

[1] Register output value will change, but will not contain useful information.

Table 5-10. Bus 2 and ECC Diagnostic Mode Summary (continued)

DIAG MODE	Name	Inputs	Possible Outputs	Possible Error Bits Set	Notes
4	ECC Malfunction test mode 2	FRAW_DATAH	FRAW_DATAH [1]	COMB2_MAL_G	
		FRAW_DATAL	FRAW_DATAL [1]	ECC_B2_MAL_ERR	
		FRAW_ECC	FRAW_ECC [1]	D_UNC_ERR	
			FUNC_ERR_ADD [1]		
			FRAW_DATAH [1]	EE_CMG	
			FRAW_DATAL [1]	EE_CME	
			FRAW_ECC [1]	EE_D_UNC_ERR	
			EE_UNC_ERR_ADD [1]		
5	Address Tag Register test mode				Not applicable
6	Reserved				Not applicable
7	ECC Data Correction Diagnostic test mode				Not applicable

Table 5-11. Port Signals Diagnostic Mode Summary

			Uncorrectable Error	Correctable Error	Address Bus Parity Error	FEE Uncorrectable Error	FEE Correctable Error
DIAG MODE	Name	Error In	ESM Group 3 Channel 7	ESM Group 1 Channel 6	ESM Group 2 Channel 4	ESM Group 1 Channel 36	ESM Group 1 Channel 35
1	ECC Data Correction test mode	Bus 2	Yes	Yes	No	No	No
		EEPROM	No	No	No	Yes	Yes
2	ECC Syndrome Reporting test mode	Bus 2	No	No	No	No	No
		EEPROM	No	No	No	No	No
3	ECC Malfunction test mode 1	Bus 2	Yes	No	No	No	No
		EEPROM	No	No	No	Yes	No
4	ECC Malfunction test mode 2	Bus 2	Yes	No	No	No	No
		EEPROM	No	No	No	Yes	No
5	Address Tag Register test mode	Bus 1	Yes	No	No	No	No
6	Reserved						
7	ECC Data Correction Diagnostic test mode	Bus 1	Yes	Yes	No	No	No

5.6.4 *Read Margin*

When the bits are programmed or erased, they are checked against a program_verify or erase_verify reference level that is far away from the normal read reference point. Over time, bit levels may drift toward the normal read point and if it is too much then a bit will read the wrong value. To counteract this, the bits can be read using different read_margin reference points to give an early detection of the problem. The bits can then be either re-programmed (most common) or the sector can be erased and reprogrammed.

5.7 Control Registers

This section details the Flash module registers, summarized in Table 5-12. A detailed description of each register and its bits is also provided.

The Flash module control registers can only be read and/or written by the CPU while in privileged mode. Each register begins on a word boundary. All registers are 32-bit, 16-bit and 8-bit accessible. The start address of the Flash module is FFF8 7000h.

Table 5-12. Flash Control Registers

Address	Acronym	Register Description	Section
FFF8 7000h	FRDCNTL	Flash Option Control Register	Section 5.7.1
FFF8 7008h	FEDACTRL1	Flash Error Detection and Correction Control Register 1	Section 5.7.2
FFF8 700Ch	FEDACTRL2	Flash Error Detection and Correction Control Register 2	Section 5.7.3
FFF8 7010h	FCOR_ERR_CNT	Flash Correctable Error Count Register	Section 5.7.4
FFF8 7014h	FCOR_ERR_ADD	Flash Correctable Error Address Register	Section 5.7.5
FFF8 7018h	FCOR_ERR_POS	Flash Correctable Error Position Register	Section 5.7.6
FFF8 701Ch	FEDACSTATUS	Flash Error Detection and Correction Status Register	Section 5.7.7
FFF8 7020h	FUNC_ERR_ADD	Flash Uncorrectable Error Address Register	Section 5.7.8
FFF8 7024h	FEDACSDIS	Flash Error Detection and Correction Sector Disable Register	Section 5.7.9
FFF8 7028h	FPRIM_ADD_TAG	Flash Primary Address Tag Register	Section 5.7.10
FFF8 702Ch	FDUP_ADD_TAG	Flash Duplicate Address Tag Register	Section 5.7.11
FFF8 7030h	FBPROT	Flash Bank Protection Register	Section 5.7.12
FFF8 7034h	FBSE	Flash Bank Sector Enable Register	Section 5.7.13
FFF8 7038h	FBBUSY	Flash Bank Busy Register	Section 5.7.14
FFF8 703Ch	FBAC	Flash Bank Access Control Register	Section 5.7.15
FFF8 7040h	FBFALLBACK	Flash Bank Fallback Power Register	Section 5.7.16
FFF8 7044h	FBPRDY	Flash Bank/Pump Ready Register	Section 5.7.17
FFF8 7048h	FPAC1	Flash Pump Access Control Register 1	Section 5.7.18
FFF8 704Ch	FPAC2	Flash Pump Access Control Register 2	Section 5.7.19
FFF8 7050h	FMAC	Flash Module Access Control Register	Section 5.7.20
FFF8 7054h	FMSTAT	Flash Module Status Register	Section 5.7.21
FFF8 7058h	FEMU_DMSW	EEPROM Emulation Data MSW Register	Section 5.7.22
FFF8 705Ch	FEMU_DLSW	EEPROM Emulation Data LSW Register	Section 5.7.23
FFF8 7060h	FEMU_ECC	EEPROM Emulation ECC Register	Section 5.7.24
FFF8 7068h	FEMU_ADDR	EEPROM Emulation Address Register	Section 5.7.25
FFF8 706Ch	FDIAGCTRL	Diagnostic Control Register	Section 5.7.26
FFF8 7070h	FRAW_DATAH	Uncorrected Raw Data High Register	Section 5.7.27
FFF8 7074h	FRAW_DATAL	Uncorrected Raw Data Low Register	Section 5.7.28
FFF8 7078h	FRAW_ECC	Uncorrected Raw ECC Register	Section 5.7.29
FFF8 707Ch	FPAR_OVR	Parity Override Register	Section 5.7.30
FFF8 70C0h	FEDACSDIS2	Flash Error Detection and Correction Sector Disable Register 2	Section 5.7.31
FFF8 7288h	FSM_WR_ENA	FSM Register Write Enable	Section 5.7.32
FFF8 72A4h	FSM_SECTOR	FSM Sector Register	Section 5.7.33
FFF8 72B8h	EEPROM_CONFIG	EEPROM Emulation Configuration Register	Section 5.7.34
FFF8 7308h	EE_CTRL1	EEPROM Emulation Error Detection and Correction Control Register 1	Section 5.7.35
FFF8 730Ch	EE_CTRL2	EEPROM Emulation Error Detection and Correction Control Register 2	Section 5.7.36
FFF8 7310h	EE_COR_ERR_CNT	EEPROM Emulation Correctable Error Count Register	Section 5.7.37
FFF8 7314h	EE_COR_ERR_ADD	EEPROM Emulation Correctable Error Address Register	Section 5.7.38
FFF8 7318h	EE_COR_ERR_POS	EEPROM Emulation Correctable Error Bit Position Register	Section 5.7.39
FFF8 731Ch	EE_STATUS	EEPROM Emulation Error Status Register	Section 5.7.40
FFF8 7320h	EE_UNC_ERR_ADD	EEPROM Emulation Uncorrectable Error Address Register	Section 5.7.41
FFF8 7400h	FCFG_BANK	Flash Bank Configuration Register	Section 5.7.42

5.7.1 Flash Option Control Register (FRDCNTL)

FRDCNTL supports pipeline mode. This register controls Flash timings for the main Flash banks. For the equivalent register that controls Flash timings for the EEPROM Emulation Flash bank (bank 7), see Section 5.7.34.

Figure 5-8. Flash Option Control Register (FRDCNTL) [offset = 00h]

31 – 16
Reserved
R-0

15 – 12	11 – 8
Reserved	RWAIT
R-0	R/WP-1

7 – 5	4	3 – 1	0
Reserved	ASWSTEN	Reserved	ENPIPE
R-0	R/WP-0	R-0	R/WP-0

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

Table 5-13. Flash Option Control Register (FRDCNTL) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reads return 0. Writes have no effect.
11-8	RWAIT	0-Fh	Random/data Read Wait State The random read wait state bits indicate how many wait states are added to a Flash read access. In Pipeline mode there is always one wait state even when RWAIT is cleared to 0. **Note:** The required wait states for each HCLK frequency can be found in the device-specific data manual.
7-5	Reserved	0	Reads return 0. Writes have no effect.
4	ASWSTEN		Address Setup Wait State Enable
		0	Address Setup Wait State is disabled.
		1	Address Setup Wait State is enabled. Address is latched one cycle before decoding to determine pipeline hit or miss. Address Setup Wait State is only available in pipeline mode. **Note:** The required address wait state for each HCLK frequency can be found in the device-specific data manual.
3-1	Reserved	0	Reads return 0. Writes have no effect.
0	ENPIPE		Enable Pipeline Mode
		0	Pipeline mode is disabled.
		1	Pipeline mode is enabled.

5.7.2 Flash Error Detection and Correction Control Register 1 (FEDACCTRL1)

This register controls ECC event detection for the main Flash banks. For the equivalent register that controls ECC event detection for the EEPROM Emulation Flash bank (bank 7), see Section 5.7.35.

Figure 5-9. Flash Error Detection and Correction Control Register 1 (FEDACCTRL1) [offset = 08h]

31	25	24
Reserved		SUSP_IGNR
R-0		R/WP-0

23	20	19	16
Reserved		EDACMODE	
R-0		R/WP-Ah	

15	11	10	9	8
Reserved		EOFEN	EZFEN	EPEN
R-0		R/WP-0	R/WP-0	R/WP-0

7	4	3	0
Reserved		EDACEN	
R-0		R/WP-5h	

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

Table 5-14. Flash Error Detection and Correction Control Register 1 (FEDACCTRL1) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	SUSP_IGNR		Suspend Ignore In emulation mode, for example, viewing memory in the debugger's window, the CPU suspend signal is set. This bit determines whether the CPU suspend signal is ignored by the Flash module.
		0	CPU suspend signal blocks error bits setting and unfreezing. The Flash module blocks all errors from setting the error bits in emulation mode and blocks the unfreezing of the bits and registers by reading the FUNC_ERR_ADD register.
		1	CPU suspend has no effect on error bit setting and unfreezing. The Flash module ignores the CPU suspend signal and allows the error bits to set even in emulation mode. It also allows the Flash module to unfreeze the error bits and other registers by reading the FUNC_ERR_ADD register even in emulation mode.
23-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	EDACMODE		Error Correction Mode for the main Flash banks. For EEPROM Emulation Flash bank (bank 7), see Section 5.7.35.
		5h	Single-bit errors during reads from OTP, ECC and the mirrored space (starting at 0x20000000) of banks 0 through 6, will be treated as uncorrectable errors by the Flash wrapper. The wrapper will assert an ESM group 3 error on channel 7 and the $\overline{\text{ERROR}}$ pin will be activated. No abort will be taken by the CPU.
		All Other Values	Single-bit errors during reads from OTP, ECC and the mirrored space (starting at 0x20000000) of banks 0 through 6, will be treated as correctable errors by the Flash wrapper. The wrapper will assert an ESM group 1 error on channel 6. The single-bit error will be corrected. **Note:** This mode does not affect reads from the main program Flash starting at address 0. **Note:** Reading ECC bits will generate an ECC error based on the contents of the 8 ECC bits and the 64 data bits they protect.
15-11	Reserved	0	Reads return 0. Writes have no effect.

Table 5-14. Flash Error Detection and Correction Control Register 1 (FEDACCTRL1) Field Descriptions (continued)

Bit	Field	Value	Description
10	EOFEN		Event on One's Fail Enable
		0	No ESM error event is generated on a single-bit error where a 1 reads as a 0 when reading from the OTP or ECC memory locations.
		1	An ESM error event is generated on a single-bit error where a 1 reads as a 0 when reading from the OTP or ECC memory locations.
			Note: When either the EOFEN or the EZFEN bit is set, an error event will be generated on ESM group 1 channel 6 when any correctable error is generated by reading the main memory.
9	EZFEN		Event on Zero's Fail Enable
		0	No ESM error event is generated on a single-bit error where a 0 reads as a 1 when reading from the OTP or ECC memory locations.
		1	An ESM error event is generated on a single-bit error where a 0 reads as a 1 when reading from the OTP or ECC memory locations.
			Note: When either the EOFEN or the EZFEN bit is set, an error event will be generated on ESM group 1 channel 6 when any correctable error is generated by reading the main memory.
8	EPEN		Error Profiling Enable.
		0	Error profiling is disabled.
		1	Error profiling is enabled.
			The correctable error event is generated (ESM group 1 channel 6) when the number of CPU accesses of correctable bit errors detected and corrected has reached the threshold value defined in the FEDACCTRL2 register.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	EDACEN		Error Detection and Correction Enable
		5h	CPU single and double error event signals are blocked.
			Note: It is recommended to enable ECC in the Flash wrapper by writing 1010 to these bits before enabling ECC in the CPU. If ECC is enabled in the CPU, but not in the wrapper, the CPU will still check and correct single-bit ECC errors, and generate aborts on uncorrectable errors for the main Flash. However, the generation of ESM events, the capture of failing addresses and the detections and correction of errors in the OTP will be prevented.
		All Other Values	Error Detection and Correction events are captured and sent to the ESM.

5.7.3 Flash Error Correction and Correction Control Register 2 (FEDACCTRL2)

This register applies to ECC event detection for the main Flash banks. For the equivalent register that applies to the EEPROM Emulation Flash bank (bank 7), see Section 5.7.36.

Figure 5-10. Flash Error Correction and Correction Control Register 2 (FEDACCTRL2) [offset = 0Ch]

31	16
Reserved	
R-0	

15	0
SEC_THRESHOLD	
R/WP-0	

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

Table 5-15. Flash Error Correction Control and Correction Register 2 (FEDACCTRL2) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	SEC_THRESHOLD		Single Error Correction Threshold When error profiling is enabled, this register contains the threshold value for the SEC (single error correction) occurrences before a correctable error event is generated (ESM group 1, channel 6). A threshold of zero disables the threshold so that it does not generate an event.

5.7.4 Flash Correctable Error Count Register (FCOR_ERR_CNT)

This register applies to the main Flash banks. For the equivalent register that applies to the EEPROM Emulation Flash bank (bank 7), see Section 5.7.37.

Figure 5-11. Flash Correctable Error Count Register (FCOR_ERR_CNT) [offset = 10h]

31	16
Reserved	
R-0	

15	0
FERRCNT	
R/WP-0	

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

Table 5-16. Flash Correctable Error Count Register (FCOR_ERR_CNT) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	FERRCNT		Single Error Correction Count This register contains the number of SEC (single error correction) occurrences. Writing any value to this register resets the count value to 0. The counter resets to 0 when it increments to be equal to the single error correction threshold. This register only increments when profiling mode is enabled. This register is not affected by the EOFEN or EZEFEN error control bits in the FEDACCTRL1 register.

5.7.5 Flash Correctable Error Address Register (FCOR_ERR_ADD)

This register applies to the main Flash banks. For the equivalent register that applies to the EEPROM Emulation Flash bank (bank 7), see Section 5.7.38.

The error address is captured during errors when either EOFEN or EZFEN enable bit is set. During error profiling mode when only EPEN is set, the error address is not captured if a correctable error is detected. This register is frozen while either the ERR_ZERO_FLG or the ERR_ONE_FLG bit is set in the FEDACSTATUS register.

During emulation mode, this address is frozen even when read. By setting the SUSP_IGNR bit (see Table 5-14), this register can be unfrozen in emulation mode.

This register is not changed with the reset signal and contains unknown data at power-up.

Figure 5-12. Flash Correctable Error Address Register (FCOR_ERR_ADD) [offset = 14h]

31–16
COR_ERR_ADD
R-u

15–3	2–0
COR_ERR_ADD	B_OFF
R-u	R-u

LEGEND: R = Read only; -*n* = value after reset; -u = unchanged value on internal reset, cleared on power up

Table 5-17. Flash Correctable Error Address Register (FCOR_ERR_ADD) Field Descriptions

Bit	Field	Value	Description
31-3	COR_ERR_ADD	0-1FFF FFFFh	Correctable Error Address COR_ERR_ADD records the CPU logical address of which a correctable error is detected by the ECC logic. This error address is frozen from begin updated until it is read by the CPU. Additional error are blocked until this register is read.
2-0	B_OFF	0-7h	Byte Offset Since ECC is checked on 64 bit data, when checking main memory or OTP, the address captured is aligned to a 64-bit boundary with address bits[2:0] equal to 0.When reading from the ECC bytes, these bits will indicate the failing address of the ECC location associated with the failure. When reading an ECC byte, the ECC is checked against the 64 data bits they protect.

5.7.6 Flash Correctable Error Position Register (FCOR_ERR_POS)

This register applies to the main Flash banks. For the equivalent register that applies to the EEPROM Emulation Flash bank (bank 7), see Section 5.7.39.

Note: The bit error position is only detected during reads of the OTP, the mirrored Flash image or the ECC bytes. Single-bit errors corrected during reads of the main memory will only capture the failing address, but not the bit position. The bit position is captured during errors when either EOFEN or EZFEN enable bit is set. During error profiling mode when only EPEN is set, the bit position is not captured if a correctable error is detected. This register is frozen while either the ERR_ZERO_FLG or the ERR_ONE_FLG bit is set in the FEDACSTATUS register.

During emulation mode, this address is frozen even when read. By setting the SUSP_IGNR bit, this register can be unfrozen in emulation mode.

This register is not changed with the reset signal and contains unknown data at power-up.

Figure 5-13. Flash Correctable Error Position Register (FCOR_ERR_POS) [offset = 18h]

31-16
Reserved
R-0

15-10	9	8	7-0
Reserved	BUS2	TYPE	ERR_POS
R-0	R-u	R-u	R-u

LEGEND: R = Read only; -*n* = value after reset; -u = unchanged value on internal reset, cleared on power up

Table 5-18. Flash Correctable Error Position Register (FCOR_ERR_POS) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return 0. Writes have no effect.
9	BUS2		Bus 2 Error
		0	The error was in the main Flash.
		1	The error was from an OTP read.
8	TYPE		ErrorType
		0	The error was one of the 64 data bits.
		1	The error was one of the 8 check bits.
7-0	ERR_POS		The bit address of the single-bit error.

5.7.7 Flash Error Detection and Correction Status Register (FEDACSTATUS)

This register applies to the main Flash banks. For the equivalent register that applies to the EEPROM Emulation Flash bank (bank 7), see Section 5.7.40.

All these error status bits can be cleared by writing a 1 to the bit. Writing a 0 has no effect.

The correctable errors in bits 2:0, and FSM_DONE bit 24, must be cleared before the end of their error event service routine or else the error event will re-issue.

Figure 5-14. Flash Error Detection and Correction Status Register (FEDACSTATUS) [offset = 1Ch]

31-26	25	24
Reserved	Reserved	FSM_DONE
R-0	R-0	RCP-u

23-20	19	18	17	16
Reserved	COMB2_MAL_G	ECC_B2_MAL_ERR	B2_UNC_ERR	B2_COR_ERR
R-0	RCP-u	RCP-u	RCP-u	RCP-u

15-13	12	11	10	9	8
Reserved	D_UNC_ERR	ADD_TAG_ERR	ADD_PAR_ERR	Reserved	B1_UNC_ERR
R-0	RCP-u	RCP-u	RCP-u	R-0	RCP-u

7-4	3	2	1	0
Reserved	D_COR_ERR	ERR_ONE_FLG	ERR_ZERO_FLG	ERR_PRF_FLG
R-0	RCP-u	RCP-u	RCP-u	RCP-u

LEGEND: R = Read only; RCP = Read and Clear in Privilege Mode; -*n* = value after reset; -u = unchanged value on internal reset, cleared on power up

Table 5-19. Flash Error Detection and Correction Status Register (FEDACSTATUS) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	Reads return 0. Writes have no effect.
25	Reserved	0	Reserved
24	FSM_DONE		Flash State Machine Done This bit is set to 1 when the Flash state machine completes a program or erase operation. This bit will generate an interrupt on VIM channel 61 if the FSM_EVT_EN bit of the FSM_ST_MACHINE register is set. This bit must be cleared by writing a 1 to it in the interrupt routine to clear the interrupt request.
23-20	Reserved	0	Reads return 0. Writes have no effect.
19	COMB2_MAL_G		Bus 2 Compare Malfunction Flag
		0	Compare Malfunction is detected on the Bus 2 SECDED in diagnostic mode 4 or is not in diagnostic mode.
		1	Compare Malfunction is not detected on the Bus 2 SECDED or entered diagnostic mode 4.
			This bit becomes 1 when entering diagnostic mode 4, with DIAG_ECC_SEL field set to 0 or 1, and will be cleared if diagnostic mode 4 triggers an error. This bit will reset to 0 and will be 0 outside of diagnostic mode 4. Writing a 1 will set this bit to 1 only in diagnostic mode 4; otherwise, writes have no effect.
18	ECC_B2_MAL_ ERR		Bus 2 ECC Malfunction Error Flag ECC_B2_MAL_ERR bit is set, if Bus 2 data is not corrected and a single-bit error is seen or data is corrected and a single-bit error is not seen.
		0	SECDED malfunction is not detected on Bus 2.
		1	SECDED malfunction is detected on Bus 2.

Table 5-19. Flash Error Detection and Correction Status Register (FEDACSTATUS) Field Descriptions (continued)

Bit	Field	Value	Description
17	B2_UNC_ERR		Bus 2 Uncorrectable Error Flag
		0	No bus 2 uncorrectable errors were detected.
		1	A bus 2 uncorrectable error was detected.
			Two or more bits in the data, or ECC field; or a single-bit error in the address field have been found in error. Address-bit errors are considered an uncorrectable error. The FUNC_ERR_ADD register should contain the Bus 2 error location. This error will generate an ESM group 3 channel 7 event.
16	B2_COR_ERR		Bus 2 Correctable Error Flag
		0	No bus 2 correctable error was detected.
		1	A bus 2 correctable error was detected.
			One bit in the data, or ECC field has been found in error. Either the ERR_ONE_FLAG or ERR_ZERO_FLAG should be set in this register along with this bit. The FCOR_ERR_ADD register should contain the error address, and the FCOR_ERR_POS register should contain the failing bit position. This error will generate an ESM group 1 channel 6 event.
15-13	Reserved	0	Reads return 0. Writes have no effect.
12	D_UNC_ERR		Diagnostic Uncorrectable Error Flag
			This bit sets when diagnostic mode 1 discovers a multi-bit error using the ECC. This means two or more bits in the data, address or ECC field have been found in error. The ECC is capable of correcting a single-bit error and this would show up in the D_COR_ERR bit. The ECC can always detect two bit errors. Three or more bit errors may escape detection with the ECC. This bit also may set during other uncorrectable errors and during the diagnostic mode like address tag errors and ECC malfunctions.
11	ADD_TAG_ERR		Address Tag Register Error Flag
		0	No address tag register error was detected.
		1	An address tag register error was detected.
			This bit is set if the primary address tag has a hit but the duplicate address tag does not match the primary address tag. This bit is functional only when pipeline mode is enabled. This error will create an ESM group 3 channel 7 event.
10	ADD_PAR_ERR		Address Parity Error Flag
		0	No address parity error was detected.
		1	A parity error was detected on the incoming address bus.
			The full 32 bit address will be stored in FUNC_ERR_ADD register. This error will create an ESM group 2 channel 4 event.
9	Reserved	0	Reads return 0. Writes have no effect.
8	B1_UNC_ERR		Bus 1 Uncorrectable Error Flag
		0	No bus 1 uncorrectable errors were detected.
		1	A bus 1 uncorrectable error was detected.
			Two or more bits in the data, or ECC field; or a single-bit error in the address field have been found in error. Address-bit errors are considered an uncorrectable error. The FUNC_ERR_ADD register will contain the Bus 1 error location. This error will generate an ESM group 3 channel 7 event..
7-4	Reserved	0	Reads return 0. Writes have no effect.
3	D_COR_ERR		Diagnostic Correctable Error Status Flag
			This bit sets when diagnostic mode 1 discovers a single-bit correctable error using the ECC. Multi-bit errors are flagged using the D_UNC_ERR bit. The uncorrectable error address must be unfrozen in order to set this bit.

Table 5-19. Flash Error Detection and Correction Status Register (FEDACSTATUS) Field Descriptions (continued)

Bit	Field	Value	Description
2	ERR_ONE_FLG		Error on One Fail Status Flag
		0	No correctable error where a 1 was read as a 0 on bus 2.
		1	A correctable error occurred on bus 2 where a 1 was read as a 0.
			This bit is set if the EOFEN (Error on One Fail Enable) bit is set then, and one bit in the data, or ECC field which should have been read as a 1 reads as a 0. During the read, the bit is corrected to a 1. The FCOR_ERR_ADD register will contain the bus 2 error address, and the FCOR_ERR_POS register will contain the failing bit position. This error will generate an ESM group 1 channel 6 event. When this bit is set, the B2_CORR_ERR bit will also be set. This error will generate an ESM group 1 channel 6 event.
1	ERR_ZERO_ FLG		Error on Zero Fail Status Flag
		0	No correctable errors on bus 1 nor any correctable errors on bus 2 where a 0 was read as a 1.
		1	A correctable error occurred on bus 1, or a correctable error occurred on bus 2 where a 0 was read as a 1.
			This bit is set if the EZFEN (Error on Zero Fail Enable) bit is set and a correctable error is detected on bus 2 where a 0 is read as a 1 and corrected to a 0, or if either the EZFEN or the EOFEN bits are set and any single-bit error is detected and corrected on bus 1. The FCOR_ERR_ADD register will contain the error address. If the error was on bus 2, then the B2_COR_ERR bit will also be set and the FCOR_ERR_POS register will contain the failing bit position. The FCOR_ERR_POS register will not indicate the failing bit position for a bus 1 error. This error will generate an ESM group 1 channel 6 event.
0	ERR_PRF_FLG		Error Profiling Status Flag
		0	Error profiling is not enabled, or the number of correctable errors has not reached the threshold programmed into the SEC_THRESHOLD register.
		1	Error profiling is enabled and the number of correctable errors has reached the threshold programmed into the SEC_THRESHOLD register.

5.7.8 Flash Uncorrectable Error Address Register (FUNC_ERR_ADD)

This register applies to ECC event detection for the main Flash banks. For the equivalent register that applies to the EEPROM Emulation Flash bank (bank 7), see Section 5.7.41.

During emulation mode, this address is frozen even when read. By setting the SUSP_IGNR bit, (see Table 5-14) this register can be unfrozen in emulation mode.

This register is not changed with the reset signal and contains unknown data at power-up.

Figure 5-15. Flash Uncorrectable Error Address Register (FUNC_ERR_ADD) [offset = 20h]

31	16
UNC_ERR_ADD	
R-u	

15	3	2	0
UNC_ERR_ADD		B_OFF	
R-u		R-u	

LEGEND: R = Read only; -*n* = value after reset; -u = unchanged value on internal reset, cleared on power up

Table 5-20. Flash Uncorrectable Error Address Register (FUNC_ERR_ADD) Field Descriptions

Bit	Field	Value	Description
31-3	UNC_ERR_ADD	0-1FFF FFFFh	Uncorrectable Error Address UNC_ERR_ADD records the CPU logical address of which an uncorrectable error is detected by the ECC logic in the CPU. The UNC_ERR_ADD also captures the error address when a address bus parity mismatch is detected. This error address is frozen from begin updated until it is read by the CPU. Additional error are blocked until this register is read. This register captures the full 32-bit incoming address when there is a bus parity error. It only captures address of 22:3 for multiple bit ECC errors. Address parity errors take priority over other errors that happen in the same cycle.
2-0	B_OFF	0-7h	Byte offset Since ECC is checked on 64 bit data, when checking main memory or OTP, the address captured is aligned to a 64-bit boundary with address bits[2:0] equal to 0.When reading from the ECC bytes, these bits will indicate the failing address of the ECC location associated with the failure. When reading an ECC byte, the ECC is checked against the 64 data bits they protect.

5.7.9 Flash Error Detection and Correction Sector Disable Register (FEDACSDIS)

This register is used to disable the SECDED function for one or two sectors from the EEPROM Emulation Flash (bank 7). An additional two sectors can have SECDED disabled by the use of the FEDACSDIS2 register (see Section 5.7.31).

Figure 5-16. Flash Error Detection and Correction Sector Disable Register (FEDACSDIS) [offset = 24h]

31 29	28	27 24	23 21	20	19 16
BankID1_Inverse	Rsvd	SectorID1_inverse	BankID1	Rsvd	SectorID1
R/WP-0	R-0	R/WP-0	R/WP-0	R-0	R/WP-0

15 13	12	11 8	7 5	4	3 0
BankID0_Inverse	Rsvd	SectorID0_inverse	BankID0	Rsvd	SectorID0
R/WP-0	R-0	R/WP-0	R/WP-0	R-0	R/WP-0

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

Table 5-21. Flash Error Detection and Correction Sector Disable Register (FEDACSDIS) Field Descriptions

Bit	Field	Value	Description
31-29	BankID1_Inverse		The bank ID inverse bits are used with the bank ID bits to select the bank for which a sector is disabled. The only bank that supports sector disable is bank 7.
		0	If BankID1 = 7h and BankID1_inverse = 0, then if a valid sector is selected by SectorID1 and SectorID1_inverse that sector will have ECC checking disabled.
		other	No sector is disabled by disable ID 1.
28	Reserved	0	Reads return 0. Writes have no effect.
27-24	SectorID1_inverse		The sector ID inverse bits are used with the sector ID bits to determine which sector is disabled. If the sector ID bits are not pointing to a valid sector (0-3) or the sector ID inverse bits are not an inverse of the sector ID bits, then no sector is disabled by disable ID 1.
23-21	BankID1		The bank ID bits are used with the bank ID inverse bits to select the bank for which a sector is disabled. The only bank that supports sector disable is bank 7.
		7h	If BankID1 = 7h and BankID1_inverse = 0, then if a valid sector is selected by SectorID1 and SectorID1_inverse that sector will have ECC checking disabled.
		other	No sector is disabled by disable ID 1.
20	Reserved	0	Reads return 0. Writes have no effect.
19-16	SectorID1		The sector ID bits are used with the sector ID inverse bits to determine which sector is disabled. If the sector ID bits are not pointing to a valid sector (0-3) or the sector ID inverse bits are not an inverse of the sector ID bits, then no sector is disabled by disable ID 1.
15-13	BankID0_Inverse		The bank ID inverse bits are used with the bank ID bits to select the bank for which a sector is disabled. The only bank that supports sector disable is bank 7.
		0	If BankID0 = 7h and BankID0_inverse = 0, then if a valid sector is selected by SectorID0 and SectorID0_inverse that sector will have ECC checking disabled.
		other	No sector is disabled by disable ID 0.
12	Reserved	0	Reads return 0. Writes have no effect.
11-8	SectorID0_inverse		The sector ID inverse bits are used with the sector ID bits to determine which sector is disabled. If the sector ID bits are not pointing to a valid sector (0-3) or the sector ID inverse bits are not an inverse of the sector ID bits, then no sector is disabled by disable ID 0.
7-5	BankID0		The bank ID bits are used with the bank ID inverse bits to select the bank for which a sector is disabled. The only bank that supports sector disable is bank 7.
		7h	If BankID0 = 7h and BankID0_inverse = 0, then if a valid sector is selected by SectorID0 and SectorID0_inverse that sector will have ECC checking disabled.
		other	No sector is disabled by disable ID 0.
4	Reserved	0	Reads return 0. Writes have no effect.
3-0	SectorID0	0-Fh	The sector ID bits are used with the sector ID inverse bits to determine which sector is disabled. If the sector ID bits are not pointing to a valid sector (0-3) or the sector ID inverse bits are not an inverse of the sector ID bits, then no sector is disabled by disable ID 0.

5.7.10 Primary Address Tag Register (FPRIM_ADD_TAG)

This register is used to test the pipeline address tag registers (see Section 5.6.2.5).

Figure 5-17. Primary Address Tag Register (FPRIM_ADD_TAG) [offset = 28h]

31	16
PRIM_ADD_TAG[31:16]	
R/WP-0	

15	4	3	0
PRIM_ADD_TAG[15:4]		0000	
R/WP-0		R-0	

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

Table 5-22. Primary Address Tag Register (FPRIM_ADD_TAG) Field Descriptions

Bit	Field	Value	Description
31-4	PRIM_ADD_TAG		Primary Address Tag Register The primary address tag register selected via DIAG_BUF_SEL bits in FDIAGCTRL register is memory mapped here. (see Section 5.7.26) This register can only be written in privileged mode when diagnostic mode is enabled with DIAG_EN_KEY = 5h and DIAG_MODE = 5h. This register will not update with new Flash data if DIAG_EN_KEY is not equal to 5h or DIAG_MODE is 0 or 7h. Valid reads can occur in any mode. The register will clear when an address tag error is found and when leaving DIAG_MODE 5.
3-0		0	Always 0000

5.7.11 Duplicate Address Tag Register (FDUP_ADD_TAG)

This register is used to test the pipeline address tag registers (see Section 5.6.2.5).

Figure 5-18. Duplicate Address Tag Register (FDUP_ADD_TAG) [offset = 2Ch]

31	16
DUP_ADD_TAG[31:16]	
R/WP-0	

15	4	3	0
DUP_ADD_TAG[15:4]		0000	
R/WP-0		R-0	

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

Table 5-23. Duplicate Address Tag Register (FDUP_ADD_TAG) Field Descriptions

Bit	Field	Value	Description
31-4	DUP_ADD_TAG		Primary Address Tag Register The duplicate address tag register selected via DIAG_BUF_SEL bits in FDIAGCTRL register is memory mapped here. (see Section 5.7.26) This register can only be written in privileged mode when diagnostic mode is enabled with DIAG_EN_KEY = 5h and DIAG_MODE = 5h. This register will not update with new Flash data if DIAG_EN_KEY is not equal to 5h or DIAG_MODE is 0 or 7h. Valid reads can occur in any mode. The register will clear when an address tag error is found and when leaving DIAG_MODE 5.
3-0		0	Always 0000

5.7.12 Flash Bank Protection Register (FBPROT)

Figure 5-19. Flash Bank Protection Register (FBPROT) [offset = 30h]

31	16
Reserved	
R-0	

15	1	0
Reserved		PROTL1DIS
R-0		R/WP-0

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

Table 5-24. Flash Bank Protection Register (FBPROT) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	PROTL1DIS		PROTL1DIS: Level 1 Protection Disabled
			Level 1 Protection Disable bit. Setting this bit disables protection from writing to the OTPPROTDIS bits as well as the Sector Enable registers FBSE for all banks. Clearing this bit enables protection and disables write access to the OTPPROTDIS register bits and FBSE register.
		0	Level 1 protection is enabled.
		1	Level 1 protection is disabled.

5.7.13 Flash Bank Sector Enable Register (FBSE)

FBSE provides one enable bit per sector for up to 16 sectors per bank. Each bank in the Flash module has one FBSE register. The bank is selected via the BANK[2:0] bits of the FMAC register (see Section 5.7.20). As only one bank at a time can be selected by FMAC, only the register for the bank selected appears at this address.

Figure 5-20. Flash Bank Sector Enable Register (FBSE) [offset = 34h]

31	16
Reserved	
R-0	

15	0
BSE	
R/WP-0	

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

Table 5-25. Flash Bank Sector Enable Register (FBSE) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	BSE		Bank Sector Enable
			Each bit corresponds to a Flash sector in the bank specified by the FMAC register. Bit 0 corresponds to sector 0, bit 1 corresponds to sector 1, and so on. These bits can be set only when PROTL1DIS = 1 and in privilege mode.
		0	The corresponding numbered sector is disabled for program or erase access.
		1	The corresponding numbered sector is enabled for program or erase access.

5.7.14 Flash Bank Busy Register (FBBUSY)

Figure 5-21. Flash Bank Busy Register (FBBUSY) [offset = 38h]

31	16
Reserved	
R-0	

15	8	7	0
		BUSY[7:0]	
R/WP-0		R-0	

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

Table 5-26. Flash Bank Busy Register (FBBUSY) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	BUSY[7:0]		Bank Busy Each bit corresponds to a Flash bank.
		0	The corresponding bank is not busy.
		1	The corresponding bank is busy with a state machine or bus 2 operation, or the bank is not implemented.

5.7.15 Flash Bank Access Control Register (FBAC)

Figure 5-22. Flash Bank Access Control Register (FBAC) [offset = 3Ch]

31 24	23 16
Reserved	OTPPROTDIS
R-0	R/WP-0

15 8	7 0
BAGP	VREADST
R/WP-0	R/WP-Fh

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

Table 5-27. Flash Bank Access Control Register (FBAC) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23-16	OTPPROTDIS		OTP Sector Protection Disable Each bit corresponds to a Flash bank. This bit can be set only when PROTL1DIS = 1 and in privilege mode.
		0	Programming of the OTP sector is disabled.
		1	Programming of the OTP sector is enabled.
15-8	BAGP	0-FFh	Bank Active Grace Period These bits contain the starting count value for the BAGP down counter. Any access to a given bank causes its BAGP counter to reload the BAGP value for that bank. After the last access to this Flash bank, the down counter delays from 0 to 255 prescaled HCLK clock cycles before putting the bank into one of the fallback power modes as determined by the FBFALLBACK register. This value must be greater than 1 when the fallback mode is not ACTIVE. **Note:** The prescaled clock used for the BAGP down counter is a clock divided by 16 from HCLK.
7-0	VREADST	0-FFh	VREAD Setup VREAD is generated by the Flash pump and used for Flash read operation. The bank power up sequencing starts VREADST HCLK cycles after VREAD power supply becomes stable. **Note**: There is not a programmable Bank Sleep counter and Standby counter register. The number of clock cycles to transition from sleep to standby and standby to active is hardcoded in the Flash wrapper design.

5.7.16 Flash Bank Fallback Power Register (FBFALLBACK)

Figure 5-23. Flash Bank Fallback Power Register (FBFALLBACK) [offset = 40h]

31	16
Reserved	
R-0505h	

15	14	13	4	3	2	1	0
BANKPWR7		Reserved		BANKPWR1		BANKPWR0	
R/WP-3h		R/WP-3FFh		R/WP-3h		R/WP-3h	

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

Table 5-28. Flash Bank Fallback Power Register (FBFALLBACK) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0505h	Do not write to these bits.
15-14	BANKPWR7		Bank 7 Fallback Power Mode
		0	Bank sleep mode
		1h	Bank standby mode
		2h	Reserved
		3h	Bank active mode
13-4	Reserved	3FFh	Do not write to these bits.
3-2	BANKPWR1		Bank 1 Fallback Power Mode
		0	Bank sleep mode
		1h	Bank standby mode
		2h	Reserved
		3h	Bank active mode
1-0	BANKPWR0		Bank 0 Fallback Power Mode
		0	Bank sleep mode
		1h	Bank standby mode
		2h	Reserved
		3h	Bank active mode

5.7.17 Flash Bank/Pump Ready Register (FBPRDY)

Figure 5-24. Flash Bank/Pump Ready Register (FBPRDY) [offset = 44h]

31	24	23	16
Reserved		BANKBUSY[7:0]	
R-0		R-1 (Unimplemented banks) or R-0 (Implemented banks)	

15	14	8	7	0
PUMPRDY	Reserved		BANKRDY[7:0]	
R-1	R-0		R-1	

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

Table 5-29. Flash Pump Access Control Register 1 (FPAC1) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23-16	BANKBUSY[7:0]		Bank busy bits (one bit for each bank)
		0	The bank is not busy.
		1	The bank is busy, not ready or this bank is not implemented.
			Note: A bank is considered busy if it is being accessed by the TCM, Bus 2 or the Flash state machine.
15	PUMPRDY		Flash pump ready flag
		0	Pump is not ready (Code must be executing from somewhere other than internal Flash).
		1	Pump is ready for Flash accesses.
14-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	BANKRDY[7:0]		Bank ready bits (one bit for each bank)
		0	Flash bank is in the sleep or standby state.
		1	Flash bank is in the active state, or the bank is not implemented.

5.7.18 Flash Pump Access Control Register 1 (FPAC1)

Figure 5-25. Flash Pump Access Control Register 1 (FPAC1) [offset = 48h]

31 — 27	26 — 16
Reserved	PSLEEP
R-0	R/WP-64h

15 — 1	0
Reserved	PUMPPWR
R-0	R/WP-1

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

Table 5-30. Flash Pump Access Control Register 1 (FPAC1) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reads return 0. Writes have no effect.
26-16	PSLEEP		Pump Sleep These bits contain the starting count value for the charge pump sleep down counter. While the charge pump is in sleep mode, the power mode management logic holds the charge pump sleep counter at this value. When the charge pump exits sleep power mode, the down counter delays from 0 to PSLEEP pump sleep down clock cycles before putting the charge pump into active power mode. **Note:** Pump sleep down counter clock is a divide by 2 input of HCLK. That is, there are 2*HCLK cycles for every PSLEEP counter cycle.
15-1	Reserved	0	Reads return 0. Writes have no effect.
0	PUMPPWR		Flash Charge Pump Fallback Power Mode
		0	Sleep (all pump circuits disabled).
		1	Active (all pump circuits active).

5.7.19 Flash Pump Access Control Register 2 (FPAC2)

Figure 5-26. Flash Pump Access Control Register 2 (FPAC2) [offset = 4Ch]

31	16
Reserved	
R-0	

15	0
PAGP	
R/WP-0	

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

Table 5-31. Flash Pump Access Control Register 2 (FPAC2) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	PAGP		Pump Active Grace Period This register contains the starting count value for the PAGP mode down counter. Any access to Flash memory causes the counter to reload with the PAGP value. After the last access to Flash memory, the down counter delays from 0 to 65535 prescaled HCLK clock cycles before entering one of the charge pump fallback power modes as determined by PUMPPWR in the FPAC1 register. **Note:** The PAGP down counter is clocked by the same prescaled clock as the BAGP down counter which is a divide by 16 of HCLK.

5.7.20 Flash Module Access Control Register (FMAC)

Figure 5-27. Flash Module Access Control Register (FMAC) [offset = 50h]

31	16
Reserved	
R-0	

15	3	2	0
Reserved		BANK	
R-0		R/WP-0	

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

Table 5-32. Flash Module Access Control Register (FMAC) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
2-0	BANK		Bank Enable. These bits select which bank is enabled for operations such as local register access, OTP sector access, and program/erase commands. These bits select only one bank at a time from up to eight banks depending on the specific device being used. For example, a 000 selects bank 0; 011 selects Bank 3. **Note:** BANK[2:0] can identify up to 8 Flash banks. If BANK[2:0] is selected for an unimplemented bank then the BANK[2:0] will set itself to the number of an implemented bank. To determine if a bank is implemented, write the bank number to BANK[2:0] and read back the value to see if what was written can be read back.

5.7.21 Flash Module Status Register (FMSTAT)

Figure 5-28. Flash Module Status Register (FMSTAT) [offset = 54h]

31							24
Reserved							
R-0							

23							16
Reserved							
R-0							

15	14	13	12	11	10	9	8
Reserved	ILA	Reserved	PGV	Reserved	EV	Reserved	BUSY
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

7	6	5	4	3	2	1	0
ERS	PGM	INVDAT	CSTAT	VOLTSTAT	ESUSP	PSUSP	SLOCK
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 5-33. Flash Module Status Register (FMSTAT) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved	0	Reads return 0. Writes have no effect.
14	ILA		Illegal Address When set, indicates that an illegal address is detected. Five conditions can set the illegal address flag. 1. Writing to a hole (unimplemented logical address space) within a Flash bank. 2. Writing to an address location to an unimplemented Flash space. 3. Input address for write is decoded to select a different bank from the bank ID register. 4. The address range does not match the type of FSM command. For example, the erase_sector command must match the address regions. 5. TI-OTP address selected but CMD_EN in FSM_ST_MACHINE is not set.
13	Reserved	0	Reads return 0. Writes have no effect.
12	PGV		Program Verify When set, indicates that a word is not successfully programmed after the maximum allowed number of program pulses are given for program operation.
11	Reserved	0	Reads return 0. Writes have no effect.
10	EV		Erase Verify When set, indicates that a sector is not successfully erased after the maximum allowed number of erase pulses are given for erase operation. During Erase verify command, this flag is set immediately if a bit is found to be 0.
9	Reserved	0	Reads return 0. Writes have no effect.
8	BUSY		Busy When set, this bit indicates that a program, erase, or suspend operation is being processed.
7	ERS		Erase Active When set, this bit indicates that the Flash module is actively performing an erase operation. This bit is set when erasing starts and is cleared when erasing is complete. It is also cleared when the erase is suspended and set when the erase resumes.
6	PGM		Program Active When set, this bit indicates that the Flash module is currently performing a program operation. This bit is set when programming starts and is cleared when programming is complete. It is also cleared when programming is suspended and set when programming is resumes.
5	INVDAT		Invalid Data When set, this bit indicates that the user attempted to program a 1 where a 0 was already present. This bit is cleared by the Clear Status command.

Table 5-33. Flash Module Status Register (FMSTAT) Field Descriptions (continued)

Bit	Field	Value	Description
4	CSTAT		Command Status Once the FSM starts any failure will set this bit. When set, this bit informs the host that the program, erase, or validate sector command failed and the command was stopped. This bit is cleared by the Clear Status command. For some errors, this will be the only indication of an FSM error because the cause does not fall within the other error bit types.
3	VOLTSTAT		Core Voltage Status When set, this bit indicates that the core voltage generator of the pump power supply dipped below the lower limit allowable during a program or erase operation. This bit is cleared by the Clear Status command.
2	ESUSP		Erase Suspended When set, this bit indicates that the Flash module has received and processed an erase suspend operation. This bit remains set until the erase resume command has been issued or until the Clear_More command is run.
1	PSUSP		Program Suspended When set, this bit indicates that the Flash module has received and processed a program suspend operation. This bit remains set until the program resume command has been issued or until the Clear_More command is run.
0	SLOCK		Sector Lock Status When set, this bit indicates that the operation was halted because the target sector was locked for erasing and programming either by the sector protect bit or by OTP write protection disable bits. (Bits BSE in FBSE register or OTPPROTDIS in register FBAC). This bit is cleared by the Clear Status command. No SLOCK FSM error will occur if all sectors in a bank erase operation are set to 1. All the sectors will be checked but no SLOCK will be set if no operation occurs due to the SECT_ERASED bits being set to all 1s. A SLOCK error will occur if attempting to do a sector erase with either BSE is cleared or SECT_ERASED is set.

5.7.22 EEPROM Emulation Data MSW Register (FEMU_DMSW)

The Flash module controller includes hardware support computing the check bits for ECC, based on the data and address being programmed into the EEPROM emulation array. To utilize this capability, the address to be programmed is written to the FEMU_ADDR register and the data to be programmed is written to the FEMU_DMSW and FEMU_DLSW registers. The write to FEMU_DLSW triggers an ECC calculation and the resulting check bits are available in the FEMU_ECC register. The value from FEMU_ECC can then be used to program the check bits into the EEPROM emulation array for the particular data word over which they were calculated.

Figure 5-29. EEPROM Emulation Data MSW Register (FEMU_DMSW) [offset = 58h]

31 … 0
EMU_DMSW[63:32]
R/WP-0

LEGEND: R/W = Read/Write; WP = Write in Privilege mode; -*n* = value after reset

Table 5-34. EEPROM Emulation Data MSW Register (FEMU_DMSW) Field Descriptions

Bit	Field	Description
31-0	EMU_DMSW	EEPROM Emulation Most-Significant Data Word. The most-significant data word of the 64-bits of data for which the ECC check bits are to be calculated should be programmed into this register. This register is also used in diagnostic modes 1 and 2 where it supplies the upper data for checking the SECDED hardware.

5.7.23 EEPROM Emulation Data LSW Register (FEMU_DLSW)

The Flash module controller includes hardware support computing the check bits for ECC, based on the data and address being programmed into the EEPROM emulation array. To utilize this capability, the address to be programmed is written to the FEMU_ADDR register and the data to be programmed is written to the FEMU_DMSW and FEMU_DLSW registers. The write to FEMU_DLSW triggers an ECC calculation and the resulting check bits are available in the FEMU_ECC register. The value from FEMU_ECC can then be used to program the check bits into the EEPROM emulation array for the particular data word over which they were calculated.

Figure 5-30. EEPROM Emulation Data LSW Register (FEMU_DLSW) [offset = 5Ch]

31 … 0
EMU_DLSW[31:0]
R/WP-0

LEGEND: R/W = Read/Write; WP = Write in Privilege mode; -*n* = value after reset

Table 5-35. EEPROM Emulation Data LSW Register (FEMU_DLSW) Field Descriptions

Bit	Field	Description
31-0	EMU_DLSW	EEPROM Emulation Least-Significant Data Word. The least-significant data word of the 64-bits of data for which the ECC check bits are to be calculated should be programmed into this register. This register is also used in diagnostic modes 1 and 2 where it supplies the lower data for checking the SECDED hardware.

5.7.24 EEPROM Emulation ECC Register (FEMU_ECC)

The Flash module controller includes hardware support computing the check bits for ECC, based on the data and address being programmed into the EEPROM emulation array. To utilize this capability, the address to be programmed is written to the FEMU_ADDR register and the data to be programmed is written to the FEMU_DMSW and FEMU_DLSW registers. The write to FEMU_DLSW triggers an ECC calculation and the resulting check bits are available in the FEMU_ECC register. The value from FEMU_ECC can then be used to program the check bits into the EEPROM emulation array for the particular data word over which they were calculated.

Figure 5-31. EEPROM Emulation ECC Register (FEMU_ECC) [offset = 60h]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		EMU_ECC[7:0]	
R-0		R/WP-3h	

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

Table 5-36. EEPROM Emulation ECC Register (FEMU_ECC) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	EMU_ECC[7:0]		EEPROM Emulation ECC Check Bit Value. This register contains the ECC check bits calculated by the FMC controller based on the address written to FEMU_ADDR and the 64-bits of data written to FEMU_DMSW and FEMU_DLSW. This register is also used in the diagnostic modes 1 and 2. In these modes, this register supplies the ECC data for checking the SECDED. In mode 1, this register is filled with the desired ECC and after the DIAG_TRIG is set this register is set to the ECC value returned from the SECDED selected by DIAG_ECC_SEL. DIAG_EN_KEY and DIAG_TRIG must be set to fill this register in the diagnostic modes. Writes to FEMU_DxSW will not affect this register in diagnostic modes 1 or 2. In mode 2, this register is filled with the desired ECC and after the DIAG_TRIG is set this register is set to the syndrome value returned from the SECDED selected by DIAG_ECC_SEL. This register is only available when the module is configured to use the ECC logic; otherwise, it is a reserved register.

5.7.25 EEPROM Emulation Address Register (FEMU_ADDR)

The Flash module controller includes hardware support computing the check bits for ECC, based on the data and address being programmed into the EEPROM emulation array. To utilize this capability, the address to be programmed is written to the FEMU_ADDR register and the data to be programmed is written to the FEMU_DMSW and FEMU_DLSW registers. The write to FEMU_DLSW triggers an ECC calculation and the resulting check bits are available in the FEMU_ECC register. The value from FEMU_ECC can then be used to program the check bits into the EEPROM emulation array for the particular data word over which they were calculated.

Figure 5-32. EEPROM Emulation Address Register (FEMU_ADDR) [offset = 68h]

31 22	21 16
Reserved	EMU_ADDR[21:16]
R-0	R/WP-0

15 3	2 0
EMU_ADDR[15:3]	Reserved
R/WP-0	R-0

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

Table 5-37. EEPROM Emulation Address Register (FEMU_ADDR) Field Descriptions

Bit	Field	Value	Description
31-22	Reserved	0	Reserved. Writes have no effect. It is not necessary to mask these upper ten bits when writing the address of bank 7 locations (0xF002xxxx); however, these bits are not used in calculating the ECC value and will read back as 0.
21-3	EMU_ADDR	0-7 FFFFh	EEPROM Emulation Address. The address of the 64-bit data word over which ECC is to be calculated is written to this field. Note that only bits 21:3 are actually written and used for the calculation. The other bits (31:22 and 2:0) are ignored, but do not need to be masked off before being written to this register. This register is also used in diagnostic modes 1 and 2 where it supplies the address bits for checking the SECDED hardware.
2-0	Reserved	0	Reserved. Writes have no effect. The lower three bits of the CPU address are not used in the ECC calculation to align the data on a 64-bit boundary. These bits will read back as 0.

5.7.26 Diagnostic Control Register (FDIAGCTRL)

Set the DIAG_MODE and the DIAG_EN_KEY first before setting up the other registers to block the other registers from causing a false error. The final write should set the DIAG_TRIG register to activate the test. Running out of ram will prevent problems with the diagnostic test corrupting the Flash access in some of the modes.

Figure 5-33. Diagnostic Control Register (FDIAGCTRL) [offset = 6Ch]

31 25	24	23 20	19 16
Reserved	DIAG_TRIG	Reserved	DIAG_EN_KEY
R-0	R/WP-0	R-0	R/WP-Ah

15	14 12	11 10	9 8	7 3	2 0
Rsvd	DIAG_ECC_SEL	Reserved	DIAG_BUF_SEL	Reserved	DIAG_MODE
R-0	R/WP-0	R-0	R/WP-0	R-0	R/WP-0

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

Table 5-38. Diagnostic Control Register (FDIAGCTRL) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reserved
24	DIAG_TRIG		Diagnostic Trigger Diagnostic trigger is the final qualifier for the diagnostic result. After setting all the other diagnostic register values, the DIAG_TRIG is set to 1. This will activate the diagnostic logic for one access and then automatically clear the DIAG_TRIG value. DIAG_EN_KEY and DIAG_MODE must be set at least one cycle before setting DIAG_TRIG. This bit always reads as 0.
23-20	Reserved	0	Reserved
19-16	DIAG_EN_KEY		Diagnostic Enable Key
		5h	Diagnostic mode is enabled.
		All other values	Diagnostic mode is disabled.
15	Reserved	0	Reserved
14-12	DIAG_ECC_SEL		Diagnostic SECDED Select
		0	Select SECDED0 for diagnostic testing.
		1h	Select SECDED1 for diagnostic testing.
		2h	Select SECDED2 for diagnostic testing (256 bit wide words only).
		3h	Select SECDED3 for diagnostic testing (256 bit wide words only).
		4h	Select BUS2 SECDED for diagnostic testing.
		5h	Select FEE SECDED for diagnostic testing (Same ECC logic as BUS2 but sets FEE registers).
		6h-7h	Reserved
11-10	Reserved	0	Reserved

Table 5-38. Diagnostic Control Register (FDIAGCTRL) Field Descriptions (continued)

<table>
<tr><th>Bit</th><th>Field</th><th>Value</th><th>Description</th></tr>
<tr><td>9-8</td><td>DIAG_BUF_ SEL</td><td>
0</td><td>Diagnostic Buffer Select
The DIAG_BUF_SEL selects the Instruction or Data buffer to read or write when accessing the FPRIM_ADD_TAG and FDUP_ADD_TAG registers. The address tags consists of matching primary and duplicate address tag registers. All the primary address tag registers are memory mapped to a common address (see Section 5.7.10) and are selected by DIAG_BUF_SEL. The same occurs for the duplicate address. (see Section 5.7.11). Bit 0 selects a data buffer if high and an instruction buffer if low. Bits 1 indicate the buffer number.
DIAG_BUF_SEL ENCODING:
<table>
<tr><th>Buffer Number Bits 1</th><th>Inst=0 Data=1 Bit0</th><th>Buffer</th></tr>
<tr><td>0</td><td>0</td><td>Instruction Buffer 0</td></tr>
<tr><td>0</td><td>1</td><td>Data Buffer 0</td></tr>
<tr><td>1</td><td>0</td><td>Instruction Buffer 1</td></tr>
<tr><td>1</td><td>1</td><td>Data Buffer 1</td></tr>
</table></td></tr>
<tr><td>7-4</td><td>Reserved</td><td>0</td><td>Reserved</td></tr>
<tr><td>2-0</td><td>DIAG_MODE</td><td></td><td>Diagnostic Mode</td></tr>
<tr><td></td><td></td><td>0</td><td>Diagnostic mode is disabled. This is the same as DIAG_EN_KEY is not equal to 5h.</td></tr>
<tr><td></td><td></td><td>1h</td><td>Diagnostic ECC Data Correction test mode (see Section 5.6.2.1).</td></tr>
<tr><td></td><td></td><td>2h</td><td>Diagnostic ECC Syndrome Reporting test mode (see Section 5.6.2.2).</td></tr>
<tr><td></td><td></td><td>3h</td><td>ECC Malfunction test mode 1 (same data) (see Section 5.6.2.3).</td></tr>
<tr><td></td><td></td><td>4h</td><td>ECC Malfunction test mode 2 (inverted data) (see Section 5.6.2.4).</td></tr>
<tr><td></td><td></td><td>5h</td><td>Address Tag Register test mode (see Section 5.6.2.5).</td></tr>
<tr><td></td><td></td><td>6h</td><td>Reserved</td></tr>
<tr><td></td><td></td><td>7h</td><td>ECC Data Correction Diagnostic test mode (see Section 5.6.2.6).</td></tr>
</table>

5.7.27 Uncorrected Raw Data High Register (FRAW_DATAH)

Figure 5-34. Uncorrected Raw Data High Register (FRAW_DATAH) [offset = 70h]

31	0
RAW_DATA[63:32]	
R/WP-u	

LEGEND: R/W = Read/Write; WP = Write in Privilege mode; -*n* = value after reset; -u = unchanged value on internal reset, cleared on power up

Table 5-39. Uncorrected Raw Data High Register (FRAW_DATAH) Field Descriptions

Bit	Field	Description
31-0	RAW_DATA [63:32]	Uncorrected Raw Data This register contains the upper 32 bits of the 64-bit raw data used in diagnostic testing of the ECC logic. **NOTE:** Raw Data and Raw ECC registers can be loaded with diagnostic values only in diagnostic modes 1 through 6 with DIAG_EN_KEY=0101. These modes must be set for at least one clock cycle before writing to any FRAW* register.

5.7.28 Uncorrected Raw Data Low Register (FRAW_DATAL)

Figure 5-35. Uncorrected Raw Data Low Register (FRAW_DATAL) [offset = 74h]

31	0
RAW_DATA[31:0]	
R/WP-u	

LEGEND: R/W = Read/Write; WP = Write in Privilege mode; -*n* = value after reset -u = unchanged value on internal reset, cleared on power up

Table 5-40. Uncorrected Raw Data Low Register (FRAW_DATAL) Field Descriptions

Bit	Field	Description
31-0	RAW_DATA [31:0]	Uncorrected Raw Data. This register contains the lower 32 bits of the 64-bit raw data used in diagnostic testing of the ECC logic. **NOTE:** Raw Data and Raw ECC registers can be loaded with diagnostic values only in diagnostic modes 1 through 6 with DIAG_EN_KEY=0101. These modes must be set for at least one clock cycle before writing to any FRAW* register.

5.7.29 Uncorrected Raw ECC Register (FRAW_ECC)

Figure 5-36. Uncorrected Raw ECC Register (FRAW_ECC) [offset = 78h]

31	16
Reserved	
R-0	

15	9	8	7	0
Reserved		PIPE BUF	RAW_ECC[7:0]	
R-0		RC-0	R/WD-u	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in Privilege mode; C = Clear by writing a 1; -*n* = value after reset -u = unchanged value on internal reset, cleared on power up

Table 5-41. Uncorrected Raw ECC Register (FRAW_ECC) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reserved
8	PIPE BUF		Error came from pipeline buffer hit When this bit is a 1, latest error came from a pipeline buffer hit and the FRAW_DATH, FRAW_DATL, and RAW_ECC fields will not contain information that matches the error address nor error status bits. This bit is cleared when the RAW_ECC field is updated with new valid information or by writing a 1 to this bit.
7-0	RAW_ECC[7:0]		Uncorrected Raw ECC This register contains the ECC data used in diagnostic testing of the ECC logic. **NOTE:** Raw Data and Raw ECC registers can loaded with diagnostic values only in a used diagnostic mode with DIAG_EN_KEY=0101. This mode must be set for at least one clock cycle before writing to any FRAW* register.

5.7.30 Parity Override Register (FPAR_OVR)

Figure 5-37. Parity Override Register (FPAR_OVR) [offset = 7Ch]

31 17	16
Reserved	BNK_INV_PAR
R-0	R/WP-0

15 12	11 9	8	7 0
BUS_PAR_DIS	PAR_OVR_KEY	ADD_INV_PAR	DAT_INV_PAR
R/WP-5h	R/WP-2h	R/WP-0	R/WP-0

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

Table 5-42. Parity Override Register (FPAR_OVR) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reserved
16	BNK_INV_PAR		Buffer Invert Parity When this value is 1 and PAR_OVR_KEY is 101 then the current system parity signal SYS_ODD_PARITY is inverted when doing bank parity calculations. This will generate parity errors and cause interrupt signals to be generated. When 0, the SYS_ODD_PARITY value is used. This bit is only implemented for parity configurations and is reserved for ECC devices.
15-12	BUS_PAR_DIS		Disable Bus Parity When this value is 1010, the address bus parity error and buffer parity error are disabled and no checking is done and no events are generated. Any other value will enable the parity checking on the Address bus and read data bus. The read data parity is never disabled from this module.
11-9	PAR_OVR_KEY		When this value is 101, the selected ADD_INV_PAR and DAT_INV_PAR fields will become active. Any other value will cause the module to use the global system parity bit in the system register DEVCR1.
8	ADD_INV_PAR		Address Odd Parity This bit is active only when PAR_OVR_KEY = 101. When ADD_INV_PAR is 1, the incoming address bus will invert the system signal SYS_ODD_PARITY for parity calculations. This will cause parity errors and generate interrupt error signals. When 0, it will use the SYS_ODD_PARITY value. This bit is set to the SYS_ODD_PAR signal value on reset.
7-0	DAT_INV_PAR		Data Odd Parity This byte is active only when PAR_OVR_KEY = 101. When a DATA_INV_PAR bit is 1, the output read data will invert the system signal SYS_ODD_PARITY for parity calculations. This will cause parity errors and generate interrupt signals. When 0, it will use the SYS_ODD_PARITY value. This byte can support up to a 64 bit data bus but when the device has a 32 bit bus, bits 7:4 are reserved. Bit 0 affects read bus bits 7:0, Bit 1 affects read bus bits 15:8 and so on. Each active bit of this field is set to the SYS_ODD_PAR signal value on reset. The DAT_INV_PAR is used in the parity for the pipeline buffer logic and for the read data bus to the CPU. When the ECC logic is in the CPU (CONF_TYPE = 5) and SIL3 is active, this field becomes the ECC corrupting value for SIL3 diagnostic mode 7. (Starting with version 1.0.0.0) In diagnostic mode 7 the FPAR_OVR should be set to 00005Axxh to allow writes to the DAT_INV_PAR field. This field should be written before entering diagnostic mode 7.

5.7.31 Flash Error Detection and Correction Sector Disable Register (FEDACSDIS2)

This register is used to disable the SECDED function for one or two sectors from the EEPROM Emulation Flash (bank 7). An additional two sectors can have SECDED disabled by the use of the FEDACSDIS register. see Section 5.7.9.

Figure 5-38. Flash Error Detection and Correction Sector Disable Register (FEDACSDIS2) [offset = C0h]

31 29	28	27 24	23 21	20	19 16
BankID3_Inverse	Rsvd	SectorID3_inverse	BankID3	Rsvd	SectorID3
R/WP-0	R-0	R/WP-0	R/WP-0	R-0	R/WP-0

15 13	12	11 8	7 5	4	3 0
BankID2_Inverse	Rsvd	SectorID2_inverse	BankID2	Rsvd	SectorID2
R/WP-0	R-0	R/WP-0	R/WP-0	R-0	R/WP-0

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

Table 5-43. Flash Error Detection and Correction Sector Disable Register (FEDACSDIS2) Field Descriptions

Bit	Field	Value	Description
31-29	BankID3_Inverse		The bank ID inverse bits are used with the bank ID bits to select the bank for which a sector is disabled. The only bank that supports sector disable is bank 7.
		0	If BankID3 = 7h and BankID3_inverse = 0, then if a valid sector is selected by SectorID3 and SectorID3_inverse that sector will have ECC checking disabled.
		other	No sector is disabled by disable ID 3.
28	Reserved	0	Reads return 0. Writes have no effect.
27-24	SectorID3_inverse		The sector ID inverse bits are used with the sector ID bits to determine which sector is disabled. If the sector ID bits are not pointing to a valid sector (0-3) or the sector ID inverse bits are not an inverse of the sector ID bits, then no sector is disabled by disable ID 3.
23-21	BankID3		The bank ID bits are used with the bank ID inverse bits to select the bank for which a sector is disabled. The only bank that supports sector disable is bank 7.
		7h	If BankID3 = 7h and BankID3_inverse = 0, then if a valid sector is selected by SectorID3 and SectorID3_inverse that sector will have ECC checking disabled.
		other	No sector is disabled by disable ID 3.
20	Reserved	0	Reads return 0. Writes have no effect.
19-16	SectorID3		The sector ID bits are used with the sector ID inverse bits to determine which sector is disabled. If the sector ID bits are not pointing to a valid sector (0-3) or the sector ID inverse bits are not an inverse of the sector ID bits, then no sector is disabled by disable ID 3.
15-13	BankID2_Inverse		The bank ID inverse bits are used with the bank ID bits to select the bank for which a sector is disabled. The only bank that supports sector disable is bank 7.
		0	If BankID2 = 7h and BankID2_inverse = 0, then if a valid sector is selected by SectorID2 and SectorID2_inverse that sector will have ECC checking disabled.
		other	No sector is disabled by disable ID 2.
12	Reserved	0	Reads return 0. Writes have no effect.
11-8	SectorID2_inverse		The sector ID inverse bits are used with the sector ID bits to determine which sector is disabled. If the sector ID bits are not pointing to a valid sector (0-3) or the sector ID inverse bits are not an inverse of the sector ID bits, then no sector is disabled by disable ID 2.
7-5	BankID2		The bank ID bits are used with the bank ID inverse bits to select the bank for which a sector is disabled. The only bank that supports sector disable is bank 7.
		7h	If BankID2 = 7h and BankID2_inverse = 0, then if a valid sector is selected by SectorID2 and SectorID2_inverse that sector will have ECC checking disabled.
		other	No sector is disabled by disable ID 2.
4	Reserved	0	Reads return 0. Writes have no effect.
3-0	SectorID2	0-Fh	The sector ID bits are used with the sector ID inverse bits to determine which sector is disabled. If the sector ID bits are not pointing to a valid sector (0-3) or the sector ID inverse bits are not an inverse of the sector ID bits, then no sector is disabled by disable ID 2.

5.7.32 FSM Register Write Enable (FSM_WR_ENA)

Figure 5-39. FSM Register Write Enable (FSM_WR_ENA) [offset = 288h]

31-16
Reserved
R-0

15-3	2-0
Reserved	WR_ENA
R-0	R/WP-2h

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

Table 5-44. FSM Register Write Enable (FSM_WR_ENA) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
2-0	WR_ENA		Flash State Machine Write Enable
		5h	This register must contain 101 in order to write to any other register in the range FFF8 7200h to FFF8 72FFh. This is the first register to be written when setting up the FSM.
		All other values	For all other values, the FSM registers cannot be written.

5.7.33 FSM Sector Register (FSM_SECTOR)

This is a banked register. A separate register is implemented for each bank, but they all occupy the same address. The correct bank must be selected in the FMAC register before reading or writing this register. See Section 5.7.20.

Figure 5-40. FSM Sector Register (FSM_SECTOR) [offset = 2A4h]

31-16
SECT_ERASED[16:0]
R/WP-0

15-0
Reserved
R-0

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

Table 5-45. FSM Sector Register (FSM_SECTOR) Field Descriptions

Bit	Field	Value	Description
31-16	SECT_ERASED[16:0]		Sectors Erased
			There is one bit for each sector. Bit 16 corresponds to sector 0, bit 17 corresponds to sector 1, and so on. After bank erase, the bit corresponding to each sector which is erased will be changed from 0 to 1.
		0	During bank erase, each sector will be erased.
		1	Each sector will not be erased.
15-0	Reserved	0	These bits are used by the state machine during bank erase. Do not write to these bits.

5.7.34 EEPROM Emulation Configuration Register (EEPROM_CONFIG)

Figure 5-41. EEPROM Emulation Configuration Register (EEPROM_CONFIG) [offset = 2B8h]

31-20	19-16
Reserved	EWAIT
R-0	R/WP-1h

15-9	8	7-0
Reserved	AUTOSUSP_EN	AUTOSTART_GRACE
R-0	R/WP-0	R/WP-2h

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

Table 5-46. EPROM Emulation Configuration Register (EEPROM_CONFIG) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	EWAIT		EEPROM Wait State Counter This register will replace the RWAIT count in the EEPROM register. The same formulas that apply to RWAIT will apply to EWAIT in the EEPROM bank.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	AUTOSUSP_EN		Auto Suspend Enable
		0	Auto Suspend is disabled.
		1	Auto Suspend is enabled.
			The auto-suspend will begin when the CPU or Bus 2 attempts to access a bank with an active and suspendable FSM operation. If this happens the FSM will automatically be issued a suspend command and exit from the FSM. It will then do the access. After the access, the FMC will wait for a time determined by the Autostart_grace field before issuing the FSM resume command.
7-0	AUTOSTART_GRACE		Auto-suspend Startup Grace Period
		1	The value in this register determines how many cycles the FMC will wait after the last CPU or Bus 2 access before issuing the FSM resume command.
		0	The FMC will wait 16 HCLK periods for each count in the AUTOSTART_GRACE field. A value of 2 will wait for 32 periods after the last access. Each access will reset the counter to the AUTOSTART_GRACE value × 16.

5.7.35 EEPROM Emulation Error Detection and Correction Control Register 1 (EE_CTRL1)

This register controls ECC event detection for the EEPROM Emulation Flash bank (bank 7). For the equivalent register that controls ECC event detection for the main Flash banks, see Section 5.7.2.

Figure 5-42. EEPROM Emulation Error Detection and Correction Control Register 1 (EE_CTRL1) [offset = 308h]

31	24
Reserved	
R-05h	

23	20	19	16
Reserved		EE_EDACMODE	
R-0		R/WP-Ah	

15	10	9	8
Reserved	EE_EOFEN	EE_EZFEN	EE_EPEN
R-0	R/WP-0	R/WP-0	R/WP-0

7	6	5	4	3	0
Reserved		EE_ALL1_OK	EE_ALL0_OK	EE_EDACEN	
R-0		R/WP-0	R/WP-0	R/WP-5h	

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

Table 5-47. EEPROM Emulation Error Detection and Correction Control Register 1 (EE_CTRL1) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	050h	Reads return 050h. Writes have no effect.
19-16	EE_ EDACMODE		Error Correction Mode for the EEPROM Emulation Flash bank (bank 7). For the main Flash banks, see Section 5.7.2.
		5h	Detection only mode. **Note:** In detection only mode single-bit errors will not be corrected, but will be treated as uncorrectable errors. The single-bit error flags and profiling mode are disabled. Detection only mode has the advantage that a triple bit error will be detected and not mistaken for a single-bit error and mis-corrected.
		All Other Values	Single-bit errors are corrected and multi-bit or address errors are detected. **Note:** It is recommended to leave the EE_EDACMODE field as 1010 to guard against soft errors from flipping EE_EDACMODE to a detection only.
15-11	Reserved	0	Reads return 0. Writes have no effect.
10	EE_EOFEN		EEPROM Emulation Event on a correctable One's Fail Enable bit
		0	No ESM event will be generated on a single-bit error when a 1 reads as a 0 and is corrected.
		1	An ESM group 1 channel 35 event will be generated on a single-bit error when a 1 reads as a 0 and is corrected.
9	EE_EZFEN		EEPROM Emulation Event on a correctable Zero's Fail Enable bit
		0	No ESM event will be generated on a single-bit error when a 0 reads as a 1 and is corrected.
		1	An ESM group 1 channel 35 event will be generated on a single-bit error when a 0 reads as a 1 and is corrected.
8	EE_EPEN		EEPROM Emulation Error Profiling Enable
		0	Error profiling is disabled.
		1	Error profiling is enabled. An ESM group 1 channel 35 event will be generated when number of correctable bit errors detected and corrected has reached the threshold value defined in the EE_CTRL2 register.
7-6	Reserved	0	Reads return 0. Writes have no effect.

Table 5-47. EEPROM Emulation Error Detection and Correction Control Register 1 (EE_CTRL1) Field Descriptions (continued)

Bit	Field	Value	Description
5	EE_ALL1_OK		EEPROM Emulation All One Condition Valid
		0	One condition valid is disabled.
			Reading of an erased location (64 data bits and the corresponding 8 ECC bits are all 1s) will generate ECC errors. The error counter for profiling will increment if all 1s are detected.
		1	One condition valid is enabled.
			Reading of an erased location (64 data bits and the corresponding 8 ECC bits are all 1s) will NOT generate ECC errors. The error counter for profiling will NOT increment if all 1s are detected.
4	EE_ALL0_OK		EEPROM Emulation All Zero Condition Valid
		0	Zero condition valid is disabled.
			Reading of all 0s (64 data bits and the corresponding 8 ECC bits are all 0s) will generate ECC errors. The error counter for profiling will increment if all 0s are detected.
		1	Zero condition valid is enabled.
			Reading of all 0s (64 data bits and the corresponding 8 ECC bits are all 0s) will NOT generate ECC errors. The error counter for profiling will NOT increment if all 0s are detected.
3-0	EE_EDACEN		EEPROM Emulation Error Detection and Correction Enable
		5h	Error Detection and Correction is disabled.
		All Other Values	Error Detection and Correction is enabled.
			Note: It is recommended to leave the EE_EDACEN field as 1010 to guard against soft errors from flipping the EE_EDACEN to a disabled state.

5.7.36 EEPROM Emulation Error Correction and Correction Control Register 2 (EE_CTRL2)

Figure 5-43. EEPROM Emulation Error Correction and Correction Control Register 2 (EE_CTRL2) [offset = 30Ch]

31 16
Reserved
R-0

15 0
EE_SEC_THRESHOLD
R/WP-0

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

Table 5-48. EEPROM Emulation Error Correction Control Register 2 (EE_CTRL2) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	EE_SEC_THRESHOLD	0-FFFFh	EEPROM Emulation Single Error Correction Threshold This register contains the threshold value for the SEC (single error correction) occurrences before a single interrupt request is generated. A threshold of zero disables the threshold so that it never triggers the profile interrupt.

5.7.37 EEPROM Emulation Correctable Error Count Register (EE_COR_ERR_CNT)

Figure 5-44. EEPROM Emulation Error Correctable Error Count Register (EE_COR_ERR_CNT) [offset = 310h]

31 16
Reserved
R-0

15 0
EE_ERRCNT
R/WP-0

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

Table 5-49. EEPROM Emulation Correctable Error Count Register (EE_COR_ERR_CNT) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	EE_ERRCNT	0-FFFFh	Single Error Correction Count This register contains the number of SEC (single error correction) occurrences. Writing any value to this register resets the count value to 0. The counter resets to 0 when it increments to be equal to the single error correction threshold. This register only increments when profiling mode is enabled. This register is not affected by the EE_ZERO_EN or EE_ONE_EN error control bits in the EE_CTRL1 register.

5.7.38 EEPROM Emulation Correctable Error Address Register (EE_COR_ERR_ADD)

During emulation mode, this address is frozen even when read. By setting the SUSP_IGNR bit, (seeTable 5-14) this register can be unfrozen in emulation mode.

This register is not changed with the reset signal and contains unknown data at power-up.

Figure 5-45. EEPROM Emulation Correctable Error Address Register (EE_COR_ERR_ADD) [offset = 314h]

31-16
COR_ERR_ADD
R-u

15-3	2-0
COR_ERR_ADD	B_OFF
R-u	R-u

LEGEND: R = Read only; -*n* = value after reset; -u = unchanged value on internal reset, cleared on power up

Table 5-50. EEPROM Emulation Correctable Error Address Register (EE_COR_ERR_ADD) Field Descriptions

Bit	Field	Value	Description
31-3	COR_ERR_ADD	0-1FFF FFFFh	Correctable Error Address COR_ERR_ADD records the CPU logical address of which a correctable error is detected by the ECC logic. This error address is frozen from begin updated until it is read by the CPU. Additional error are blocked until this register is read.
2-0	B_OFF	0-7h	Byte offset Since ECC is checked on 64 bit data, when checking main memory or OTP, the address captured is aligned to a 64-bit boundary with address bits[2:0] equal to 0. When reading from the ECC bytes, these bits will indicate the failing address of the ECC location associated with the failure. When reading an ECC byte, the ECC is checked against the 64 data bits they protect.

5.7.39 EEPROM Emulation Correctable Error Position Register (EE_COR_ERR_POS)

The bit position is captured during errors when either EE_EOFEN or EE_EZFEN enable bit is set. During error profiling mode when only EE_EPEN is set, the bit position is not captured if a correctable error is detected. This register is frozen while either the EE_ERR_ZERO_FLG or the EE_ERR_ONE_FLG bit is set in the EE_EDACSTATUS register.

During emulation mode, this address is frozen even when read. By setting the SUSP_IGNR bit, this register can be unfrozen in emulation mode.

This register is not changed with the reset signal and contains unknown data at power-up.

Figure 5-46. EEPROM Emulation Correctable Error Position Register (EE_COR_ERR_POS) [offset = 318h]

31	16
Reserved	
R-0	

15	8	7	0
Reserved	TYPE	EE_ERR_POS	
R-0	R-u	R-u	

LEGEND: R = Read only; -*n* = value after reset; -u = unchanged value on internal reset, cleared on power up

Table 5-51. EEPROM Emulation Correctable Error Position Register (EE_COR_ERR_POS) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reads return 0. Writes have no effect.
8	TYPE		Error Type
		0	The error was one of the 64 data bits.
		1	The error was one of the 8 check bits.
7-0	EE_ERR_POS	0-FFh	The bit address of the single-bit error.

5.7.40 EEPROM Emulation Error Status Register (EE_STATUS)

During emulation mode, this address is frozen even when read. By setting the SUSP_IGNR bit in FEDACCTRL1 register, (see Table 5-14) this register can be unfrozen in emulation mode.

This register is not changed with the reset signal and contains unknown data at power-up.

All these error status bits can be deactivated by writing a 1 to the bit. Writing a 0 has no effect.

Bits 0 to 3 show correctable errors while bits 4 to 12 show uncorrectable errors. When the uncorrectable errors are triggered, the current address is stored in the EE_UNC_ERR_ADD register.

These error bits are not set while the FMC is in the suspend mode but they can be cleared in suspend by writing 1s to the bits. By setting the SUSP_IGNR bit to 1, these error bits can be set in suspend mode (see Table 5-14).

Figure 5-47. EEPROM Emulation Error Status Register (EE_STATUS) [offset = 31Ch]

31-24
Reserved
R-0

23-16
Reserved
R-0

15-13	12	11-9	8
Reserved	EE_D_UNC_ERR	Reserved	EE_UNC_ERR
R-0	RCP-u	R-0	RCP-u

7	6	5	4	3	2	1	0
Reserved	EE_CMG	Reserved	EE_CME	EE_D_COR_ERR	EE_ERR_ONE_FLG	EE_ERR_ZERO_FLG	EE_ERR_PRF_FLG
R-0	R-0	R-0	R-0	RCP-u	RCP-u	RCP-u	RCP-u

LEGEND: R = Read only; RCP = Read and Clear in Privilege Mode; -*n* = value after reset; -u = unchanged value on internal reset, cleared on power up

Table 5-52. EEPROM Emulation Error Status Register (EE_STATUS) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reads return 0. Writes have no effect.
12	EE_D_UNC_ERR		Diagnostic Mode Uncorrectable Error Status Flag
		0	No uncorrectable error was detected in diagnostic mode 1.
		1	An uncorrectable error was detected in diagnostic mode 1. This means two or more bits in the data or ECC field have been found in error, or one or more bits in the address have been found in error.
11-9	Reserved	0	Reads return 0. Writes have no effect.
8	EE_UNC_ERR		EEPROM Emulation Uncorrectable Error Flag
		0	No uncorrectable errors were detected in bank 7.
		1	An uncorrectable error was detected in bank 7.
7	Reserved	0	Reads return 0. Writes have no effect.
6	EE_CMG		EEPROM Emulation Compare Malfunction Good
		0	Compare malfunction was detected on the Bus2 SECDED logic.
		1	Compare malfunction was not detected on the Bus2 SECDED logic.
5	Reserved	0	Reads return 0. Writes have no effect.
4	EE_CME		EEPROM Emulation Compare Malfunction Error
		0	Compare malfunction was not detected on the Bus2 SECDED logic.
		1	Compare malfunction was detected on the Bus2 SECDED logic.

Table 5-52. EEPROM Emulation Error Status Register (EE_STATUS) Field Descriptions (continued)

Bit	Field	Value	Description
3	EE_D_COR_ERR		Diagnostic Correctable Error Flag
		0	No correctable error was detected.
		1	A correctable error was detected.
2	EE_ERR_ONE_FLG		Error on One Fail Error Flag
		0	No correctable error was detected.
		1	A correctable error was detected.
1	EE_ERR_ZERO_FLG		Error on Zero Fail Error Flag
		0	No correctable error was detected.
		1	A correctable error was detected.
0	EE_ERR_PRF_FLG		Error Profiling Error Flag
		0	No correctable error was detected.
		1	A correctable error was detected.

5.7.41 EEPROM Emulation Uncorrectable Error Address Register (EE_UNC_ERR_ADD)

During emulation mode, this address is frozen even when read. By setting the SUSP_IGNR bit, (see Table 5-14) this register can be unfrozen in emulation mode.

This register is not changed with the reset signal and contains unknown data at power-up.

Figure 5-48. EEPROM Emulation Uncorrectable Error Address Register (EE_UNC_ERR_ADD) [offset = 320h]

31-16
UNC_ERR_ADD
R-u

15-3	2-0
UNC_ERR_ADD	B_OFF
R-u	R-u

LEGEND: R = Read only; -*n* = value after reset; -u = unchanged value on internal reset, cleared on power up

Table 5-53. EEPROM Emulation Uncorrectable Error Address Register (EE_UNC_ERR_ADD) Field Descriptions

Bit	Field	Value	Description
31-3	UNC_ERR_ADD	0-1FFF FFFFh	Uncorrectable Error Address UNC_ERR_ADD records the CPU logical address of which an uncorrectable error is detected by the ECC logic. This error address is frozen from begin updated until it is read by the CPU. Additional error are blocked until this register is read.
2-0	B_OFF	0-7h	Byte offset Since ECC is checked on 64-bit data, when checking main memory or OTP, the address captured is aligned to a 64-bit boundary with address bits[2:0] equal to 0.When reading from the ECC bytes, these bits will indicate the failing address of the ECC location associated with the failure. When reading an ECC byte, the ECC is checked against the 64 data bits they protect.

5.7.42 Flash Bank Configuration Register (FCFG_BANK)

Figure 5-49. Flash Bank Configuration Register (FCFG_BANK) [offset = 400h]

31 20	19 16
EE_BANK_WIDTH	Reserved
R-90h	R-1

15 4	3 0
MAIN_BANK_WIDTH	Reserved
R-90h	R-2h

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

Table 5-54. Flash Bank Configuration Register (FCFG_BANK) Field Descriptions

Bit	Field	Value	Description
31-20	EE_BANK_WIDTH	90h	Bank 7 width (144-bits wide) This read-only value indicates the maximum number of bits that can be programmed in the bank in one operation. The 144 bits includes 128 data bits and 16 ECC bits.
19-16	Reserved	1	Writes have no effect.
15-4	MAIN_BANK_WIDTH	90h	Width of main Flash banks (144-bits wide) This read-only value indicates the maximum number of bits that can be programmed in the bank in one operation. The 144 bits includes 128 data bits and 16 ECC bits.
3-0	Reserved	2h	Writes have no effect.

Chapter 6

SPNU514C–March 2018

Tightly-Coupled RAM (TCRAM) Module

This chapter describes the tightly-coupled RAM (TCRAM) module.

Topic **Page**

6.1 Overview 327
6.2 RAM Memory Map 328
6.3 Safety Features 329
6.4 TCRAM Auto-Initialization 330
6.5 Trace Module Support 331
6.6 Emulation / Debug Mode Behavior 331
6.7 TCRAM_MODULE_CONTROL_AND_STATUS Registers 332

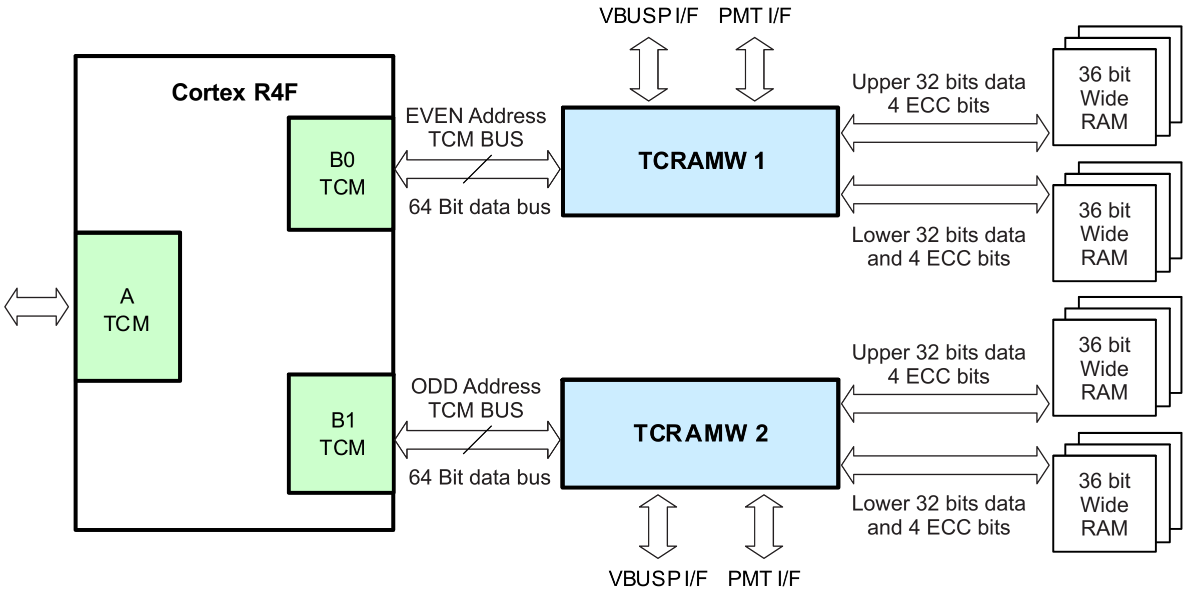
6.1 Overview

The Hercules family of microcontrollers are based on the ARM Cortex-R4F processor. This CPU has two tightly-coupled memory interfaces – ATCM and BTCM, which are used to interface to the program and data memories, respectively. The Hercules MCUs use the ATCM interface for the main flash memory and the BTCM interface for the CPU data RAM.

6.1.1 B0TCM and B1TCM Connection Diagram

The BTCM interface is further divided into two parts – B0TCM and B1TCM, which are both used to interface to actual RAM banks as shown in Figure 6-1.

Figure 6-1. TCRAM Module Connections



6.1.2 Main Features

The main features of the tightly-coupled RAM interface module are:

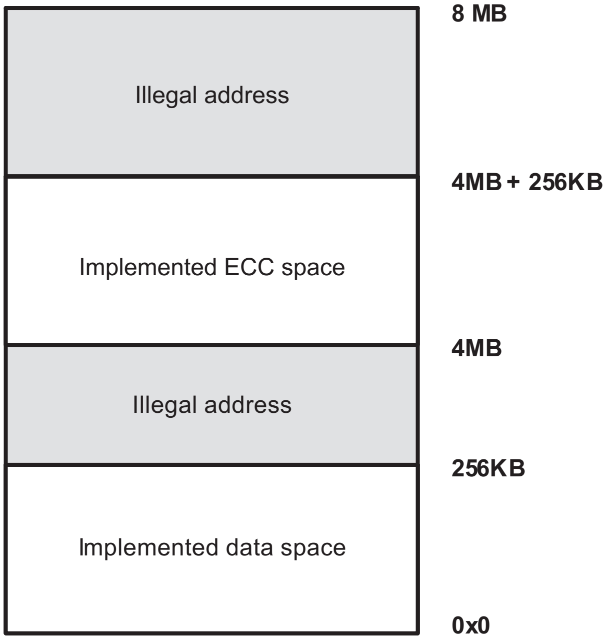
- Controls read/write accesses to the data RAM
- Decodes addresses within the memory region allocated for the RAM
- Supports read and write accesses in 64-bit, 32-bit, 16-bit or 8-bit access sizes
  - Does not support bit-wise operations
- Safety Features:
  - Support for Cortex-R4F CPU's Built-In Single-Error-Correction Double-Error-Detection (SECDED) Logic
    - Uses the CPU's Event bus and maintains the SECDED status in memory-mapped registers
    - Captures the number of occurrences of single-bit or multi-bit errors as well as the RAM address that has the fault
    - Generates signals for indicating single-bit and multi-bit errors to the Error Signaling Module (ESM)
  - Support for Cortex-R4F CPU's Parity Protection Logic for BTCM Address Bus and Control Signals
    - Uses the CPU's TCM Address Parity Scheme and indicates an address bus parity error to the ESM

- Redundant Address Decode Scheme
  - Checks the decoding of CPU address lines and generation of correct memory selects for the RAM banks
  - Also supports checking of the redundant address decode comparators themselves
- Supports auto-initialization of the CPU data RAM banks

6.2 RAM Memory Map

The ARM Cortex-R4F CPU allows up to 8MB to be accessed through the BTCM interface. The Hercules family of microcontrollers support up to 256KB RAM on the BTCM interface. Check the specific part's datasheet to identify the actual amount of TCRAM supported on the part. This RAM is protected by ECC allowing the CPU to correct any single-bit errors and detect double-bit errors within a 64-bit value. The error correction codes (ECC) are stored in the RAM memory space as well. The memory map for the TCRAM and the corresponding ECC space is shown in Figure 6-2. Any access to an unimplemented TCRAM location results in an error response from the TCRAM module.

Figure 6-2. RAM Memory Map



Each RAM data word is 64 bits wide. These 64 bits are divided into two 32 bits per RAM bank as shown in Figure 6-1. The 8 bits of ECC are also divided into 4 bits per RAM bank.

For every 64-bit read from the RAM, an 8-bit ECC is also read by the CPU on its ECC bus. Similarly, for every 64-bit write to the RAM, the CPU also writes an 8-bit ECC using the same ECC bus.

NOTE: **Read-Modify-Write Requirement for Writes to RAM:** The TCRAM interface module supports 64-bit, 32-bit, 16-bit or 8-bit writes to the RAM. However the ECC is calculated by the CPU for 64-bit values only. For any write access smaller than 64 bits, it is necessary to force the CPU to perform a 64-bit read-modify-write operation in order to ensure that the correct ECC is also written. This can be done by setting the bit 1: BTCMRMW of c15, the Secondary Auxiliary Control Register of the CPU. **This bit is already set by default.**

The ECC memory can also be directly accessed via memory-mapped offset addresses starting from 4MB, as shown in Figure 6-2. A read from the ECC space results in the 8-bit ECC value appearing on each byte of the 64-bit CPU data. The ECC memory can only be written to as a 64-bit access. The write to the ECC space must also first be enabled via the RAM Control Register (RAMCTRL).

NOTE: **No ECC Error Generated for Accesses to ECC Memory:** A read from the ECC space send the ECC value on both the 64-bit TCM read data bus as well as the 8-bit ECC bus. This could result in the detection of a multi-bit error by the SECDED logic inside the CPU. The TCRAM interface module ignores the ECC error indicated by the CPU for the access to the ECC space.

6.3 Safety Features

The TCRAM interface module incorporates some features that are designed specifically with safety considerations. These are described in the following sections.

6.3.1 Support for Cortex-R4F CPU's Single-Error-Correction Double-Error-Detection (SECDED)

The TCRAM interface module monitors the CPU's event bus. The CPU's event bus signals single-bit or multi-bit errors for B0TCM as well as B1TCM separately. These signals are monitored by the TCRAM modules for each of these interfaces.

TCRAM Interface Module Features dedicated for SECDED Support:

- Dedicated single-bit error counter
  - This counter is stored in a memory-mapped register called RAMOCCUR
  - RAMOCCUR is used to count the single-bit errors corrected by the CPU's SECDED logic
  - The TCRAM interface module allows the application to generate an interrupt via the RAMINTCTRL register when the number of single-bit errors corrected by the CPU exceeds a programmable threshold, RAMTHRESHOLD
- RAM Error Status Register
  - The errors detected by the TCRAM interface module as well as those indicated by the CPU are flagged in the RAMERRSTATUS register
  - There are separate bits to indicate single-bit error, double-bit error, address decode failure, address compare logic failure, read-address parity failure, and write-address parity failure
- ECC Error Address Capture
  - Separate registers to hold the address on which a single-bit error is detected (RAMSERRADDR) or a double-bit error is detected (RAMUERRADDR)
  - The RAMSERRADDR register is only updated when the RAMTHRESHOLD value is set to 1
  - Both the RAMSERRADDR and RAMUERRADDR capture the 64-bit-aligned address for the access to the TCRAM as an offset from the base address of the TCRAM (0x08000000 by default)

NOTE: **Cortex-R4F CPU Event Bus Signaling Not Enabled By Default:** Upon power-up and after a CPU reset, the event signaling mechanism inside the Cortex-R4F CPU is disabled. This feature must be enabled by setting the Export (X-bit) of the Performance Monitoring and Control Register (PMNC) in the CPU. The TCRAM interface module can only capture the single-bit or double-bit ECC error occurrences once the CPU's event signaling mechanism is enabled.

6.3.2 Support for Cortex-R4F CPU's Address and Control Bus Parity Checking

The Cortex-R4F CPU calculates a single parity-bit for the TCRAM address and control signals. The TCRAM interface module also computes this parity bit based on the CPU's address bus and control signals. The computed parity bit is compared against the parity bit received from the CPU. A mismatch is signaled as an Address Parity Failure to the Error Signaling Module (ESM) group2 channel 10 or 12. There is a separate address parity failure error channel for B0TCM and B1TCM.

The 64-bit TCRAM address which fails the parity check is captured in the RAMPERRADDR register as an offset from the base address of the TCRAM (0x08000000 by default). The TCRAM interface module also indicates the type of access, read or write, that failed the parity check. This is indicated by the RADDR_PAR_FAIL or the WADDR_PAR_FAIL status flags in the RAMERRSTATUS register.

The RAMERRSTATUS and RAMPERRADDR registers must be cleared by the application in order for the TCRAM interface module to continue capturing subsequent errors and error addresses.

The parity scheme used for the described parity checking mechanism is defined by the global system parity selection. This can be configured using the DEVPARSEL field of the DEVCR1 control register in the system module. This device-wide parity scheme can be overridden inside the TCRAM interface module by configuring the Address Parity Override field in the RAMCTRL register.

NOTE: **No Change Of Parity Scheme On-The-Fly:** The TCRAM interface module does not support on-the-fly change to the parity scheme being used for checking the CPU address bus and control bus. The application must ensure that the parity polarity (odd or even) is not changed while there is an ongoing access to the TCRAM.

6.3.3 Redundant Address Decode

The TCRAM interface module generates the memory selects for each of the TCRAM banks as well as the ECC memory based on the CPU address. The logic to generate these memory selects is duplicated and the outputs compared to detect any address decode errors. A mismatch is indciated as an Address Error to the Error Signaling Module (ESM), one signal for B0TCM and one for B1TCM. The TCRAM or ECC address that caused the fault is captured in the RAMUERRADDR register. This 64-bit-aligned address is stored as an offset from the base of the TCRAM or ECC memory.

As described earlier, each individual physical RAM bank is 36 bits wide. Each RAM bank contributes 32 bits of data and 4 bits of ECC when the bus master performs a 64-bit read from the TCRAM. Each TCRAM bank receives a memory select and the address from the TCRAM interface module. Any difference between the address and the memory selects results in wrong data and ECC pair being sent to the CPU. The CPU's SECDED block will detect this data error.

The TCRAM interface module also supports a mechanism to test the operation of the redundant address decode logic and the compare logic. This testing is supported by providing a test stimulus, and can be triggered by the application by configuring the RAMTEST register. The address of any error identified during testing of the redundant address decode and compare logic is not captured in the RAMUERRADDR register.

NOTE: **Address decode checking when in compare logic test mode:** When the address decode and compare logic test mode is enabled, the redundant address decode and compare logic is not available for checking the proper generation of the memory selects for the TCRAM and ECC memory.

6.4 TCRAM Auto-Initialization

The RAM memory can be initialized by using the dedicated auto-initialization hardware. The TCRAM Module initializes the entire memory when the auto-initialization is enabled by the INIT_DOMAIN register upon receiving a MMI_INIT pulse from the system module. All enabled RAM data memory locations are initialized to zeros and the ECC memory is initialized to the correct ECC value for zeros, that is, 0Ch.

6.5 Trace Module Support

No data is traced for an access to ECC memory.

6.6 Emulation / Debug Mode Behavior

The following describes the behavior of the TCRAM Module when in debug mode:

- The RAMOCCUR register continues to count the single-bit error corrections performed by the Cortex-R4F CPU's SECDED logic.
- No single-bit error interrupt is generated nor is any single-bit error address captured even when the RAMOCCUR counter reaches the programmed single-bit error correction threshold.
- No uncorrectable error interrupt is generated nor is any double-bit error address captured.
- No address parity error interrupt is generated nor is any parity error address captured.
- The RAMUERRADDR register is not cleared by a read in debug mode.
  - That is, if a double-bit error address is captured and is not read by the CPU before entering debug mode, then it remains frozen during debug mode even if it is read.
- The RAMPERRADDR register is not cleared by a read in debug mode.

6.7 TCRAM_MODULE_CONTROL_AND_STATUS Registers

Table 6-1 lists the memory-mapped registers for the TCRAM_MODULE_CONTROL_AND_STATUS. All register offset addresses not listed in Table 6-1 should be considered as reserved locations and the register contents should not be modified.

The TCRAM Module registers are accessed through the system module registers' space in the Cortex-R4F CPU's memory map. All registers are 32-bit wide and are located on a 32-bit boundary. Reads and writes to registers are supported in 8-, 16-, and 32-bit accesses. The base address for the control registers is FFFF F800h for even RAM ECC and FFFF F900h for odd RAM ECC.

Table 6-1. TCRAM_MODULE_CONTROL_AND_STATUS Registers

Offset	Acronym	Register Name	Section
0h	RAMCTRL	TCRAM Module Control Register	Section 6.7.1
4h	RAMTHRESHOLD	TCRAM Module Single-Bit Error Correction Threshold Register	Section 6.7.2
8h	RAMOCCUR	TCRAM Module Single-Bit Error Occurrences Control Register	Section 6.7.3
Ch	RAMINTCTRL	TCRAM Module Interrupt Control Register	Section 6.7.4
10h	RAMERRSTATUS	TCRAM Module Error Status Register	Section 6.7.5
14h	RAMSERRADDR	TCRAM Module Single-Bit Error Address Register	Section 6.7.6
1Ch	RAMUERRADDR	TCRAM Module Uncorrectable Error Address Register	Section 6.7.7
30h	RAMTEST	TCRAM Module Test Mode Control Register	Section 6.7.8
38h	RAMADDRDECVECT	TCRAM Module Test Mode Vector Register	Section 6.7.9
3Ch	RAMPERRADDR	TCRAM Module Parity Error Address Register	Section 6.7.10
40h	INIT_DOMAIN	Auto-Memory Initialization Enable Register	Section 6.7.11

Complex bit access types are encoded to fit into small table cells. Table 6-2 shows the codes that are used for access types in this section.

Table 6-2. TCRAM_MODULE_CONTROL_AND_STATUS Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W1CP	1C P W	1 to clear Requires privileged access Write
WP	P W	Requires privileged access Write
Reset or Default Value		
-n		Value after reset or the default value

6.7.1 RAMCTRL Register (Offset = 0h) [reset = 0005000Ah]

RAMCTRL is shown in Figure 6-3 and described in Table 6-3.

Return to Summary Table.

RAMCTRL controls the safety features supported by the TCRAM Module.

Figure 6-3. RAMCTRL Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td></tr>
<tr><td colspan="4">RESERVED</td><td colspan="4">ADDR_PARITY_OVERRIDE</td></tr>
<tr><td colspan="4">R-0h</td><td colspan="4">R/WP-0h</td></tr>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="4">RESERVED</td><td colspan="4">ADDR_PARITY_DISABLE</td></tr>
<tr><td colspan="4">R-0h</td><td colspan="4">R/WP-5h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="7">RESERVED</td><td>ECC_WR_EN</td></tr>
<tr><td colspan="7">R-0h</td><td>R/WP-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="4">RESERVED</td><td colspan="4">ECC_DETECT_EN</td></tr>
<tr><td colspan="4">R-0h</td><td colspan="4">R/WP-Ah</td></tr>
</table>

Table 6-3. RAMCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	Reads return 0. Writes have no effect.
27-24	ADDR_PARITY_OVERRIDE	R/WP	0h	Address Parity Override. This field, when set to Dh, will invert the parity scheme selected by the device global parity selection. The address parity checker would then work on the inverted parity scheme. By default, the parity scheme is the same as the global device parity scheme. All other values = Parity scheme is the same as the device global parity scheme. Dh = Parity scheme is opposite to the device global parity scheme.
23-20	RESERVED	R	0h	Reads return 0. Writes have no effect.
19-16	ADDR_PARITY_DISABLE	R/WP	5h	Address Parity Detect Disable. This field, when set to Ah, disables the parity checking for the address bus. The parity checking is enabled when this field is set to any other value. **Note: The application must ensure that the WADDR_PAR_FAIL and RADDR_PAR_FAIL bits in RAMERRSTATUS register are cleared before enabling address parity checking.** All other values = Address parity checking is enabled. Ah = Address parity checking is disabled.
15-9	RESERVED	R	0h	Reads return 0. Writes have no effect.
8	ECC_WR_EN	R/WP	0h	ECC Memory Write Enable. This bit is provided to prevent accidental writes to the ECC memory. A write access to the ECC memory is allowed only when the ECC_WR_EN bit is set to 1. If this bit is cleared, then any writes to ECC memory are ignored. **Note: Reads are allowed from the ECC memory regardless of the state of the ECC_WR_EN bit.** 0h = ECC memory writes are disabled. 1h = ECC memory writes are enabled.
7-4	RESERVED	R	0h	Reads return 0. Writes have no effect.

Table 6-3. RAMCTRL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3-0	ECC_DETECT_EN	R/WP	Ah	ECC Detect Enable. This is a 4-bit key to enable the ECC detection feature in the TCRAM Module. If this field is set to any value other than 5h, then the TCRAM Module starts monitoring the TCM event bus and generates the corresponding error status flags. The error status updates are done only when the ECC_DETECT_EN field is not 5h. The ECC detection is enabled by default, as the ECC_DETECT_EN field default value is Ah. All other values = ECC detection is enabled. 5h = ECC detection is disabled.

6.7.2 RAMTHRESHOLD Register (Offset = 4h) [reset = 0h]

RAMTHRESHOLD is shown in Figure 6-4 and described in Table 6-4.

Return to Summary Table.

RAMTHRESHOLD allows the application to configure the number of single-bit error corrections by the SECDED logic inside the Cortex-R4F CPU before generating a single-bit error interrupt.

Figure 6-4. RAMTHRESHOLD Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">RESERVED</td><td colspan="16">THRESHOLD</td></tr>
<tr><td colspan="16">R-0h</td><td colspan="16">R/WP-0h</td></tr>
</table>

Table 6-4. RAMTHRESHOLD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reads return 0. Writes have no effect.
15-0	THRESHOLD	R/WP	0h	Single-bit Error Threshold Count. This field contains the threshold value for the Single-bit Error Correction (SEC) occurrences before the single-bit error interrupt is generated. If this threshold is set to 1 then all single-bit error addresses are captured. To enable the error occurrence detection, the threshold must be set to a non-zero value.

6.7.3 RAMOCCUR Register (Offset = 8h) [reset = 0h]

RAMOCCUR is shown in Figure 6-5 and described in Table 6-5.

Return to Summary Table.

RAMOCCUR indicates the number of single-bit error corrections performed by the SEC logic inside the Cortex-R4F CPU.

Figure 6-5. RAMOCCUR Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">RESERVED</td></tr>
<tr><td colspan="16">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">SINGLE_ERROR_OCCURRENCES</td></tr>
<tr><td colspan="16">R/WP-0h</td></tr>
</table>

Table 6-5. RAMOCCUR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reads return 0. Writes have no effect.
15-0	SINGLE_ERROR_ OCCURRENCES	R/WP	0h	Single-bit Error Correction Occurrences. This 16-bit counter contains the number of single-bit error occurrences. RAMOCCUR is reset to zero when it becomes equal to the THRESHOLD value set in the RAMTHRESHOLD register. The application must clear the RAMOCCUR register by writing 0x0 before setting the THRESHOLD value. If the RAMOCCUR value is already higher than the programmed THRESHOLD value, then the counter increments and wraps around (overflow) to zero. **Note: If the application tries to clear the RAMOCCUR register at the same time as the TCRAM Module tried to update it, then the TCRAM Module takes priority.** **Note: When the RAMTHRESHOLD register is set to 1, then the RAMOCCUR register must be cleared whenever a single-bit error correction occurs in order to count subsequent single-bit error corrections.**

6.7.4 RAMINTCTRL Register (Offset = Ch) [reset = 0h]

RAMINTCTRL is shown in Figure 6-6 and described in Table 6-6.

Return to Summary Table.

RAMINTCTRL enables the generation of an interrupt to the CPU whenever the number of single-bit error corrections (RAMOCCUR) reaches the programmed threshold (RAMTHRESHOLD).

Figure 6-6. RAMINTCTRL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							SERR_EN
R-0h							R/WP-0h

Table 6-6. RAMINTCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reads return 0. Writes have no effect.
0	SERR_EN	R/WP	0h	Single-bit Error Correction Interrupt Enable. This bit, when set to 1, enables the generation of the single-bit error interrupt when the RAMOCCUR count reaches the programmed RAMTHRESHOLD. If the interrupt is not enabled, the single-bit error counter continues to count by resetting back to zero without generating any error interrupt. The SERR status flag in the RAMERRSTATUS register gets set regardless of whether the SERR interrupt is enabled or not. 0h = Single-bit error generation is disabled. 1h = Single-bit error generation is enabled.

6.7.5 RAMERRSTATUS Register (Offset = 10h) [reset = 0h]

RAMERRSTATUS is shown in Figure 6-7 and described in Table 6-7.

Return to Summary Table.

RAMERRSTATUS indicates the status of the various error conditions monitored by the TCRAM Module.

For equality test:

If the comparator matches (no true silicon fail), there is no status bit set for ADDR_COMP_LOGIC_FAIL or ADDR_DEC_FAIL.

If there is true silicon malfunction, ADDR_COMP_LOGIC_FAIL and ADDR_DEC_FAIL will be set, no UERRADDRESS is captured.

For inequality test, the compare vector will not match since non-inverted and inverted values of the same test vector are fed to the comparator:

If there is no silicon malfunction on any of the comparator bits, then only ADDR_DEC_FAIL will be set. This is chosen so that we can ensure the functional ADDR_DEC_FAIL status bit data path can be tested.

If there is a silicon malfunction on any of the comparator bits, then, ADDR_COMP_LOGIC_FAIL and ADDR_DEC_FAIL will be set, no UERRADDRESS is captured.

Figure 6-7. RAMERRSTATUS Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="6">RESERVED</td><td>WADDR_PAR_FAIL</td><td>RADDR_PAR_FAIL</td></tr>
<tr><td colspan="6">R-0h</td><td>R/W1CP-0h</td><td>R/W1CP-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">RESERVED</td><td>DERR</td><td>ADDR_COMP_LOGIC_FAIL</td><td>RESERVED</td><td>ADDR_DEC_FAIL</td><td>RESERVED</td><td>SERR</td></tr>
<tr><td colspan="2">R-0h</td><td>R/W1CP-0h</td><td>R/W1CP-0h</td><td>R-0h</td><td>R/W1CP-0h</td><td>R-0h</td><td>R/W1CP-0h</td></tr>
</table>

Table 6-7. RAMERRSTATUS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0h	Reads return 0. Writes have no effect.
9	WADDR_PAR_FAIL	R/W1CP	0h	This bit indicates a Write Address Parity Failure. This bit must be cleared by writing 1 to it in order to enable the capture of parity error address for subsequent failures. This bit must be in a cleared state for generation of any new parity error interrupt.
8	RADDR_PAR_FAIL	R/W1CP	0h	This bit indicates a Read Address Parity Failure. This bit must be cleared by writing 1 to it in order to enable the capture of parity error address for subsequent failures. This bit must be in a cleared state for generation of any new parity error interrupt.
7-6	RESERVED	R	0h	Reads return 0. Writes have no effect.
5	DERR	R/W1CP	0h	This bit indicates a multi-bit error detected by the Cortex-R4F SECDED logic.
4	ADDR_COMP_LOGIC_FAIL	R/W1CP	0h	Address decode logic element failed. This bit indicates that the redundant address decode logic test scheme has detected that a compare element has malfunctioned during the testing of the logic. This bit has to be cleared by writing 1 to it in order to enable the capture of uncorrectable error address for subsequent failures. This bit has to be in a cleared state for generation of a new uncorrectable error interrupt. This bit only gets set in the test mode, and has no relevance in functional mode.

Table 6-7. RAMERRSTATUS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	RESERVED	R	0h	Reads return 0. Writes have no effect.
2	ADDR_DEC_FAIL	R/W1CP	0h	Address decode failed. This bit indicates that an address error interrupt was generated by the redundant address decode and compare logic due to a functional failure. This bit must be cleared by writing 1 to it in order to enable the capture of uncorrectable error address for subsequent failures. This bit has to be in a cleared state for generation of a new address error interrupt.
1	RESERVED	R	0h	Reads return 0. Writes have no effect.
0	SERR	R/W1CP	0h	Single Error Status. This bit indicates that the single-bit error threshold has been reached. This bit is set even if the single-bit error threshold interrupt is disabled. This bit must be cleared by writing 1 to it in order to clear the interrupt request and to enable subsequent single-bit error interrupt generation.

6.7.6 RAMSERRADDR Register (Offset = 14h) [reset = 0h]

RAMSERRADDR is shown in Figure 6-8 and described in Table 6-8.

Return to Summary Table.

RAMSERRADDR captures the address for which the Cortex-R4F CPU detected a single-bit error.
The SERR bit in the RAMERRSTATUS register must be cleared, by writing 1 to the bit, in order to enable the RAMSERRADDR register to capture a subsequent new error address.

Figure 6-8. RAMSERRADDR Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="14">RESERVED</td><td colspan="2">SINGLE_ERROR_ADDRESS</td></tr>
<tr><td colspan="14">R-0h</td><td colspan="2">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="13">SINGLE_ERROR_ADDRESS</td><td colspan="3">RESERVED</td></tr>
<tr><td colspan="13">R-0h</td><td colspan="3">R-0h</td></tr>
</table>

Table 6-8. RAMSERRADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-18	RESERVED	R	0h	Reads return 0. Writes have no effect.
17-3	SINGLE_ERROR_ADDRESS	R	0h	This register captures bits 17-3 of the address for which the Cortex-R4F CPU detects a single-bit error when the RAMTHRESHOLD register is set to 1. The lower 3 bits are always tied to zero so that the address captured is a double-word (64-bit) address. This is a 64-bit-aligned address is stored as an offset from the base of the TCRAM or ECC memory. This register can only be reset by asserting power-on reset, and holds the last error address even after a system reset.
2-0	RESERVED	R	0h	Reads return 0. Writes have no effect.

6.7.7 RAMUERRADDR Register (Offset = 1Ch) [reset = 0h]

RAMUERRADDR is shown in Figure 6-9 and described in Table 6-9.

Return to Summary Table.

RAMUERRADDR captures the address for which the Cortex-R4F CPU detected a multi-bit error.

Figure 6-9. RAMUERRADDR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED									UNCORRECTABLE_ERROR_ADDRESS						
R-0h									R-0h						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNCORRECTABLE_ERROR_ADDRESS													RESERVED		
R-0h													R-0h		

Table 6-9. RAMUERRADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-23	RESERVED	R	0h	Reads return 0. Writes have no effect.
22-3	UNCORRECTABLE_ ERROR_ADDRESS	R	0h	This register captures the address for which there was an uncorrectable error. The uncorrectable error is indicated by the Cortex-R4F CPU's SECDED logic. For the SECDED multi-bit or double-bit uncorrectable error this register stores bits 17-3 of the TCM access address. The lower 3 bits 2-0 are always read as zeros to indicate that the latched address is a double-word address. The address bits 31-18 are read as zeros. This is a 64-bit-aligned address is stored as an offset from the base of the TCRAM or ECC memory. For a redundant address decode and compare logic error this register stores the complete TCM access address rounded to a double-word boundary (bits 22-3). This error is also indicated by the ADDR_DEC_FAIL flag in the RAMERRSTATUS register. No error address is stored as a result of a redundant address logic test. The register has to be read-cleared to enable further error address captures. Reading the register does not clear its contents but enables the register to be updated with an uncorrectable error address. This register can only be reset by asserting power-on reset, and holds the last error address even after a system reset.
2-0	RESERVED	R	0h	Reads return 0. Writes have no effect.

6.7.8 RAMTEST Register (Offset = 30h) [reset = 5h]

RAMTEST is shown in Figure 6-10 and described in Table 6-10.

Return to Summary Table.

RAMTEST controls the test mode of the TCRAM Module.
For equality test:

If the comparator matches (no true silicon fail), there is no status bit set for ADDR_COMP_LOGIC_FAIL or ADDR_DEC_FAIL.

If there is true silicon malfunction, ADDR_COMP_LOGIC_FAIL and ADDR_DEC_FAIL will be set, no UERRADDRESS is captured.

For inequality test, the compare vector will not match since non-inverted and inverted values of the same test vector are fed to the comparator:

If there is no silicon malfunction on any of the comparator bits, then only ADDR_DEC_FAIL will be set. This is chosen so that we can ensure the functional ADDR_DEC_FAIL status bit data path can be tested.

If there is a silicon malfunction on any of the comparator bits, then, ADDR_COMP_LOGIC_FAIL and ADDR_DEC_FAIL will be set, no UERRADDRESS is captured.

Figure 6-10. RAMTEST Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="7">RESERVED</td><td>TRIGGER</td></tr>
<tr><td colspan="7">R-0h</td><td>R/WP-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">TEST_MODE</td><td colspan="2">RESERVED</td><td colspan="4">TEST_ENABLE</td></tr>
<tr><td colspan="2">R/WP-0h</td><td colspan="2">R-0h</td><td colspan="4">R/WP-5h</td></tr>
</table>

Table 6-10. RAMTEST Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Reads return 0. Writes have no effect.
8	TRIGGER	R/WP	0h	Test Trigger. This is an auto reset test trigger used to test the redundant address decode and compare logic. A redundant address decode test is executed when test mode is enabled and the test trigger is applied by writing a 1 to this bit. The trigger is valid only if test is enabled and the correct mode is configured in the TEST_MODE field, and the ADDR_DEC_FAIL, ADDR_COMP_LOGIC_FAIL, and DERR flags are cleared in the RAMERRSTATUS register and the RAMUERRADDR register is read-cleared.

Table 6-10. RAMTEST Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	TEST_MODE	R/WP	0h	Test Mode. This field selects either equality or inequality testing schemes. 0h = Reserved 1h = Inequality check is done. The test stimulus stored in ADDRTEST_VECT register is inverted and fed into one channel and the non-inverted vector is fed into the other channel. If the XOR of these inputs is zero, then the UERR interrupt is generated and ADDR_COMP_LOGIC_FAIL flag is set in RAMERRSTATUS register. 2h = Equality check is done. The test stimulus stored in ADDRTEST_VECT register is fed directly to both the channels of the comparator. If the XOR of these two inputs is not zero, then UERR interrupt is generated and ADDR_COMP_LOGIC_FAIL flag is set in RAMERRSTATUS register. 3h = Reserved
5-4	RESERVED	R	0h	Reads return 0. Writes have no effect.
3-0	TEST_ENABLE	R/WP	5h	Test Enable. This is a 4-bit key to enable the redundant address decode and compare logic test scheme. If the test scheme is enabled, then the compare logic uses the test vector inputs from the ADDRTEST_VECT register. The functional path comparison is disabled when test mode is enabled. All other values = Test mode is disabled. Ah = Test mode is enabled.

6.7.9 RAMADDRDECVECT Register (Offset = 38h) [reset = 0h]

RAMADDRDECVECT is shown in Figure 6-11 and described in Table 6-11.

Return to Summary Table.

RAMADDRDECVECT is used for testing the redundant address decode and compare logic of the TCRAM Module.

Figure 6-11. RAMADDRDECVECT Register

31	30	29	28	27	26	25	24
RESERVED					ECC_SELECT	RESERVED	
R-0h					R/WP-0h	R-0h	
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RAM_CHIP_SELECT							
R/WP-0h							
7	6	5	4	3	2	1	0
RAM_CHIP_SELECT							
R/WP-0h							

Table 6-11. RAMADDRDECVECT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-27	RESERVED	R	0h	Reads return 0. Writes have no effect.
26	ECC_SELECT	R/WP	0h	ECC Select. This bit is used to store the ECC select value for the redundant address decode and compare logic. The stored value is passed as test stimulus for the built-in test scheme.
25-16	RESERVED	R	0h	Reads return 0. Writes have no effect.
15-0	RAM_CHIP_SELECT	R/WP	0h	RAM Chip Select. This field is used to store the RAM chip select value for the redundant address decode and compare logic. The stored value is passed as test stimulus for the built-in test scheme.

6.7.10 RAMPERRADDR Register (Offset = 3Ch) [reset = X]

RAMPERRADDR is shown in Figure 6-12 and described in Table 6-12.

Return to Summary Table.

RAMPERRADDR stores the address for which an address-parity error was detected.

Figure 6-12. RAMPERRADDR Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="9">RESERVED</td><td colspan="7">ADDRESS_PARITY_ERROR_ADDRESS</td></tr>
<tr><td colspan="9">R-0h</td><td colspan="7">R-X</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="13">ADDRESS_PARITY_ERROR_ADDRESS</td><td colspan="3">RESERVED</td></tr>
<tr><td colspan="13">R-X</td><td colspan="3">R-0h</td></tr>
</table>

Table 6-12. RAMPERRADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-23	RESERVED	R	0h	Reads return 0. Writes have no effect.
22-3	ADDRESS_PARITY_ERROR_ADDRESS	R	X	Parity Error Address. This register stores the double-word boundary (bits 22-3) of the TCM access address for which there was an address parity error. This register must be read-cleared to enable further error address captures. Reading the register does not clear the register contents but enables the register to be updated with a new parity error address. This is a 64-bit aligned address stored as an offset from the base of the TCRAM or ECC memory.
2-0	RESERVED	R	0h	Reads return 0. Writes have no effect.

6.7.11 INIT_DOMAIN Register (Offset = 40h) [reset = FFh]

INIT_DOMAIN is shown in Figure 6-13 and described in Table 6-13.

Return to Summary Table.

INIT_DOMAIN is not supposed to be written into when a memory initialization is proceeding. In other words, the INIT_DOMAIN register has to be programmed prior to the generation of the SYS_TCRAMW_MMI_INIT_I pulse from the system module. The INIT_DOMAIN register has to be programmed based on the power domains that are awake and the hardware shall not provide any indication if a powered down domain is selected for auto-memory initialization.

Figure 6-13. INIT_DOMAIN Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								AUTO_MEM_INIT_ENABLE							
R-0h								R/WP-FFh							

Table 6-13. INIT_DOMAIN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	RESERVED	R	0h	Reads return 0. Writes have no effect.
7-0	AUTO_MEM_INIT_ENABLE	R/WP	FFh	These bits are used to enable auto-memory initialization per power domain. These bits are all 1 by default and need to be written to 0 to prevent a certain power domain from getting initialized. Note that these bits are only enable bits for their respective power domains and the indication for the start of Memory Initialization would still come from the system module through the pulsed input signal SYS_TCRAMW_MMI_INIT_I. bit[0] represents the enable bit for power domain 0. bit[1] represents the enable bit for power domain 1. bit[2] represents the enable bit for power domain 2. bit[3] represents the enable bit for power domain 3. bit[4] represents the enable bit for power domain 4. bit[5] represents the enable bit for power domain 5. bit[6] represents the enable bit for power domain 6. bit[7] represents the enable bit for power domain 7.

Programmable Built-In Self-Test (PBIST) Module

This chapter describes the programmable built-in self-test (PBIST) controller module used for testing the on-chip memories on the Hercules microcontrollers.

Topic **Page**

7.1 Overview 348
7.2 RAM Grouping and Algorithm 349
7.3 PBIST Flow 350
7.4 Memory Test Algorithms on the On-chip ROM 352
7.5 PBIST Control Registers 354
7.6 PBIST Configuration Example 367

7.1 Overview

The PBIST (Programmable Built-In Self-Test) controller architecture provides a run-time-programmable memory BIST engine for varying levels of coverage across many embedded memory instances.

7.1.1 *Features of PBIST*

- Information regarding on-chip memories, memory groupings, memory background patterns and test algorithms stored in dedicated on-chip PBIST ROM
- Host processor interface to configure and start BIST of memories
- Supports testing of PBIST ROM itself as well
- Supports testing of each memory at its maximum access speed in application
- Implements intelligent clock gating to conserve power

NOTE: Refer to the device datasheet for the maximum PBIST ROM clock frequency supported.

7.1.2 *PBIST vs. Application Software-Based Testing*

The PBIST architecture consists of a small coprocessor with a dedicated instruction set targeted specifically toward testing memories. This coprocessor executes test routines stored in the PBIST ROM and runs them on multiple on-chip memory instances. The on-chip memory configuration information is also stored in the PBIST ROM. The testing is done in parallel for each of the CPU data RAMs, while it is done sequentially for the rest of the memories.

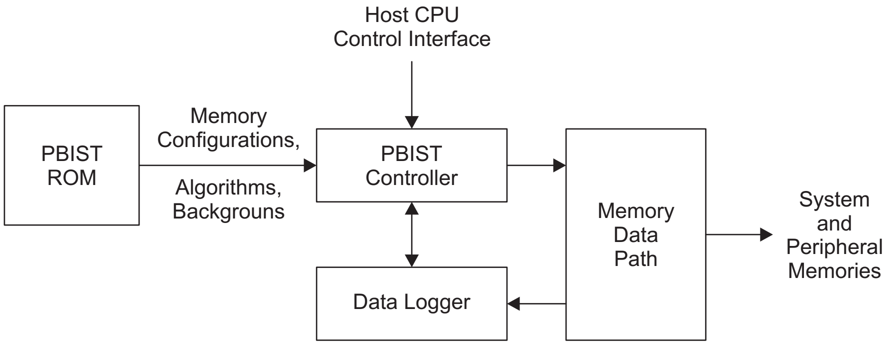
The PBIST Controller architecture offers significant advantages over tests running on the main Cortex-R4F processor (application software-based testing):

- Embedded CPUs have a long access path to memories outside the tightly-couple memory sub-system, while the PBIST controller has a dedicated path to the memories specifically for the self-test
- Embedded CPUs are designed for their targeted use and are often not easily programmed for memory test algorithms.
- The memory test algorithm code on embedded CPUs is typically significantly larger than that needed for PBIST.
- The embedded CPU is significantly larger than the PBIST controller.

7.1.3 *PBIST Block Diagram*

Figure 7-1 illustrates the basic PBIST blocks and its wrapper logic for the device.

Figure 7-1. PBIST Block Diagram



7.1.3.1 On-chip ROM

The on-chip ROM contains the information regarding the algorithms and memories to be tested.

7.1.3.2 Host Processor Interface to the PBIST Controller Registers

The Cortex-R4F CPU can select the algorithm and RAM groups for the memories' self-test from the on-chip ROM based on the application requirements. Once the self-test has executed, the CPU can query the PBIST controller registers to identify any memories that failed the self-test and to then take appropriate next steps as required by the application's author.

7.1.3.3 Memory Data Path

This is the read and write data path logic between different system and peripheral memories tightly coupled to the PBIST memory interface. The PBIST controller executes each selected algorithm on each valid memory group sequentially until all the algorithms are executed.

NOTE: Not all algorithms are designed to run on all RAM groups. If an algorithm is selected to run on an incompatible memory, this will result in a failure. Refer to Table 2-5 and for RAM grouping and algorithm information.

7.2 RAM Grouping and Algorithm

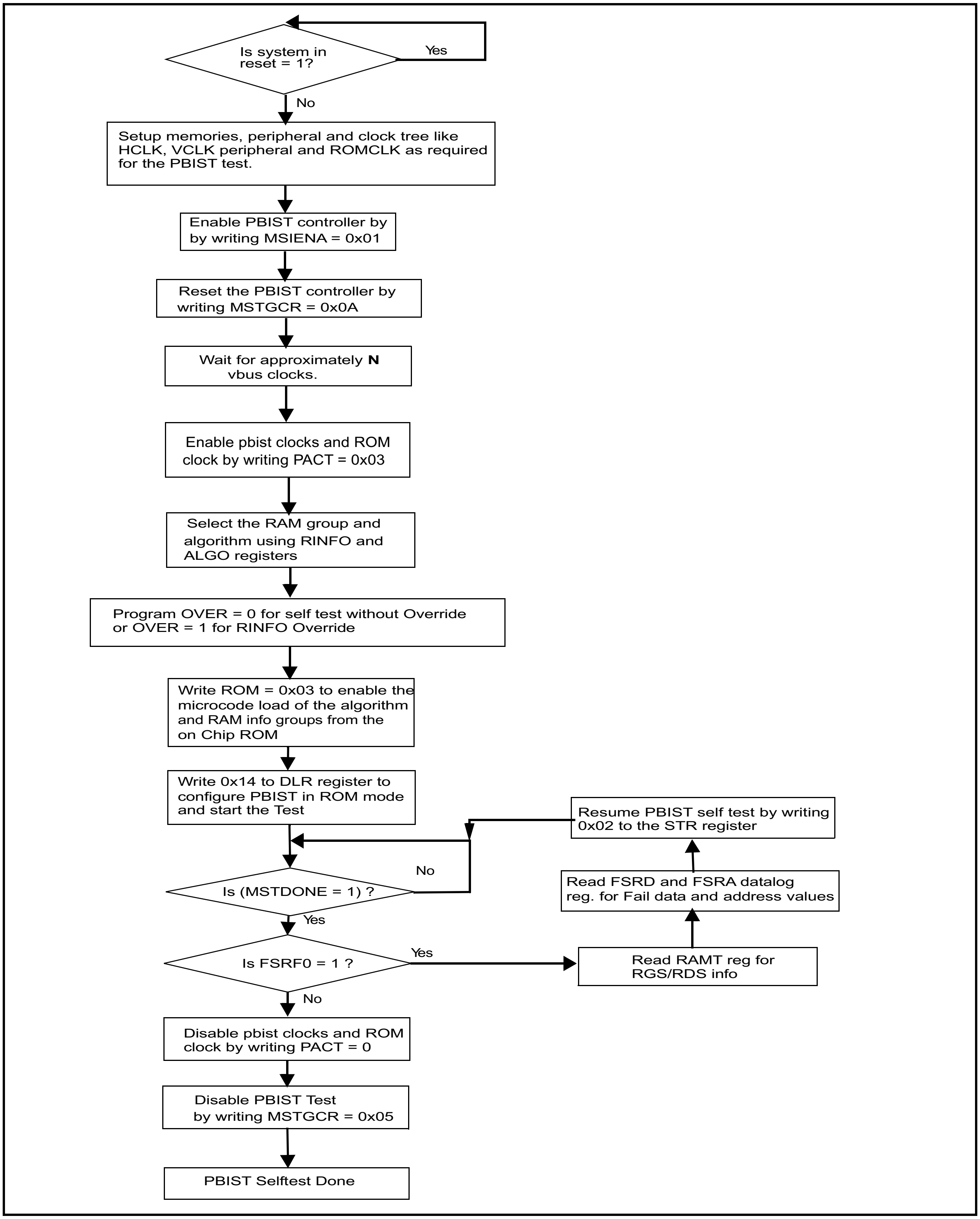
Table 2-5 gives the list of RAM groups and their types supported on the device. maps the different algorithms supported in application mode for the RAM groups with the background patterns used for the particular algorithm.

NOTE: March13 is the most recommended algorithm for the memory self-test.

7.3 PBIST Flow

Figure 7-2 illustrates the memory self-test flow.

Figure 7-2. PBIST Memory Self-Test Flow Diagram


Is system in reset = 1?
Yes
No
Setup memories, peripheral and clock tree like HCLK, VCLK peripheral and ROMCLK as required for the PBIST test.
Enable PBIST controller by by writing MSIENA = 0x01
Reset the PBIST controller by writing MSTGCR = 0x0A
Wait for approximately **N** vbus clocks.
Enable pbist clocks and ROM clock by writing PACT = 0x03
Select the RAM group and algorithm using RINFO and ALGO registers
Program OVER = 0 for self test without Override or OVER = 1 for RINFO Override
Write ROM = 0x03 to enable the microcode load of the algorithm and RAM info groups from the on Chip ROM
Write 0x14 to DLR register to configure PBIST in ROM mode and start the Test
Resume PBIST self test by writing 0x02 to the STR register
No
Is (MSTDONE = 1) ?
Read FSRD and FSRA datalog reg. for Fail data and address values
Yes
Yes
Is FSRF0 = 1 ?
Read RAMT reg for RGS/RDS info
No
Disable pbist clocks and ROM clock by writing PACT = 0
Disable PBIST Test by writing MSTGCR = 0x05
PBIST Selftest Done

7.3.1 PBIST Sequence

1. Configure the device clock sources and domains so that they are running at their target frequencies.
2. Program the HCLK to PBIST ROM clock ratio by configuring the ROM_DIV field (bits 9:8) of the MSTGCR register of the system module. Check the device datasheet for the maximum supported PBIST ROM clock frequency.
3. Enable PBIST Controller by setting bit 1 of MSIENA register in system module.
4. Enable the PBIST self-test by writing a value of 0x0A to bits 3:0 of the MSTGCR in the system module.
5. Wait for N VBUS clock cycles based on the HCLK to PBIST ROM clock ratio:

   N = 16 when HCLK:PBIST ROM clock is 1:1
   N = 32 when HCLK:PBIST ROM clock is 1:2
   N = 64 when HCLK:PBIST ROM clock is 1:4
   N = 64 when HCLK:PBIST ROM clock is 1:8
6. Write 1h to PACT register to enable the PBIST internal clocks.
7. Program the ALGO register to decide which algorithm from the instruction ROM must be selected (the default value of ALGO register is all 1's, meaning all algorithms are selected). Similarly, program the RINFOL and RINFOU registers to indicate whether a particular RAM group in the instruction ROM would get executed or not.

NOTE: In case of RAM Override (Override Register (OVER) = 00), the user should make sure that only the algorithms that run on similar RAMs are selected. If a single port algorithm is selected in ROM Algorithm Mask Register (ALGO), the RAM Info Mask Lower Register (RINFOL) and RAM Info Mask Upper Register (RINFOU) must select only the single port RAM's. The same applies for two port RAM's. Check Architecture chapter for information on the memory types.

8. Program OVER = 1h to run PBIST self-test without RAM override. Program OVER = 0 to run PBIST self-test with RAM Override.
9. Write a value of 3h to the ROM mask register should the microcode for the Algorithms as well as the RAM groups loaded from the on-chip PBIST ROM.
10. Write DLR (Data Logger register) with 14h to configure the PBIST run in ROM mode and to enable the configuration access. This starts the memory self-tests.
11. Wait for the PBIST self-test done by polling MSTDONE bit of MSTCGSTAT register in System Module.
12. Once self-test is completed, check the Fail Status register FSRF0.

    In case there is a failure (FSRF0 = 1h):
    a. Read RAMT register that indicates the RGS and RDS values of the failure RAM
    b. Read FSRC0 and FSRC1 registers that contains the failure count
    c. Read FSRA0 and FSRA1 registers that contains the address of first failure
    d. Read FSRDL0 and FSRDL1 registers that contains the failure data.
    e. Write a value of 2h to the STR register to resume the test.

    In case there is no failure (FSRF0 = 0) the memory self-test is completed.
    a. Disable the PBIST internal clocks by writing a 0 to the PACT register.
    b. Disable the PBIST self-test by writing a value of 5h to bits 3:0 of the MSTGCR in the system module.
13. Repeat steps 2 through 9 for subsequent runs with different RAM group and algorithm configurations.
14. After required Memory tests are completed, Resume or Start the Normal Application software.

NOTE: The contents of the selected memory before the test will be completely lost. User software must take care of data backup if required. Typically the PBIST tests are carried out at the beginning of Application software.

NOTE: Memory test fail information is reported in terms of RGS:RDS and not RAM GROUP. Check Table 2-5 for information on the RGS:RDS information applicable to each memory being tested.

7.4 Memory Test Algorithms on the On-chip ROM

This section provides a brief description for some of the test algorithms used for memory self-test.

1. **March13N:**
   - March13N is the baseline test algorithm for SRAM testing. It provides the highest overall coverage. The other algorithms provide additional coverage of otherwise missed boundary conditions of the SRAM operation.
   - The concept behind the general march algorithm is to indicate:
     - The bit cell can be written and read as both a 1 and a 0.
     - The bits around the bit cell do not affect the bit cell.
   - The basic operation of the march is to initialize the array to a know pattern, then march a different pattern through the memory.
   - Type of faults detected by this algorithm:
     - Address decoder faults
     - Stuck-At faults
     - Coupled faults
     - State coupling faults
     - Parametric faults
     - Write recovery faults
     - Read/write logic faults
2. **Map Column:**
   - The MAP COLUMN algorithm is used to identify bit line sensitivities in the memory array. The memory array is loaded with a row stripe pattern of all 1s in the first row followed by all 0s in the second row and repeated throughout the array. Then the values are read down each column on consecutive cycles. The pattern in memory is inverted and run the column reads again.
   - This particular pattern is looking for the following SRAM failure mechanisms:
     - Leakage due to a low resist path in a bit
     - An Open in the bit cell
     - Leakage on a BIT or BITN line
     - Miss-balance in the sense amp
     - Leakage in the sense
     - High resist in the sense amp
     - Failure of the pre-charge circuits after read operations
3. **Pre-Charge:**
   - The Pre-Charge algorithm exercises the pre-charge capability within the SRAM array. It is important to specifically target this issue as it is the only part of the analog portion of the SRAM that is frequency sensitive.
   - Similar to the MAP COLUMN algorithm, this algorithm works its way down the columns of the SRAM. However, unlike the MAP COLUMN, this algorithm sandwiches a write between two reads to force the worst-case conditions for the pre-charge circuits in the array.
   - This test will fail when an increase in system frequency nears the minimum access time of the array, at this boundary:
     - High voltage should operate better than low voltage.
     - Likewise, low temperature should operate better than high temperature.

4. **DOWN1a:**
   - The Down1 pattern forces the switching of all data bits and most address bits on consecutive read cycles. This is primarily a read/write test of the CPU/memory subsystem.
   - The aggressive writes target at-speed write failures.
   - It also targets row/column decode in the memory array.
   - Targets the sense amps and sense amp multiplexors.
   - Memory array output buffers.
   - This algorithm operates as follows:
     - Load 1st half of the memory under test with one pattern.
     - Load 2nd half of the memory under test with the bit-wise inverse of the pattern.
     - Alternate sequential reads sequences between one sequence starting at the beginning of the array and a second sequence starting at the end of the array.
     - Upon completion of the read back, invert the patterns in both halves of the array and repeat the above step.
     - Perform an aggressive write sequence by alternating writes between the bottom half of the memory upwards with a data pattern and the top half of the memory downwards with the inverse data pattern.
     - Invert the data pattern for the above two steps to perform another sequence of aggressive writes.
5. **DTXN2a:**

This algorithm is used to target the global column decode Logic.

7.5 PBIST Control Registers

PBIST controller uses configuration registers for programming the algorithm and its execution. All the configuration registers are memory mapped for access by the CPU through the Peripheral Bus interface. The base address for the control registers is FFFF E400h.

NOTE: There is no watchdog functionality implemented in the PBIST controller. If a bad code is executed, the PBIST runs forever. The PBIST controller does not guard against this situation.

Registers are accessible only when the clock to the PBIST controller is active. The clock is activated by first writing 1h to the PACT register.

Table 7-1. PBIST Registers

Offset	Acronym	Register Description	Section
000h - 15Ch	Reserved	Reserved locations. Do not write to these locations.	
160h	RAMT	RAM Configuration Register	Section 7.5.1
164h	DLR	Datalogger Register	Section 7.5.2
168h - 17Ch	Reserved	Reserved locations. Do not write to these locations.	
180h	PACT	PBIST Activate/Clock Enable Register	Section 7.5.3
184h	PBISTID	PBIST ID Register	Section 7.5.4
188h	OVER	Override Register	Section 7.5.5
190h	FSRF0	Fail Status Fail Register 0	Section 7.5.6
198h	FSRC0	Fail Status Count Register 0	Section 7.5.7
19Ch	FSRC1	Fail Status Count Register 1	Section 7.5.7
1A0h	FSRA0	Fail Status Address 0 Register	Section 7.5.8
1A4h	FSRA1	Fail Status Address 1 Register	Section 7.5.8
1A8h	FSRDL0	Fail Status Data Register 0	Section 7.5.9
1B0h	FSRDL1	Fail Status Data Register 1	Section 7.5.9
1C0h	ROM	ROM Mask Register	Section 7.5.10
1C4h	ALGO	ROM Algorithm Mask Register	Section 7.5.11
1C8h	RINFOL	RAM Info Mask Lower Register	Section 7.5.12
1CCh	RINFOU	RAM Info Mask Upper Register	Section 7.5.13

7.5.1 RAM Configuration Register (RAMT)

This register is divided into the following internal registers, none of which have a default value after reset. Figure 7-3 and Table 7-2 illustrate this register.

This register provides the information regarding the memory being currently tested. In case of a PBIST failure, the application can read this register to identify the RGS:RDS values for the memory that failed the self-test.

Figure 7-3. RAM Configuration Register (RAMT) [offset = 0160h]

31	24	23	16
RGS		RDS	
R/W-X		R/W-X	

15	8	7	6	5	2	1	0
DWR		SMS		PLS		RLS	
R/W-X		R/W-X		R/W-X		R/W-X	

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

Table 7-2. RAM Configuration Register (RAMT) Field Descriptions

Bit	Field	Description
31-24	RGS	Ram Group Select. Refer to Table 2-5 for information on the RGS value for each memory.
23-16	RDS	Return Data Select. Refer to Table 2-5 for information on the RDS values for each memory.
15-8	DWR	Data Width Register
7-6	SMS	Sense Margin Select Register
5-2	PLS	Pipeline Latency Select
1-0	RLS	RAM Latency Select

7.5.2 Datalogger Register (DLR)

This register puts the PBIST controller into the appropriate comparison modes for data logging. Figure 7-4 and Table 7-3 illustrate this register.

Figure 7-4. Datalogger Register (DLR) [offset = 0164h]

31		16
	Reserved	
	R-0	

15	5	4	3	2	1	0
Reserved		DLR4	Rsvd	DLR2	Reserved	
R-0		R/W-0	R/W-1	R/W-0	R/W-0	

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

Table 7-3. Datalogger Register (DLR) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reads return 0. Do not change these bits from their default value.
4	DLR4		Config access: setting this bit allows the host processor to configure the PBIST controller registers.
3	Reserved	1	Do not change this bit from its default value of 1.
2	DLR2		ROM-based testing: setting this bit enables the PBIST controller to execute test algorithms that are stored in the PBIST ROM.
1-0	Reserved	00	Do not change these bits from their default value of 00.

- **DLR2: ROM-based testing mode**

Writing a 1 to this register starts the ROM-based testing. This register is used to initiate ROM-based testing from Config and ATE interfaces. Also, since a 1 in this bit position means the instruction ROM is used for memory testing, all the intermediate interrupts and PBIST done signal after each memory test are masked until all the selected algorithms in the ROM are executed for all RAM groups. However, a failure would stop the test and report the status immediately.

- **DLR4: Config access mode**

This mode, when set, indicates the CPU is being used to access PBIST.

7.5.3 PBIST Activate/Clock Enable Register (PACT)

This is the first register that needs to be programmed to activate the PBIST controller. Bit [0] is used for static clock gating, and unless a 1 is written to this bit, all the internal PBIST clocks are shut off. Figure 7-5 and Table 7-4 illustrate this register.

NOTE: This register must be programmed to 1h during application self-test.

Figure 7-5. PBIST Activate/ROM Clock Enable Register (PACT) [offset = 0180h]

31	16
Reserved	
R-0	

15	1	0
Reserved		PACT0
R-0		R/W-0

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

Table 7-4. PBIST Activate/ROM Clock Enable Register (PACT) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	PACT0		PBIST internal clocks enable.
		0	Disable PBIST internal clocks.
		1	Enable PBIST internal clocks.

- **PACT0**

This bit must be set to 1 to turn on the PBIST internal clocks. Setting this bit asserts an internal signal that is used as the clock gate enable. As long as this bit is 0, any access to the PBIST will not go through and the PBIST will remain in an almost zero-power mode.

7.5.4 PBIST ID Register

Functionality of this register is described in Figure 7-6 and Table 7-5.

Figure 7-6. PBIST ID Register [offset = 184h]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		PBIST ID	
R-0		R/W-0	

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

Table 7-5. PBIST ID Register Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	PBIST ID		This is a unique ID assigned to each PBIST controller in a device with multiple PBIST controllers.

7.5.5 Override Register (OVER)

Functionality of the register is described in Figure 7-7 and Table 7-6.

Figure 7-7. Override Register (OVER) [offset = 0188h]

31–16
Reserved
R-0

15–3	2	1	0
Reserved	Reserved		OVER0
R-0	R-0	R-0	R/W-1

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

Table 7-6. Override Register (OVER) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	Reserved	0	Reserved. This bit must not be changed from its default value of 0.
1	Reserved	0	Reserved. This bit must not be changed from its default value of 0.
0	OVER0		RINFO Override Bit
		0	The RAM info registers RINFOL and RINFOU are used to select the memories for test.
		1	The memory information available from ROM will override the RAM selection from the RAM info registers RINFOL and RINFOU.

- **OVER0**

While doing ROM-based testing, each algorithm downloaded from the ROM has a memory mask associated with it that defines the applicable memory groups the algorithm will be run on. By default, this bit is set to 1, which means the memory mask that is downloaded from the ROM will overwrite the RAM info registers. The override bit can be reset by writing a 0 to it. In this case, the application can select the RAM groups to be tested by configuring the RAM info registers.

NOTE: When this override bit = 0, each algorithm selected in ALGO register will run on each RAM selected in RINFOL and RINFOU register. It must be ensured that:

1. Only the same type of memories (single port or two port) are selected, and
2. Only memories that are valid for all algorithms enabled via the ALGO register are selected.

If the above two requirements are not met, the memory self-test will fail.

7.5.6 Fail Status Fail Register (FSRF0)

This register indicates if a Port0 failure occurred during a memory self-test. Bit [0] gets set whenever a failure occurs. Functionality of the register is described in Figure 7-8 and Table 7-7.

Figure 7-8. Fail Status Fail Register 0 (FSRF0) [offset = 0190h]

31	16
Reserved	
R-0	

15	1	0
Reserved		FSRF0
R-0		R-0

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

Table 7-7. Fail Status Fail Register 0 (FSRF0) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	FSRF0		Fail Status 0. This bit would be cleared by reset of the module using MSTGCR register in system module.
		0	No failure occurred.
		1	Failure occurred on port 0.

7.5.7 Fail Status Count Registers (FSRC0 and FSRC1)

These registers keep count of the number of failures observed during the memory self-test. The PBIST controller stops executing the memory self-test whenever a failure occurs in any memory instance for any of the test algorithms. The value in FSRC0 / FSRC1 gets incremented by one whenever a failure occurs and gets decremented by one when the failure is processed. FSRC0 is for Port 0 and FSRC1 is for Port 1. Figure 7-9 and Table 7-8 illustrate the FSRC0 register, while Figure 7-10 and Table 7-9 illustrate the FSRC1 register.

Figure 7-9. Fail Status Count 0 Register (FSRC0) [offset = 0198h]

31	16
Reserved	
R-0	

15 8	7 0
Reserved	FSRC0
R-0	R-0

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

Table 7-8. Fail Status Count 0 Register (FSRC0) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	FSRC0		Fail Status Count 0. Indicates the number of failures on port 0.

Figure 7-10. Fail Status Count Register 1 (FSRC1) [offset = 019Ch]

31	16
Reserved	
R-0	

15 8	7 0
Reserved	FSRC1
R-0	R-0

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

Table 7-9. Fail Status Count Register 1 (FSRC1) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	FSRC1		Fail Status Count 1. Indicates the number of failures on port 1.

7.5.8 Fail Status Address Registers (FSRA0 and FSRA1)

These registers capture the memory address of the first failure on port 0 and port 1, respectively. Figure 7-11 and Table 7-10 illustrate the FSRA0 register, while Figure 7-12 and Table 7-11 illustrate the FSRA1 register.

Figure 7-11. Fail Status Address 0 Register (FSRA0) [offset = 01A0h]

31	16
Reserved	
R-0	

15	0
FSRA0	
R-0	

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

Table 7-10. Fail Status Address 0 Register (FSRA0) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	FSRA0		Fail Status Address 0. Contains the address of the first failure.

Figure 7-12. Fail Status Address 1 Register (FSRA1) [offset = 01A4h]

31	16
Reserved	
R-0	

15	0
FSRA1	
R-0	

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

Table 7-11. Fail Status Address 1 Register (FSRA1) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	FSRA1		Fail Status Address 1. Contains the address of the first failure.

7.5.9 Fail Status Data Registers (FSRDL0 and FSRDL1)

These registers are used to capture the failure data in case of a memory self-test failure. FSRDL0 corresponds to Port 0, while FSRDL1 corresponds to Port 1. Figure 7-13 and Table 7-12 illustrate the FSRDL0 register, while Figure 7-14 and Table 7-13 illustrate the FSRDL1 register.

Figure 7-13. Fail Status Data Register 0 (FSRDL0) [offset = 01A8h]

31	16
FSRDL0	
R-AAAAh	
15	0
FSRDL0	
R-AAAAh	

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

Table 7-12. Fail Status Data Register 0 (FSRDL0) Field Descriptions

Bit	Field	Description
31-0	FSRDL0	Failure data on port 0

Figure 7-14. Fail Status Data Register 1 (FSRDL1) [offset = 01B0h]

31	16
FSRDL1	
R-AAAAh	
15	0
FSRDL1	
R-AAAAh	

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

Table 7-13. Fail Status Data Register 1 (FSRDL1) Field Descriptions

Bit	Field	Description
31-0	FSRDL1	Failure data on port 1

7.5.10 ROM Mask Register (ROM)

This two-bit register sets appropriate ROM access modes for the PBIST controller. The default value is 11b. This register is illustrated in Figure 7-15. It can be programmed according to Table 7-14.

Figure 7-15. ROM Mask Register (ROM) [offset = 01C0h]

31	16
Reserved	
R-0	

15	2	1	0
Reserved		ROM	
R-0		R/W-3h	

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

Table 7-14. ROM Mask Register (ROM) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reads return 0. Writes have no effect.
1-0	ROM		ROM Mask
		0	No information is used from ROM.
		1h	Only RAM Group information from ROM.
		2h	Only Algorithm information from ROM.
		3h	Both Algorithm and RAM Group information from ROM. This option should be selected for application self-test.

7.5.11 ROM Algorithm Mask Register (ALGO)

This register is used to indicate the algorithm(s) to be used for the memory self-test routine. Each bit corresponds to a specific algorithm. For example, bit [0] controls whether algorithm 1 is enabled or not. Figure 7-16 and Table 7-15 illustrate this register.

Figure 7-16. ROM Algorithm Mask Register (ALGO) [offset = 01C4h]

31	24	23	16
ALGO3		ALGO2	
R/W-FFh		R/W-FFh	

15	8	7	0
ALGO1		ALGO0	
R/W-FFh		R/W-FFh	

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

Table 7-15. Algorithm Mask Register (ALGO) Field Descriptions

Bit	Field	Value	Description
31		0	Algorithm 32 is not selected.
		1	Selects algorithm 32 for PBIST run.
30		0	Algorithm 31 is not selected.
		1	Selects algorithm 31 for PBIST run.
:			
0		0	Algorithm 1 is not selected.
		1	Selects algorithm 1 for PBIST run.
31-0		0	None of the algorithms are selected.

NOTE: Please refer to for available algorithms and the memories on which each algorithm can be run.

7.5.12 RAM Info Mask Lower Register (RINFOL)

This register is to select RAM groups to run the algorithms selected in the ALGO register. For an algorithm to be executed on a particular RAM group, the corresponding bit in this register must be set to 1. The default value of this register is all 1s, which means all the RAM Groups are selected. Figure 7-17 and Table 7-16 illustrate this register.

The information from this register is used only when bit 0 in OVER register is not set.

Figure 7-17. RAM Info Mask Lower Register (RINFOL) [offset = 01C8h]

31 24	23 16
RINFOL3	RINFOL2
R/W-FFh	R/W-FFh
15 8	7 0
RINFOL1	RINFOL0
R/W-FFh	R/W-FFh

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

Table 7-16. RAM Info Mask Lower Register (RINFOL) Field Descriptions

Bit	Field	Value	Description
31		0 1	RAM Group 32 is not selected. Selects group 32 for PBIST run.
30		0 1	RAM Group 31 is not selected. Selects RAM group 31 for PBIST run.
:			
0		0 1	RAM Group 1 is not selected. Selects RAM Group 1 for PBIST run.
31-0		0	None of the RAM Groups 1 to 32 are selected.

NOTE: Please refer to Table 2-5 for RAM info groups.

7.5.13 RAM Info Mask Upper Register (RINFOU)

This register is to select RAM groups to run the algorithms selected in the ALGO register. For an algorithm to be executed on a particular RAM group, the corresponding bit in this register should be set to 1. The default value of this register is all 1s, which means all the RAM Info Groups would be selected. Figure 7-18 and Table 7-17 illustrate this register.

Figure 7-18. RAM Info Mask Upper Register (RINFOU) [offset = 01CCh]

31 24	23 16
RINFOU3	RINFOU2
R/W-FFh	R/W-FFh

15 8	7 0
RINFOU1	RINFOU0
R/W-FFh	R/W-FFh

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

Table 7-17. RAM Info Mask Upper Register (RINFOU) Field Descriptions

Bit	Field	Value	Description
31		0	RAM Group 64 is not selected.
		1	Selects group 64 for PBIST run.
30		0	RAM Group 63 is not selected.
		1	Selects RAM group 63 for PBIST run.
:			
0		0	RAM Group 33 is not selected.
		1	Selects RAM Group 33 for PBIST run.
31-0		0	None of RAM Groups 33 to 64 are selected.

7.6 PBIST Configuration Example

The following examples assume that the PLL is locked and selected as clock source with HCLK = 160 MHz and VCLK = 80 MHz.

7.6.1 Example 1 : Configuration of PBIST Controller to Run Self-Test on RAM Group 3

This example explains the configurations for running March13, Down1A and Map Column algorithms on RAM Group 3 (see device datasheet for RAM Group information).

1. Program the HCLK to PBIST ROM clock ratio to 1:2 in System Module.

   MSTGCR[9:8] = 1
2. Enable PBIST Controller in System Module.

   MSIENA[31:0] = 0x00000001
3. Enable the PBIST self-test in System Module.

   MSTGCR[3:0] = 0xA
4. Wait for at least 32 VCLK cycles in a software loop.
5. Enable the PBIST internal clocks.

   PACT = 0x1
6. Disable RAM Override. This will make the PBIST controller use the information provided by the application in the RINFOx and ALGO registers for the memory self-test.

   OVER = 0x0
7. Select the Algorithm (refer to).

   ALGO = 0x00000054 (Algo 3 = March13N, Algo 5 = down1A_red, Algo 7 = Map column for two-port RAM Group 3)
8. Program the RAM group Info to select RAM Group 3 (refer to Table 2-5).

   RINFOL = 0x00000004 (select RAM Group 3)
   RINFOU = 0x00000000 (since this device supports only 28 RAM Groups)
9. Select both Algorithm and RAM information from on-chip PBIST ROM.

   ROM = 0x3
10. Configure PBIST to run in ROM Mode and start PBIST run.

    DLR = 0x14
11. Wait for PBIST test to complete by polling MSTDONE bit in System Module.

    while (MSTDONE !=1)
12. Once self-test is completed, check the Fail Status register FSRF0.
    a. In case there is a failure (FSRF0 = 0x01):
       i. Read RAMT register that indicates the RGS and RDS values of the failure RAM.
       ii. Read FSRC0 and FSRC1 registers that contains the failure count.
       iii. Read FSRA0 and FSRA1 registers that contains the address of first failure.
       iv. Read FSRDL0 and FSRDL1 registers that contains the failure data.
       v. Resume the Test if required using Program Control register (offset = 0x16C) STR = 2.
    b. In case there is no failure (FSRF0 = 0x00), the memory self-test is completed:
       i. Disable the PBIST internal clocks.

          PACT = 0
       ii. Disable the PBIST self-test.

          MSTGCR[3:0] = 0x5

7.6.2 Example 2 : Configuration of PBIST Controller to Run Self-Test on ALL RAM Groups

This example explains the configurations for running March13, Down1A and Mapcolumn algorithms on all RAM groups defined in the PBIST ROM.

1. Program the HCLK to PBIST ROM clock ratio to 1:2 in System Module.
   MSTGCR[9:8] = 1
2. Enable PBIST Controller in System Module.
   MSIENA[31:0] = 0x00000001
3. Enable the PBIST self-test in System Module.
   MSTGCR[3:0] = 0xA
4. Wait for at least 32 VCLK cycles in a software loop.
5. Enable the PBIST internal clocks.
   PACT = 0x1
6. Enable RAM Override.
   OVER = 0x1
7. Select the Algorithms to be run (refer to).
   ALGO = 0x000000FC (select March13N, Down1A and Map Column algorithms for single-port and two-port RAMs)
8. Select both Algorithm and RAM information from on-chip PBIST ROM.
   ROM = 0x3
9. Configure PBIST to run in ROM Mode and kickoff PBIST test.
   DLR = 0x14
10. Wait for PBIST test to complete by polling MSTDONE bit in System Module.
    while (MSTDONE !=1)
11. Once self-test is completed, check the Fail Status register FSRF0:
    a. In case there is a failure (FSRF0 = 0x01):
       i. Read RAMT register that indicates the RGS and RDS values of the failure RAM.
       ii. Read FSRC0 and FSRC1 registers that contains the failure count.
       iii. Read FSRA0 and FSRA1 registers that contains the address of first failure.
       iv. Read FSRDL0 and FSRDL1 registers that contains the failure data.
       v. Resume the Test if required using Program Control register (offset = 0x16C) STR = 2.
    b. In case there is no failure (FSRF0 = 0x00), the memory self-test is completed:
       i. Disable the PBIST internal clocks.
          PACT = 0
       ii. Disable the PBIST self-test.
          MSTGCR[3:0] = 0x5

Chapter 8

SPNU514C – March 2018

CPU Self-Test Controller (STC) Module

This chapter describes the basics and configuration of the CPU self-test controller present in the device.

Topic **Page**

8.1 General Description 370

8.2 Application Self-Test Flow 372

8.3 STC Test Coverage and Duration 374

8.4 STC Control Registers 375

8.5 STC Configuration Example 384

8.1 General Description

The CPU self-test controller (STC) is used to test the ARM-CPU core using the Deterministic Logic Built-in Self-Test (LBIST) Controller as the test engine. To achieve better coverage for the self-test of complex cores like Cortex-R4, on-chip logic BIST is the preferred solution.

8.1.1 CPU Self-Test Controller Features

The CPU self-test controller has the following features:

- Capable of running the complete test as well as running a few intervals at a time
  - Ability to continue from the last executed interval (test set) as well as the ability to restart from the beginning (first test set)
  - Total of 24 intervals supported in this device
- Complete isolation of the self-tested CPU core from the rest of the system during the self-test run
  - The self-tested CPU core master bus transaction signals are configured to be in idle mode during the self-test run
  - Any master access to the CPU core under self-test (example: DMA access to CPU TCM) will be held until the completion of the self-test
- Ability to capture the failure interval number
- Timeout counter for the CPU self-test run as a fail-safe feature
- Able to read the MISR data (shifted from LBIST controller) of the last executed interval of the self-test run for debugging purposes
- STCCLK determines the self-test execution speed, STC clock divider (STCCLKDIV) register in the system module is used to divide HCLK (system clock) to generate STCCLK

8.1.2 STC Block Diagram

STC module provides an interface to the LBIST controller implemented on the core.

The CPU STC is composed of following blocks of logic:

- ROM Interface
- FSM and Sequence Control
- Register Block
- Peripheral Bus Interface (VBUSP Interface)
- STC Bypass/ATE Interface

8.1.2.1 ROM Interface

This block handles the ROM address and control signal generation to read the self-test microcode from the ROM. The test microcode and golden signature value for each interval are stored in ROM.

8.1.2.1.1 FSM and Sequence Control

This block generates the signals and data to the LBIST controller based on the seed, test_type and scan chain depth.

8.1.2.1.2 Clock Control

The CLOCK CNTRL sub-block handles the internal clock selection and clock generation for the ROM and LBIST controller.

8.1.2.2 Register Block

This block handles the control of the self-test controller. This block contains various configuration and status registers that provide the result of a self-test run. These registers are memory-mapped and accessible through the Peripheral Bus (VBUSP) Interface. This block controls the reseeding (reloading the existing seed of the PRPG) in the LBIST controller.

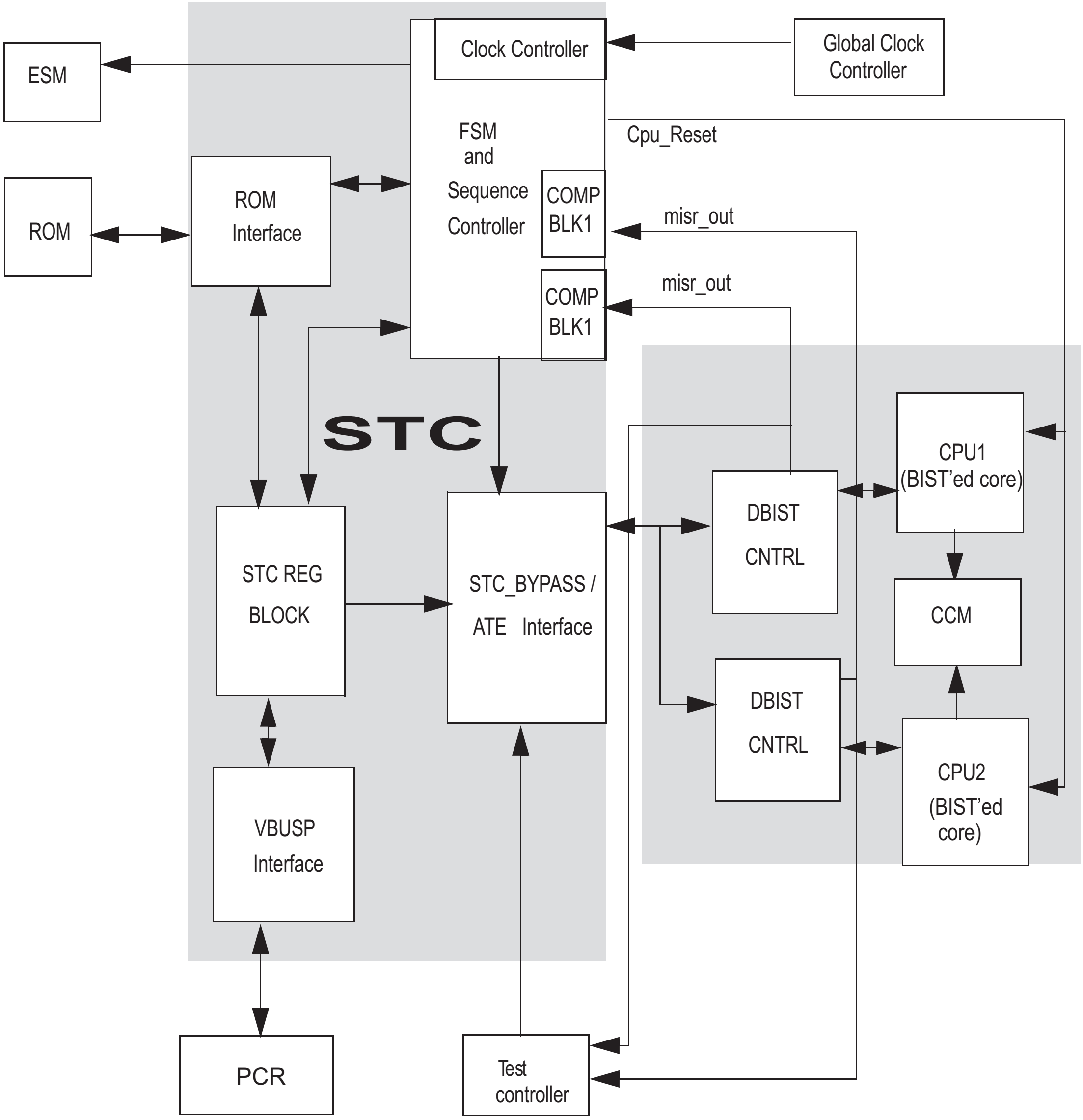
8.1.2.3 STC Bypass / ATE Interface

This is a production test interface. Only for TI internal use.

8.1.2.4 Peripheral Bus (VBUSP) Interface

STC control registers are accessed through Peripheral Bus (VBUSP) Interface. During application programming, configuration registers are programmed through the Peripheral Bus Interface to enable and run the self-test controller.

Figure 8-1. STC Block Diagram



8.2 Application Self-Test Flow

This section describes the STC module configuration and the application self-test flow that the user should follow for successful execution. The following two configurations must be part of the STC initialization code:

- STC clock rate configuration, STC clock divider (STCCLKDIV) register in system module is used to divide HCLK (system clock) to generate STCCLK
- Clear SYSESR register before triggering an STC test

8.2.1 STC Module Configuration

- Configure the test interval count using STCGCR0[31:16] register. A maximum of 24 intervals are supported in the device. You can run 24 intervals together or in slices. If the tests are run in slices, the user software can specify to the self-test controller whether to continue the run from the next interval onwards or to restart from interval 0 using bit STCGCR0[0]. This bit gets reset after the completion of the self-test run.
- Configure self-test run timeout counter preload register STCTPR. This register contains the total number of VBUS clock cycles it will take before a self-test timeout error (TO_ERR) will be triggered after the initiation of the self-test run.
- Enable CPU self-test by writing the enable key to STCGCR1 register.

8.2.2 Context Saving

STC generates a CPU reset after completion of the test regardless of pass or fail. You can run the STC test during startup or can divide STC into 24 or fewer intervals and run them during normal operation.

If STC is ran only on startup, the user software need not save the CPU contents since the reset caused will go through all startup configurations. You should check the STCGSTAT register for the self-test status before going to the application software.

If STC is divided into intervals and ran, user software must save the CPU contents and reload them after the CPU reset caused by the completion of the STC test interval. The check for STC status should bypass STC run if the reset is caused by an STC run to prevent a cyclic reset, that is, if reset is caused by STC the second time through, then it should not be ran again. You should also check the STCGSTAT register for the self-test status before restoring the application software.

Following are some of the registers that are required to be backed up before and restored after self-test:

1. CPU core registers (all modes R0-R15, PC, CPSR)
2. CP15 System Control Coprocessor registers - MPU control and configuration registers, Auxiliary Control Register used to Enable ECC, Fault Status Register etc.
3. CP13 Coprocessor Registers - FPU configuration registers, General Purpose Registers
4. Hardware Break Point and watch point registers like BVR, BSR, WVR, WSR etc.

For more information on the CPU reset, refer to the ARM® Cortex®-R4F Technical Reference Manual.

NOTE: Check all reset source flags in the SYSESR register after a CPU BIST execution. If a flag, in addition to CPU reset, is set, clear the CPU reset flag and service the other reset sources accordingly.

8.2.3 Entering CPU Idle Mode

After enabling the STC test by writing the STC enable key, the test is triggered only after the CPU is taken to idle mode by executing the CPU Idle Instruction **asm(“ WFI”).**

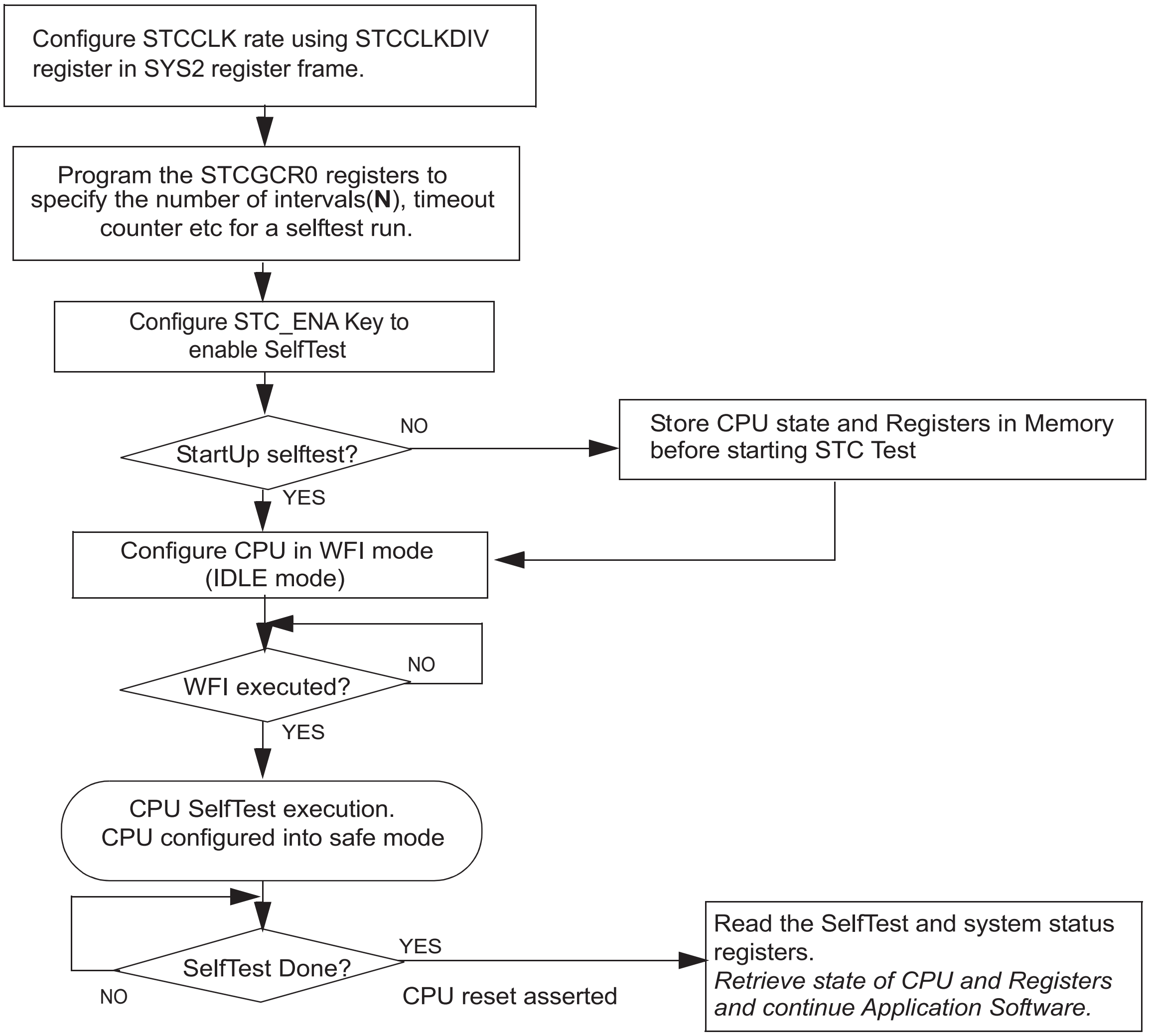
8.2.4 Self-Test Completion and Error Generation

At the end of each interval, the 128 bit MISR value (reflected in registers CPUx_CURMISR[3:0]) from the DBIST controller is shifted into the STC. This is compared with the golden MISR value stored in the ROM.

At the end of a CPU self-test, the STC controller updates the status flags in the Global Status Register (STCGSTAT) and resets the CPU. In case of a MISR mismatch or a test timeout, an error is generated through the ESM module. TEST_ERR signal is asserted when an MISR miss-compare occurs during the self-test. A TO_ERR is asserted when a timeout occurs during the self-test, meaning the test could not complete within the time specified in the timeout counter preload register STCTPR. However, at the device level, these two errors are combined and mapped to a single ESM channel. To identify which error occurred, user software must check the global status register (STCGSTAT) and fail status register STCFSTAT in the ESM interrupt service routine.

Figure 8-2 illustrates the application self-test test flow chart, drawn based on the assumption that the device has gone through startup, necessary clocks initialized and SYSESR register bits cleared.

Figure 8-2. Application Self-Test Flow Chart


Configure STCCLK rate using STCCLKDIV register in SYS2 register frame.
Program the STCGCR0 registers to specify the number of intervals(N), timeout counter etc for a selftest run.
Configure STC_ENA Key to enable SelfTest
StartUp selftest?
NO
Store CPU state and Registers in Memory before starting STC Test
YES
Configure CPU in WFI mode (IDLE mode)
WFI executed?
NO
YES
CPU SelfTest execution. CPU configured into safe mode
SelfTest Done?
YES
NO
CPU reset asserted
Read the SelfTest and system status registers.
Retrieve state of CPU and Registers and continue Application Software.

8.3 STC Test Coverage and Duration

The test coverage and number of test execution cycles (STCCLK) for each test interval when the device is running at HCLK = 180 MHz, VCLK = 90 MHz, and STCCLK = 90 MHz are shown in Table 8-1.

Table 8-1. STC Test Coverage and Duration

Intervals	Test Coverage (%)	Test Time (Cycles)	Test Time (µs)
0	0	0	0
1	62.13	1365	15.17
2	70.09	2730	30.33
3	74.49	4095	45.50
4	77.28	5460	60.67
5	79.28	6825	75.83
6	80.90	8190	91.00
7	82.02	9555	106.17
8	83.10	10920	121.33
9	84.08	12285	136.50
10	84.87	13650	151.67
11	85.59	15015	166.83
12	86.11	16380	182.00
13	86.67	17745	197.17
14	87.16	19110	212.33
15	87.61	20475	227.50
16	87.98	21840	242.67
17	88.38	23205	257.83
18	88.69	24570	273.00
19	88.98	25935	288.17
20	89.28	27300	303.33
21	89.50	28665	318.50
22	89.76	30030	333.67
23	90.01	31395	348.83
24	90.21	32760	364.00

Table 8-2 gives the typical STC execution times for 24 intervals at different clock rates.

Table 8-2. Typical STC Execution Times

Number of Intervals	@ HCLK = 180 MHz
	VCLK = 90 MHz
	STCCLK = 90 MHz
		@ HCLK = 100 MHz
		VCLK = 100 MHz
		STCCLK = 50MHz
			@ HCLK = 160 MHz
			VCLK = 80 MHz
			STCCLK = 80 MHz

24	364 µs	655.20 µs	409.50 µs

8.4 STC Control Registers

STC control registers are accessed through Peripheral Bus (VBUSP) interface. Read and write access in 8, 16, and 32 bit are supported. The base address for the control registers is FFFF E600h.

NOTE: In suspend mode, all registers can be written irrespective of user or privilege mode.

Table 8-3. STC Control Registers

Offset	Acronym	Register Description	Section
00h	STCGCR0	STC Global Control Register 0	Section 8.4.1
04h	STCGCR1	STC Global Control Register 1	Section 8.4.2
08h	STCTPR	Self-Test Run Timeout Counter Preload Register	Section 8.4.3
0Ch	STC_CADDR	STC Current ROM Address Register	Section 8.4.4
10h	STCCICR	STC Current Interval Count Register	Section 8.4.5
14h	STCGSTAT	Self-Test Global Status Register	Section 8.4.6
18h	STCFSTAT	Self-Test Fail Status Register	Section 8.4.7
1Ch	CPU1_CURMISR3	CPU1 Current MISR Register	Section 8.4.8
20h	CPU1_CURMISR2	CPU1 Current MISR Register	Section 8.4.8
24h	CPU1_CURMISR1	CPU1 Current MISR Register	Section 8.4.8
28h	CPU1_CURMISR0	CPU1 Current MISR Register	Section 8.4.8
2Ch	CPU2_CURMISR3	CPU2 Current MISR Register	Section 8.4.9
30h	CPU2_CURMISR2	CPU2 Current MISR Register	Section 8.4.9
34h	CPU2_CURMISR1	CPU2 Current MISR Register	Section 8.4.9
38h	CPU2_CURMISR0	CPU2 Current MISR Register	Section 8.4.9
3Ch	STCSCSCR	Signature Compare Self-Check Register	Section 8.4.10

8.4.1 STC Global Control Register 0 (STCGCR0)

This register is described in Figure 8-3 and Table 8-4.

Figure 8-3. STC Global Control Register 0 (STCGCR0) [offset = 00]

31	16
INTCOUNT	
R/W-1	

15	1	0
Reserved		RS_CNT
R-0		R/WP-0

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

Table 8-4. STC Global Control Register 0 (STCGCR0) Field Descriptions

Bit	Field	Value	Description
31-16	INTCOUNT		Number of intervals of self-test run This register specifies the number of intervals to run for the self-test run. This corresponds to the number of intervals to be ran from the value reflected in the current interval counter.
15-1	Reserved	0	Read returns 0. Writes have no effect.
0	RS_CNT		Restart or Continue This bit specifies whether to continue the run from next interval onwards or to restart from interval 0. This bit gets reset after the completion of a self-test run.
		0	Continue STC run from the previous interval.
		1	Restart STC run from interval 0.

NOTE: On a power-on reset or system reset, this register gets reset to its default values.

8.4.2 STC Global Control Register 1 (STCGCR1)

This register is described in Figure 8-4 and Table 8-5.

Figure 8-4. STC Global Control Register 1 (STCGCR1) [offset = 04h]

31	16
Reserved	
R-0	

15	4	3	0
Reserved		STC_ENA	
R-0		R/WP-5h	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after nPORST (power-on reset) or System reset

Table 8-5. STC Global Control Register 1 (STCGCR1) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	STC_ENA		Self-test run enable key
		Ah	Self-test run is enabled.
		All Others	Self-test run is disabled.

NOTE: On a power-on reset or system reset, this register resets to its default values. Also, this register automatically resets to its default values at the completion of a self-test run.

8.4.3 Self-Test Run Timeout Counter Preload Register (STCTPR)

This register is described in Figure 8-5 and Table 8-6.

Figure 8-5. Self-Test Run Timeout Counter Preload Register (STCTPR) [offset = 08h]

31 … 16
RTOD
R/WP-FFFFh

15 … 0
RTOD
R/WP-FFFFh

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after nPORST (power-on reset) or System reset

Table 8-6. Self-Test Run Timeout Counter Preload Register (STCTPR)

Bit	Field	Description
31-0	RTOD	Self-test timeout count preload This register contains the total number of VBUS clock cycles it will take before an self-test timeout error (TO_ERR) will be triggered after the initiation of the self-test run. This is a fail safe feature to prevent the device from hanging up due to a run away test during the self-test. The preload count value gets loaded into the self-test time out down counter whenever a self-test run is initiated (STC_KEY is enabled) and gets disabled on completion of a self-test run.

NOTE: On a power-on reset or system reset, this register gets reset to its default values.

8.4.4 STC Current ROM Address Register (STC_CADDR)

This register is described in Figure 8-6 and Table 8-7.

Figure 8-6. STC Current ROM Address Register (STC_CADDR) [offset = 0Ch]

31 … 16
ADDR
R-0

15 … 0
ADDR
R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after nPORST (power-on reset) or System reset

Table 8-7. STC Current ROM Address Register (STC_CADDR) Field Descriptions

Bit	Field	Description
31-0	ADDR	Current ROM Address This register reflects the current ROM address (for micro code load) which is the current value of the STC program counter.

NOTE: When the RS_CNT bit in STCGCR0 is set to a 1 on the start of a self-test run, or on a power-on reset or system reset, this register resets to all zeroes.

8.4.5 STC Current Interval Count Register (STCCICR)

This register is described in Figure 8-7 and Table 8-8.

Figure 8-7. STC Current Interval Count Register (STCCICR) [offset = 10h]

31	16
Reserved	
R-0	

15	0
N	
R-0	

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

Table 8-8. STC Current Interval Count Register (STCCICR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-0	N		Interval Number This specifies the last executed interval number.

NOTE: When the RS_CNT bit in STCGCR0 is set to a 1 or on a power-on reset, the current interval counter resets to the default value.

8.4.6 Self-Test Global Status Register (STCGSTAT)

This register is described in Figure 8-8 and Table 8-9.

Figure 8-8. Self-Test Global Status Register (STCGSTAT) [offset = 14h]

31	8
Reserved	
R-0	

2	1	0
Reserved	TEST_FAIL	TEST_DONE
R-0	R/WP-0	R/WP-0

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

Table 8-9. Self-Test Global Status Register (STCGSTAT) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Read returns 0. Writes have no effect.
1	TEST_FAIL		Test Fail
		0	Self-test run has not failed.
		1	Self-test run has failed.
0	TEST_DONE		Test Done
		0	Not completed.
		1	Self-test run completed.
			The test done flag is set to a 1 for any of the following conditions: 1. When the STC run is complete without any failure 2. When a failure occurs on a STC run 3. When a timeout failure occurs Reset is generated to the CPU on which the STC run is being performed when TEST_DONE goes high (the test is completed).

NOTE: The two status bits can be cleared to their default values on a write of 1 to the bits. Additionally when the STC_ENA key is written from a disabled state to enabled state, the two status flags get cleared to their default values. This register gets reset to its default value with power-on reset assertion.

8.4.7 Self-Test Fail Status Register (STCFSTAT)

This register is described in Figure 8-9 and Table 8-10.

Figure 8-9. Self-Test Fail Status Register (STCFSTAT) [offset = 18h]

31	8
Reserved	
R-0	

3	2	1	0
Reserved	TO_ERR	CPU2_FAIL	CPU1_FAIL
R-0	R/WP-0	R/WP-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after nPORST (power-on reset) or System reset

Table 8-10. Self-Test Fail Status Register (STCFSTAT) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Read returns 0. Writes have no effect.
2	TO_ERR		Timeout Error
		0	No time out error occurred.
		1	Self-test run failed due to a timeout error.
1	CPU2_FAIL		CPU2 failure info
		0	No MISR mismatch for CPU2.
		1	Self-test run failed due to MISR mismatch for CPU2.
0	CPU1_FAIL		CPU1 failure info
		0	No MISR mismatch for CPU1.
		1	Self-test run failed due to MISR mismatch for CPU1.

NOTE: The three status bits can be cleared to their default values on a write of 1 to the bits. Additionally when the STC_ENA key in STCGCR1 is written from a disabled state to an enabled state, the three status bits get cleared to their default values. This register gets reset to its default value with power-on reset assertion.

8.4.8 CPU1 Current MISR Register (CPU1_CURMISR[3:0])

This register is described in Figure 8-10 through Figure 8-13 and Table 8-11.

Figure 8-10. CPU1 Current MISR Register (CPU1_CURMISR3) [offset = 1Ch]

31	16
MISR[31:16]	
R-0	

15	0
MISR[15:0]	
R-0	

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

Figure 8-11. CPU1 Current MISR Register (CPU1_CURMISR2) [offset = 20h]

31	16
MISR[63:48]	
R-0	

15	0
MISR[47:32]	
R-0	

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

Figure 8-12. CPU1 Current MISR Register (CPU1_CURMISR1) [offset = 24h]

31	16
MISR[95:80]	
R-0	

15	0
MISR[79:64]	
R-0	

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

Figure 8-13. CPU1 Current MISR Register (CPU1_CURMISR0) [offset = 28h]

31	16
MISR[127:112]	
R-0	

15	0
MISR[111:96]	
R-0	

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

Table 8-11. CPU1 Current MISR Register (CPU1_CURMISR[3:0]) Field Descriptions

Bit	Field	Description
127-0	MISR	MISR data from CPU1 This register contains the MISR data from the CPU1 for the most recent interval. This value is compared with the GOLDEN MISR value copied from ROM.

NOTE: This register gets reset to its default value with power-on or system reset assertion.

8.4.9 CPU2_CURMISR[3:0] (CPU2 Current MISR Register)

This register is described in Figure 8-14 through Figure 8-17 and Table 8-12.

Figure 8-14. CPU2 Current MISR Register (CPU2_CURMISR3) [offset = 2Ch]

31	16
MISR[31:16]	
R-0	

15	0
MISR[15:0]	
R-0	

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

Figure 8-15. CPU2 Current MISR Register (CPU2_CURMISR2) [offset = 30h]

31	16
MISR[63:48]	
R-0	

15	0
MISR[47:32]	
R-0	

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

Figure 8-16. CPU2 Current MISR Register (CPU2_CURMISR1) [offset = 34h]

31	16
MISR[95:80]	
R-0	

15	0
MISR[79:64]	
R-0	

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

Figure 8-17. CPU2 Current MISR Register (CPU2_CURMISR0) [offset = 38h]

31	16
MISR[127:112]	
R-0	

15	0
MISR[111:96]	
R-0	

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

Table 8-12. CPU2 Current MISR Register (CPU2_CURMISR[3:0]) Field Descriptions

Bit	Field	Description
127-0	MISR	MISR data from CPU2 This register contains the MISR data from the CPU2 for the most recent interval. This value is compared with the GOLDEN MISR value copied from ROM.

NOTE: This register gets reset to its default value with power-on or system reset assertion.

8.4.10 STCSCSCR (Signature Compare Self-Check Register)

This register is described in Figure 8-18. This register is used to enable the self-check feature of the CPU Self-Test Controller's (STC) signature compare logic. Self-check can only be done for the STC interval 0 by setting the RS_CNT bit in STCGCR0 to 1 to restart the self-test. The STC run will fail for signature miss-compare, provided the signature compare logic is operating correctly. To proceed with regular CPU self-test, STCSCSCR should be programmed to disable the self-check feature and clear the RS_CNT bit in STCGCR0 to 0. This register gets reset to its default value with any system reset assertion.

Figure 8-18. Signature Compare Self-Check Register (STCSCSCR) [offset = 3Ch]

31-16
Reserved
R-0

15-5	4	3-0
Reserved	FAULT_INS	SELF_CHECK_KEY
R-0	R/WP-0	R/WP-5h

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after nPORST (power-on reset) or System reset

Table 8-13. Signature Compare Self-Check Regsiter (STCSCSCR) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved		Reads return zeros, writes have no effect
4	FAULT_INS		Enable / Disable fault insertion.
		0	No fault is inserted.
		1	Insert stuck-at-fault inside CPU so that STC signature compare will fail.
3-0	SELF_CHECK_KEY		Signature compare logic self-check enable key
		Ah	Signature compare logic self-check is enabled. This allows a fault to be inserted using the FAULT_INS field.
		Any other value	Signature compare logic self-check is disabled The FAULT_INS field has no effect in this case.

8.5 STC Configuration Example

The following examples assume that the PLL is locked and selected as the system clock source with HCLK = 180 MHz and VCLK = 90 MHz.

8.5.1 Example 1: Self-Test Run for 24 Interval

This example explains the configurations for running STC Test for maximum Test Intervals 24.

1. Maximum STC clock rate support at 180 MHz HCLK is 90 MHz. Divide HCLK by 2 to achieve this clock rate. STCCLKDIV[26:24] register in the secondary system module frame at location 0xFFFF E108 is used.

   STCCLKDIV[26:24] = 1
2. Clear CPU_RST status bit in the System Exception Status Register in the system module.

   SYSESR[5] = 1
3. Configure the test interval count in STC module.

   STCGCR0[31:16] = 24
4. Configure self-test run time out counter preload register.

   STCTPR[31:0] = 0xFFFFFFFF
5. Enable CPU self-test.

   STCGCR1[3:0]= 0xA
6. Perform a context save of CPU state and configuration registers that get reset on CPU reset.
7. Put the CPU in idle mode by executing the CPU idle instruction.
   asm(“ WFI”)
8. Upon CPU reset, verify the CPU_RST status bit in the System Exception Status Register is set. This also verifies that no other resets occurred during the self-test.

   SYSESR[5] == 1
9. Check the STCGSTAT register for the self-test status.

   Check TEST_DONE bit before evaluating TEST_FAIL bit.

   If TEST_DONE = 0 the self-test is not completed. restart the STC test by going to Step 5.

   If (TEST_DONE = 1 and TEST_FAIL = 1) the self-test is completed and Failed.

   - Read STC Fail Status Register STCFSTAT[2:0] to identify the type of Failure (Timeout, CPU1 fail, CPU2 fail).

   In case there is no failure (TEST_DONE = 1 and TEST_FAIL = 0), the CPU self-test is completed successfully.

   - Recover the CPU status, configuration registers and continue the application software.

CPU Compare Module for Cortex-R4F (CCM-R4F)

This chapter describes the CPU compare module for Cortex-R4F (CCM-R4F). This device implements two instances of the Cortex-R4F CPU which are running in lock step to detect faults which may result in unsafe operating conditions. The CCM-R4F detects faults and signals them to an error signaling module (ESM).

NOTE: In general, R4F is used when referencing the Cortex CPU used in the Hercules family of devices; however, the floating-point functionality is a device-specific option and may not be included in some devices. Consult your device-specific datasheet to determine which core is included on your specific device being used.

Topic **Page**

9.1 Main Features **386**
9.2 Block Diagram **386**
9.3 Module Operation **387**
9.4 CCM-R4F Control Registers **390**

9.1 Main Features

Safety-critical applications require run-time detection of faults in the Central Processing Unit (CPU). For this purpose, the CPU Compare Module for Cortex-R4F (CCM-R4F) compares the core compare bus outputs of two Cortex-R4F CPUs running in a 1oo1D (one-out-of-one, with diagnostics) lockstep configuration. Any difference in the core compare bus outputs of the CPUs is flagged as an error. For diagnostic purposes, the CCM-R4F also incorporates a self-test capability to allow for boot time checking of hardware faults within the CCM-R4F itself.

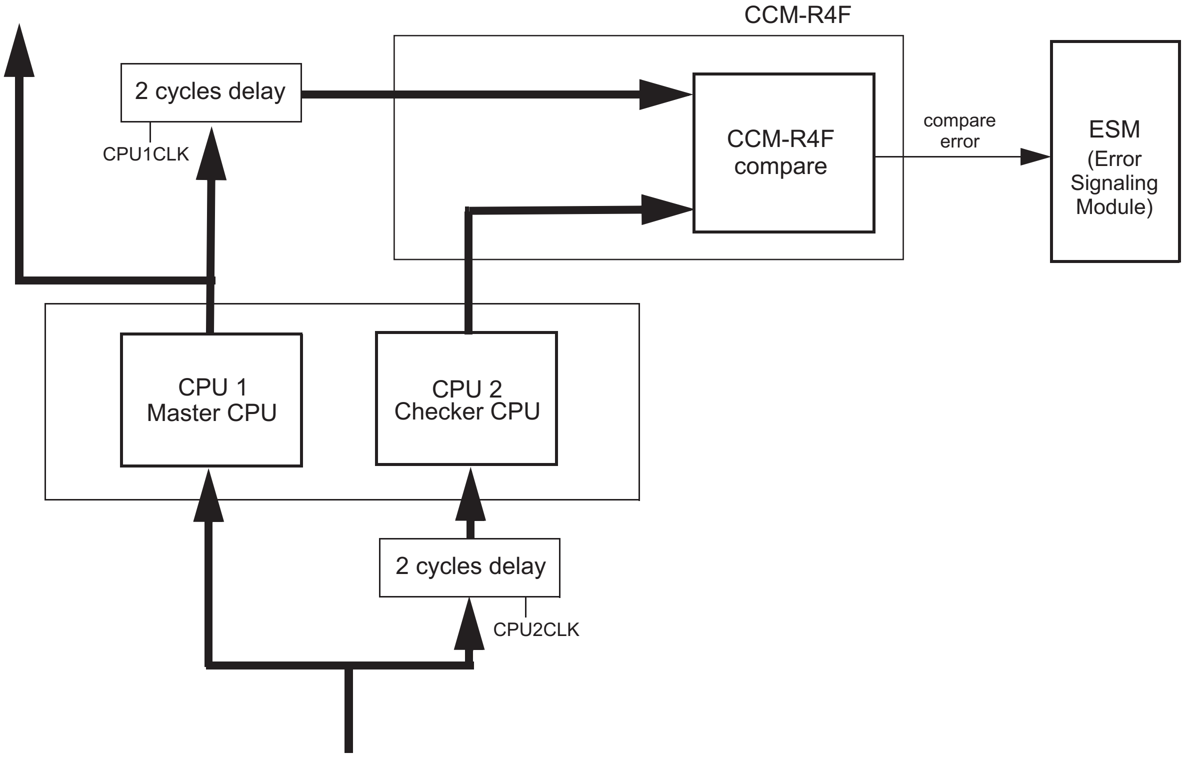
The main features of the CCM-R4F are:

- run-time detection of faults
- self-test capability
- error forcing capability

9.2 Block Diagram

Figure 9-1 shows the interconnection diagram of the CCM-R4F with the two Cortex-R4F CPUs. The core compare bus outputs of the CPUs are compared in the CCM-R4F. To avoid common mode impacts, the signals of the CPUs to be compared are temporally diverse. The output signals of the master CPU are delayed 2 cycles while the input signals of checker CPU are delayed 2 cycles. The CCM-R4F is constantly comparing about 900 signals from each of the two CPUs. These signals include the address, data and control signals from the flash and RAM TCMs and from the AXI peripheral bus. An internal register or ALU error will be flagged as a mis-compare when the faulty value is stored, used as an index, or causes a change in program execution.

Figure 9-1. Block Diagram



9.3 Module Operation

The CCM-R4F compares the core compare bus outputs of the master and checker Cortex-R4F CPUs on the microcontroller and signals an error on any mismatch. This comparison is started 6 CPU clock cycles after the CPU comes out of reset to ensure that CPU output signals have propagated to a known value after reset. Once comparison is started, the CCM module continues to monitor the outputs of two CPUs without any software intervention. Upon an error software needs to handle it.

The CCM-R4F can run in one of the following four operating modes:

1. 1oo1D lock step
2. self-test
3. error forcing
4. self-test error forcing

The operating mode can be selected by writing a dedicated key to the key register (MKEY).

9.3.1 1oo1D Lock Step Mode

This is the default mode on start-up.

In lock step mode, the compare bus output signals of both CPUs are compared. A difference in the CPU compare bus outputs is indicated by signaling an error to the ESM which sets the error flag “CCM-R4F - compare”.

NOTE: The CPU compare error asserts “CCM-R4F self-test error” flag as well. By doing this, the CPU compare error has two paths (“CCM-R4F - compare” and “CCM-R4F self-test error” flag) to the ESM, so that even if one of the paths fails, the error is still propagated to the ESM.

Not all internal registers of the Cortex-R4F CPU have fixed values upon reset. To avoid an erroneous CCMR4F compare error, the application software needs to ensure that the CPU registers of both CPUs are initialized with the same values before the registers are used, including function calls where the register values are pushed onto the stack.

9.3.2 Self-Test Mode

In self-test mode, the CCM-R4F checks itself for faults. During self-test, the compare error module output signal is deactivated. Any fault detected inside the CCM-R4F will be flagged by ESM error “CCM-R4F - self-test”.

In self-test mode, the CCM-R4F automatically generates test patterns to look for any hardware faults. If a fault is detected, then a self-test error flag is set, a self-test error signal is asserted and sent to the ESM, and the self-test is terminated immediately. If no fault is found during self-test, the self-test complete flag is set. In both cases, the CCM-R4F remains in self-test mode after the test has been terminated or completed, and the application needs to switch the CCM-R4F mode by writing another key to the mode key register (MKEY). During the self-test operation, the compare error signal output to the ESM is inactive irrespective of the compare result.

There are two types of patterns generated by CCM-R4F during self-test mode:

i. Compare Match Test
ii. Compare Mismatch Test

CCM-R4F first generates Compare Match Test patterns, followed by Compare Mismatch Test patterns. Each test pattern is applied on both CPU signal inputs of the CCM-R4F’s compare block and clocked for one cycle. The duration of self-test is 3615 CPU clock cycles (GCLK).

NOTE: During self-test, both CPUs can execute normally, but the compare logic will not be checking any CPU signals. Also during self-test, only the compare unit logic is tested and not the memory mapped register controls for the CCM-R4F. The self-test is not interruptible.

9.3.2.1 Compare Match Test

During the Compare Match Test, there are four different test patterns generated to stimulate the CCM-R4F. An identical vector is applied to both input ports at the same time expecting a compare match. These patterns cause the self-test logic to exercise every CPU compare bus output signal in parallel. If the compare unit produces a compare mismatch then the self-test error flag is set, the self-test error signal is generated, and the Compare Match Test is terminated.

The four test patterns used for the Compare Match Test are:

- All 1s on both CPU signal ports
- All 0s on both CPU signal ports
- 0xAs on both CPU signal ports
- 0x5s on both CPU signal ports

These four test patterns will take four clock cycles to complete. illustrates the sequence of Compare Match Test.

Table 9-1. Compare Match Test Sequence

CPU 1 Signal Position									CPU 2 Signal Position									Cycle
n:8	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**n:8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
1s	1	1	1	1	1	1	1	1	1s	1	1	1	1	1	1	1	1	0
0s	0	0	0	0	0	0	0	0	0s	0	0	0	0	0	0	0	0	1
0xA	1	0	1	0	1	0	1	0	0xA	1	0	1	0	1	0	1	0	2
0x5	0	1	0	1	0	1	0	1	0x5	0	1	0	1	0	1	0	1	3

9.3.2.2 Compare Mismatch Test

During the Compare Mismatch Test, the number of test patterns is equal to twice the number of CPU output signals to compare in lock step mode. An all 1s vector is applied to the CCM-R4F’s CPU1 input port and the same pattern is also applied to the CCM-R4F’s CPU2 input port but with one bit flipped starting from signal position 0. The un-equal vector will cause the CCM-R4F to expect a compare mismatch at signal position 0, if the CCM-R4F logic is working correctly. If, however, the CCM-R4F logic reports a compare match, the self-test error flag is set, the self-test error signal is asserted, and the Compare Mismatch Test is terminated.

This Compare Mismatch Test algorithm repeats in a domino fashion with the next signal position flipped while forcing all other signals to logic level 1. This sequence is repeated until every single signal position is verified on both CPU signal ports.

The Compare Mismatch Test is terminated if the CCM-R4F reports a compare match versus the expected compare mismatch. This test ensures that the compare unit is able to detect a mismatch on every CPU signal being compared. Table 9-2 illustrates the sequence of Compare Mismatch Test. There is no error signal is sent to ESM if the expected errors are seen with each pattern.

Table 9-2. Compare Mismatch Test Sequence

CPU 1 Signal Position											CPU 2 Signal Position											Cycle
n	n-1:8		7	6	5	4	3	2	1	0	n	n-1:8		7	6	5	4	3	2	1	0	
1	1	1s	1	1	1	1	1	1	1	1	1	1	1s	1	1	1	1	1	1	1	0	0
1	1	1s	1	1	1	1	1	1	1	1	1	1	1s	1	1	1	1	1	1	0	1	1
1	1	1s	1	1	1	1	1	1	1	1	1	1	1s	1	1	1	1	1	0	1	1	2
1	1	1s	1	1	1	1	1	1	1	1	1	1	1s	1	1	1	1	0	1	1	1	3
::																						
1	1	1s	1	1	1	1	1	1	1	1	1	0	1s	1	1	1	1	1	1	1	1	n-1
1	1	1s	1	1	1	1	1	1	1	1	0	1	1s	1	1	1	1	1	1	1	1	n
1	1	1s	1	1	1	1	1	1	1	0	1	1	1s	1	1	1	1	1	1	1	1	n+1
1	1	1s	1	1	1	1	1	1	0	1	1	1	1s	1	1	1	1	1	1	1	1	n+2
1	1	1s	1	1	1	1	1	0	1	1	1	1	1s	1	1	1	1	1	1	1	1	n+3
1	1	1s	1	1	1	1	0	1	1	1	1	1	1s	1	1	1	1	1	1	1	1	n+4
::																						
1	0	1s	1	1	1	1	1	1	1	1	1	1	1s	1	1	1	1	1	1	1	1	2n-1
0	1	1s	1	1	1	1	1	1	1	1	1	1	1s	1	1	1	1	1	1	1	1	2n

9.3.3 Error Forcing Mode

In error forcing mode, a test pattern is applied to the CPU related inputs of the CCM-R4F compare logic to force an error in the compare error output signal of the compare unit. The ESM error flag “CCM-R4F - compare” is expected after the error forcing mode completes. As a side effect, the “CCM-R4F self-test error” flag is also asserted whenever the CPU compare error is asserted.

Error forcing mode is similar to the Compare Mismatch Test operation of self-test mode in which an unequal vector is applied to the CCM-R4F CPU signal ports. The error forcing mode forces the compare mismatch to actually assert the compare error output signal. This ensures that faults in the path between CCM-R4F and ESM is detected.

Only one hardcoded test pattern is applied into CCM-R4F during error forcing mode. A repeated 0x5 pattern is applied to CPU1 signal port of CCM-R4F input while a repeated 0xA pattern is applied to the CPU2 signal port of CCM-R4F input. The error forcing mode takes one cycle to complete. Hence, the failing signature is presented for one clock cycle. After that, the mode is automatically switched to lock step mode. The key register (MKEY) will indicate the lock step key mode once it is switched to lock step mode. During the one cycle required by the error forcing test, the CPU output signals are not compared. User should expect the ESM to trigger a response (report the CCM-R4F fail). If no error is detected by ESM, then a hardware fault is present.

9.3.4 Self-Test Error Forcing Mode

In self-test error forcing mode, an error is forced at the self-test error signal. The compare unit is still running in lockstep mode and the key is switched to lockstep after one clock cycle. The ESM error flag “CCM-R4F - self-test” is expected after the self-test error forcing mode completes. Once the expected errors are seen, the application can clean the error through ESM module.

9.3.5 Operation During CPU Debug Mode

Certain debug operations place the CPU in a halting debug state where the code execution is halted. Because halting debug events are asynchronous, there is a possibility for the debug requests to cause loss of lockstep. CCM-R4F will disable upon detection of halting debug requests. Core compare error will not be generated and flags will not update. A CPU reset is needed to ensure the CPUs are again in lockstep and will also re-enable the CCM-R4F.

9.4 CCM-R4F Control Registers

Table 9-3 lists the CCM-R4F registers. Each register begins on a 32-bit word boundary. The registers support 32-bit, 16-bit and 8-bit accesses. The base address for the control registers is FFFF F600h.

Table 9-3. CCM-R4F Control Registers

Offset	Acronym	Register Description	Section
00h	CCMSR	CCM-R4F Status Register	Section 9.4.1
04h	CCMKEYR	CCM-R4F Key Register	Section 9.4.2

9.4.1 CCM-R4F Status Register (CCMSR)

Figure 9-2. CCM-R4F Status Register (CCMSR) (Address = FFFF F600h)

31	17	16
Reserved		CPME
R-0		R/WPC-0

15	9	8	7	2	1	0
Reserved		STC	Reserved		STET	STE
R-0		R-0	R-0		R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; C = Clear; WP = Write in Privileged mode only; -*n* = value after reset

Table 9-4. CCM-R4F Status Register (CCMSR) Field Descriptions [1]

Bit	Field	Value	Description
31-17	Reserved	0	Reads return zeros and writes have no effect.
16	CMPE	0	Compare Error **Read in User and Privileged mode. Write in Privileged mode only.** Read: CPU signals are identical. Write: Leaves the bit unchanged.
		1	Read: CPU signal compare mismatch. Write: Clears the bit.
15-9	Reserved	0	Reads return zeros and writes have no effect.
8	STC	0	Self-test Complete **Note:** This bit is always 0 when not in self-test mode. Once set, switching from self-test mode to other modes will clear this bit. **Read/Write in User and Privileged mode.** Read: Self-test on-going if self-test mode is entered. Write: Writes have no effect.
		1	Read: Self-test is complete. Write: Writes have no effect.
7-2	Reserved	0	Reads return zeros and writes have no effect.
1	STET	0	Self-test Error Type **Read/Write in User and Privileged mode.** Read: Self-test failed during Compare Match Test if STE = 1. Write: Writes have no effect.
		1	Read: Self-test failed during Compare Mismatch Test if STE = 1. Write: Writes have no effect.
0	STE	0	Self-test Error **Note:** This bit gets updated when the self-test is complete or an error is detected. **Read/Write in User and Privileged mode.** Read: Self-test passed. Write: Writes have no effect.
		1	Read: Self-test failed. Write: Writes have no effect.

[1] The contents of this register should be interpreted in context of what test was selected. That is what mode is CCM operating in.

9.4.2 CCM-R4F Key Register (CCMKEYR)

Figure 9-3. CCM-R4F Key Register (CCMKEYR) (Address = FFFF F604h)

31	16
Reserved	
R-0	

15	4	3	0
Reserved		MKEY	
R-0		R/WP-0	

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

Table 9-5. CCM-R4F Key Register (CCMKEYR) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return to zeros and writes have no effect.
3-0	MKEY		Mode Key **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: Returns current value of the MKEY. Write: Lockstep mode.
		6h	Read: Returns current value of the MKEY. Write: Self-test mode.
		9h	Read: Returns current value of the MKEY. Write: Error Forcing mode.
		Fh	Read: Returns current value of the MKEY. Write: Self-test Error Forcing mode.
		Other values	**Note:** It is recommended to not write any other key combinations. Invalid keys will result in switching operation to lockstep mode.

Chapter 10

SPNU514C–March 2018

Oscillator and PLL

This chapter describes the oscillator and PLL clock source paths for the device.

Topic **Page**

10.1 Introduction **394**
10.2 Quick Start **395**
10.3 Oscillator **396**
10.4 Low-Power Oscillator and Clock Detect (LPOCLKDET) **398**
10.5 PLL **401**
10.6 PLL Control Registers **411**
10.7 Phase-Locked Loop Theory of Operation **415**
10.8 Programming Example **417**

10.1 Introduction

This chapter provides an overview of the oscillator and PLL clock source paths for the device.

The oscillator macro will pass a signal driven into the OSCIN pin to clock source 0 that is the device default clock source on reset. When a crystal or resonator with appropriate load circuitry is connected to OSCIN and OSCOUT, the oscillator macro drives the crystal/resonator to generate the input waveform. In addition to being directly usable as clock source 0, the oscillator clock is the input to the PLL.

The oscillator frequency is continuously monitored by a dedicated clock detect circuit. If the frequency falls out of a fixed range, the clock detect switches the clock from the oscillator to an internally generated, free-running frequency (generated by the low-power oscillator (LPO)).

The phase lock loop (PLL), a circuit in the microcontroller, is used to multiply the input frequency to some higher (device operation) frequency. This frequency synthesis is useful for generating higher frequencies than can be conveniently achieved with an external crystal or resonator. Additionally, the PLL allows the flexibility to be able to synthesize one of multiple frequency options from a given crystal or resonator.

Frequency modulation can be superimposed on the synthesized frequency. The modulation provides a means to reduce the impact of electromagnetic radiation from the device; this reduction in measured radiation can be useful in sensitive applications.

10.1.1 Features

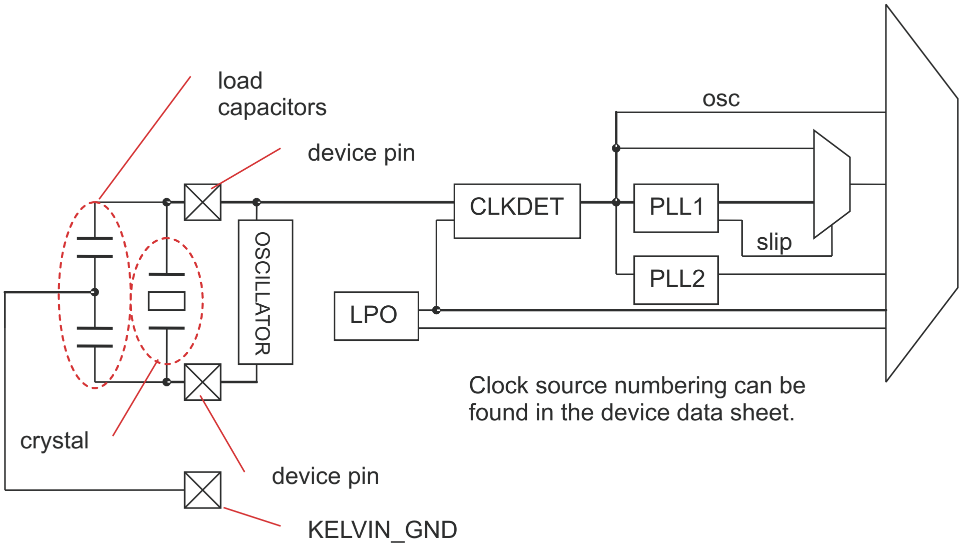
The main features of the source clock path are:

- The oscillator may drive a crystal/resonator or be driven from an external source
- The clock detect provides continuous monitoring of the oscillator frequency and provides an automatic switch over to a free-running clock in case of oscillator failure.
- The FM-PLL module can be operated in either modulation or non-modulation mode.
- The phase-frequency detector assures lock to the fundamental reference frequency.
- $$f_{PLL} = \frac{f_{OSCIN}}{NR} \times \frac{NF}{OD \times R} \quad (1)$$
  - Configurable prescale divider (NR) for the input clock
  - Configurable multiplier (NF)
  - Configurable postscale dividers (OD, R)
- The PLL may be used with modulation enabled.
  - Configurable modulation frequency (NS)
  - Configurable modulation depth (NV)
- The slip control circuitry provides flexible response to a PLL failure (slip) including reset or automatic switch over to oscillator.

10.2 Quick Start

The purpose of this section is to provide an overview of how to configure the oscillator and PLL clock paths on power-up. More detailed descriptions are presented in later sections. Figure 10-1 shows the oscillator and PLL clock paths.

Figure 10-1. Clock Path From Oscillator Through PLL To Device



Clock source numbering can be found in the device data sheet.

While power-on reset is asserted (low), the oscillator and low-power oscillator (LPO) are enabled and start-up by default. After power-on reset is released to a high level, the clock detect circuit (CLKDET) begins to monitor the oscillator. If the oscillator is within a valid range, the oscillator becomes the default clock for the device as it exits reset; if the oscillator is not within a valid range, the clock detect selects the high-frequency low-power oscillator as the default clock for the device.

The low-power oscillator has a wide frequency range which also creates a large valid window for the clock detect; in order to refine the clock detect window, the low-power oscillator can be trimmed. The initial trim value is stored in one-time programmable section of the flash memory, address 0xF008_01B4. Bits 31:16 of this word contain a 16 bit value that may be programmed into LPOMONCTL(15:0) in order to initialize the trim for both HF LPO and LF LPO. Software should read the initial trim values from flash and write them to the control register.

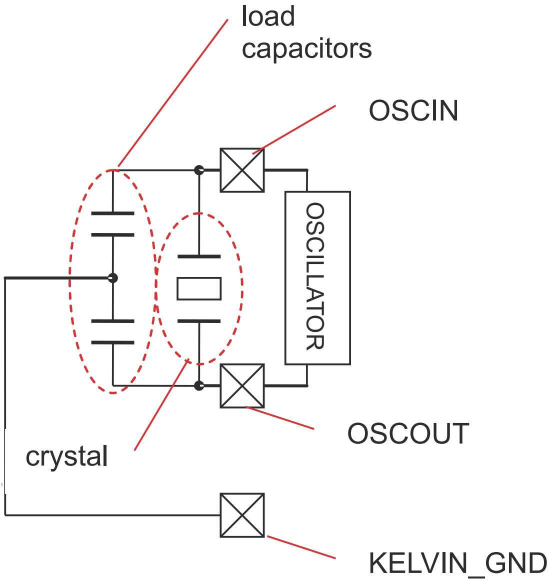
The PLL is disabled by default on power-up. The PLL control registers (PLLCTL1 and PLLCTL2) must be configured to set the desired output frequency. Then, the system PLL may be enabled (CSDISCLR.1). Similarly, the second PLL must be configured in PLLCTL3 and enabled (CSDISCLR.6). Each PLL has a valid bit that indicates the PLL is locked (CLKSRnV bit in the Clock Source Valid Status Register (CSVSTAT) of the System and Peripheral Control Registers).

Prior to selecting the PLL clock as the source for a clock domain (for example, GCLK, HCLK, VCLKA1), the domain and modules on the domain must be configured to accept the new frequency. An example of a module that should be configured prior to selecting the PLL as clock source for GCLK and HCLK is the memory wrapper to insure that access times are maintained correctly.

10.3 Oscillator

The clock generation path through the PLL begins with the oscillator. The oscillator consists of three separate pads -- OSCIN, OSCOUT, and Kelvin_GND (see Figure 10-2).

Figure 10-2. Clock Generation Path



The oscillator is responsible for two independent functions:

1. The oscillator is responsible for generating positive feedback in the external crystal/resonator with appropriate load and tank circuitry. At start-up, the oscillator amplifies random noise. The external circuitry acts like a band-pass and selects the crystal/resonator frequency to provide as positive feedback into the amplifier. The positive feedback increases the amplitude of the output waveform into the crystal/resonator (and the load circuitry), and the voltage waveform shows an envelope of increasing amplitude. The oscillator can drive a crystal frequency that is within the data sheet range $t_{c(OSC)}$.

   Looking at the input waveform into OSCIN, the voltage waveform is an AC-coupled, filtered version of the OSCOUT waveform. The band-pass functionality of the crystal/resonator removes distortion from the OSCOUT waveform, leaving a sinusoidal input waveform.

   NOTE: Vendor Validation of Resonators/Crystals

   The crystal is a very tight bandpass filter while a resonator is a somewhat wider bandpass. The load circuitry pulls the center frequency of the bandpass.

   Texas Instruments strongly encourages each customer to submit samples of the device to the resonator/crystal vendor for validation. The vendor is equipped to determine what load capacitances will best tune their resonator/crystal to the microcontroller device for optimum start-up and operation over temperature and voltage extremes. The vendor also factors in margins for variations in the microcontroller process.

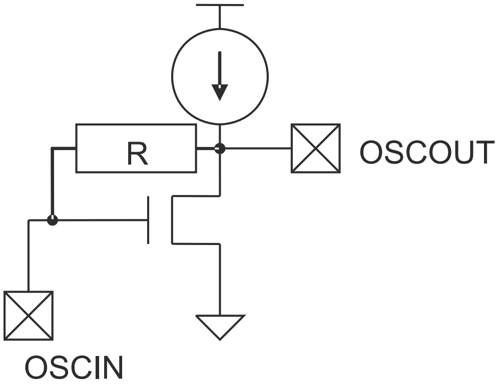
2. The oscillator is also responsible for squaring-up the input waveform. This squaring-up converts the sinusoid into a square wave at the core logic levels. The input path limits the input frequency range as a low-pass filter with a cutoff frequency.

   The oscillator has a frequency range that is determined by the driving capability of external crystals/resonators (feedback path). If a clock is driven directly into the oscillator, then the feedback path is not relevant and the frequency range is determined solely by the forward path (which typically allows a higher frequency); the device can support inputs within the data sheet range $t_{c(OSC\_Sqr)}$.

10.3.1 Oscillator Implementation

The oscillator operates at 3.3V and uses a constant current source to drive current onto the OSCOUT node. An internal transistor shunts the current (and current from the external circuitry) to GND. This current steering drives the voltage waveform on OSCOUT.

Figure 10-3. Oscillator Implementation



10.3.2 Oscillator Enable

The oscillator is enabled asynchronously when nPORRST is low.

The oscillator is enabled by clearing bit 0 in the Clock Source Disable Register (CSDIS) or setting bit 0 in the Clock Source Disable Clear Register (CSDISCLR) of the System and Peripheral Control Registers. The bit sends a start signal to the oscillator. Bit 0 of CSDIS is cleared to 0 by default on a system or power-on reset so that the oscillator starts-up by default. After the oscillator swings at a high-enough amplitude to pass an input clock into the core domain and nPORRST is released, 1024 oscillator periods are counted before setting the CLKSR0V bit in the Clock Source Valid Status Register (CSVSTAT) of the System and Peripheral Control Registers. The oscillator generates clock source 0 in the global clock module (GCM).

10.3.3 Oscillator Disable

The clock sources (for example, OSC, PLL) are disabled by setting the appropriate bit in the Clock Source Disable Register (CSDIS) or setting the appropriate bit in the Clock Source Disable Set Register (CSDISSET) of the System and Peripheral Control Registers. These bits *allow* the clock source to disable but do not force the behavior until the clock is no longer used as the source for a clock domain (for example, GCLK, VCLK, VCLK2, RTICLK). The CLKSR0V bit in the Clock Source Valid Status Register (CSVSTAT), of the System and Peripheral Control Registers, is cleared after clock disable is asserted (which occurs after all clock domains are stopped).

The oscillator disable signal places the oscillator into a low-power state, disconnects the feedback (bias) resistor between OSCIN and OSCOUT, and OSCIN is grounded.

10.4 Low-Power Oscillator and Clock Detect (LPOCLKDET)

The low-power oscillator (LPO) is comprised of two oscillators -- HF LPO and LF LPO -- in a single macro. The low-power oscillator and clock detect (LPOCLKDET) uses a relaxation oscillator to generate an internal clock whose frequency is NOT tightly controlled. This frequency is used to monitor the oscillator input frequency and is also available as an independent clock source in the GCM.

The LPO produces two frequencies:

- High-frequency low-power oscillator (HF LPO) with a nominal frequency of 9.6MHz and a range from 5.5MHz to 19.5MHz; the HF LPO generates clock source 5 in the GCM.
- Low-frequency low-power oscillator (LF LPO) with a nominal frequency of 85kHz; the LF LPO generates clock source 4 in the GCM.

A single current source drives current onto a capacitor; when the voltage on the capacitor exceeds some threshold, the clock toggles. The LPO uses a single current source and the two different comparators to generate the HF LPO and LF LPO frequencies. The LPO is controlled by 4 different bit fields -- CSDIS.(5:4), HFTRIM(4:0), LFTRIM(4:0), and BIASEN.

- CSDIS.5 enables/disables the comparator that generates HF LPO.
- CSDIS.4 enables/disables the comparator that generates LF LPO.
- The HF TRIM and LF TRIM bit fields vary the current into the comparator to independently trim the HF LPO and LF LPO frequencies.
- BIAS ENABLE (LPOMONCTL.24) enables/disables the current source that drives the LPO.

10.4.1 Clock Detect

The LPO HF clock frequency is typically near 9.6MHz but ranges from 5.5MHz to 19.5MHz. The clock detect establishes a window for the oscillator by:

OSCIN > HF LPO$_{min}$ / 4	OSCIN / 4 < HF LPO$_{max}$
OSCIN > 5.5[MHz] / 4 = 1.375[MHz]	OSCIN < 4 × 19.5 = 78[MHz]

The clock detect circuit works by checking for a rising edge on one clock (oscillator or HF LPO) between rising edges of the other clock. The result is that in addition to flagging incorrect, repeating frequencies, the circuit also fails due to transient conditions.

The low end of the clock detect window ignores a transient low phase of at least 12 HF LPO cycles.

NOTE: Clock Detection of Oscillator MUST be Disabled Before Disabling HF LPO

The HF LPO frequency is the comparison frequency for the oscillator. The clock detection must be disabled prior to disabling the HF LPO frequency.

If the clock detection is NOT disabled prior to disabling the HF LPO, the clock detect circuitry will fail the oscillator as too fast (compared to the non-existent HF LPO). The clock detect circuitry will switch to the non-existent clock, leaving the device without a valid clock.

10.4.2 Behavior on Oscillator Failure

If the oscillator frequency fails, the clock detects supplies:

- the HF LPO clock to GCM clock source 0 instead of the oscillator
- the HF LPO clock to GCM clock source 1 instead of the PLL

The HF LPO signal will be available as three different clock sources:

- GCM clock source 0 (replacing the oscillator)
- GCM clock source 1 (replacing the PLL)
- GCM clock source 5 as HF LPO

The automatic switch-over from oscillator to HF LPO allows the application to execute at a reduced frequency and respond to a problem with the external crystal/resonator. During and after an oscillator failure, the oscillator CLKSRnV bit in the Clock Source Valid Status Register (CSVSTAT), of the System and Peripheral Control Registers, is set along with the OSCFAIL flag in the Global Status Register (GLBSTAT), of the System and Peripheral Control Registers.

It is useful to explicitly change the GHVSRC register, defining the current clock source for GCLK/HCLK/VCLK domains, to the HF LPO after an oscillator failure.

When reset on oscillator failure is set, PLLCTL1.23 (ROF), the device responds to an oscillator failure by generating a device reset.

10.4.3 Recovery from Oscillator Failure

If the oscillator fails, the clock detect switches the HF LPO frequency onto the oscillator source into the GCM. The OSCFAIL flag in the Global Status Register (GLBSTAT), of the System and Peripheral Control Registers, is also set.

The oscillator may be re-enabled (though if the failure was caused by a hard-fault, the re-enable will fail) through the following procedure:

1. Switch all clock domains from the oscillator to the HF LPO (for example, GHVSRC uses HF LPO, VCLKAn uses HF LPO or VCLK, and so on).
2. If the PLL is used, disable the PLL by setting the appropriate bit in the Clock Source Disable Set Register (CSDISSET) of the System and Peripheral Control Registers.
3. Disable the oscillator by setting the appropriate bit in the Clock Source Disable Set Register (CSDISSET). This action resets the clock detect and allows the oscillator to propagate through GCM clock source 0.
4. Re-enable the oscillator by setting the appropriate bit in the Clock Source Disable Clear Register (CSDISCLR) of the System and Peripheral Control Registers.
5. Clear the OSCFAIL flag in the Global Status Register (GLBSTAT) by writing a 1 to the bit. The PLL slip bits may also be set on an oscillator failure. These can also be cleared.
6. Switch the clock domains back to the oscillator.
7. Re-enable the PLL by setting the appropriate bit in the Clock Source Disable Clear Register (CSDISCLR).

NOTE: Clock Re-Enable Procedure Will Fail If Caused by a Hard Failure

Although it is possible to re-enable the oscillator after a failure, if the oscillator failure was caused by a hard fault (for example, disconnected crystal/resonator terminal), the re-enable process will fail.

10.4.4 LPOCLKDET Enable

The LPO is enabled by default while nPORRST is low. During this time, the current source initializes, holding the relaxation oscillator in reset until initialized. After the current source releases the HF LPO and the LF LPO, these clock frequencies slew to their final frequencies; the final frequency may be achieved while nPORRST is active or after its release. After, nPORRST is released, the HF LPO Valid signal is set 32 HF LPO clock cycles later.

The clock detect is enabled once the oscillator and HF LPO are valid. Because an oscillator failure could occur from reset, the clock detect logic must provide an override path. If the HF LPO is valid and the oscillator is not valid, the clock detect circuitry will become active (overriding the oscillator invalid signal) after 16K LF LPO cycles (about 200ms).

10.4.5 LPOCLKDET Disable

10.4.5.1 Disable Clock Detect

It is possible to disable the clock detect circuitry. For protection, this clock detect disable employs a 2-bit key:

- RANGE DET ENA SSET (CLKTEST.24) must be set to 1
- RANGE DET CTRL (CLKTEST.25) must be cleared to 0

In this case, the LPO HF and LF clocks are still active but the clock detect circuitry is disabled. The clock detect unconditionally switches GCM_CLK_SRC(0) back to the oscillator so care should be taken to insure that the oscillator is good before disabling the clock detect circuitry.

10.4.5.2 Disable LPO HF and LF Clocks

The LPO may be disabled by holding the relaxation oscillator clocks (HF and LF) in reset. The clock detect must be disabled, and any clock domains using either HF or LF clocks must be switched to a different clock source. The LPO HF clock is reset by setting CSDIS.5; CSDISSET.5 is an easy way to set specific bits without disturbing the rest of the register. The HF LPO clock disables several HF LPO cycles after CSDIS is set.

Similarly, the LPO LF clock is reset by setting CSDIS.4, and in a similar way CSDISSET.4 can set the specific CSDIS register bit without using a read-modify-write construction. The LF LPO disables several LF LPO cycles after CSDIS is set.

Restarting the LPO clocks from this condition is fast and is known as a warm re-start. The CSDISCLR register allows the user to clear CSDIS bits without using a read-modify-write code-construct.

10.4.5.3 Disable LPO Current Bias

The LPO current source may be disabled after the clock detect is disabled and HF and LF clock sources are disabled. Turning off this current source places the LPOCLKDET into its lowest power configuration. The bias may be disabled by clearing the BIAS ENABLE bit (LPOMONCTL.24).

Restarting the LPO when the bias current has been disabled requires the current source to initialize first and is, therefore slower than a warm re-start; re-enabling the LPO from this condition is known as a warm re-start (similar to what happens during nPORRST active).

10.4.6 Trimming the HF LPO Oscillator

The HF LPO range varies considerably around 9.6MHz from device to device. In order to provide tighter monitoring of the crystal/resonator, it is useful to trim the oscillator. During device test, a trim value is written into the one-time programmable section of the flash memory (OTP), address 0xF008_01B4. Bits 31:16 of this OTP word contain a 16 bit value that may be programmed into LPOMONCTL(15:0) in order to initialize the trim for both HF LPO and LF LPO.

When trimming the HF LPO, it is recommended to step the trim value so as not to make a large change to any TRIM setting.

After the initial trim, further trimming may be done in LPOMONCTL, using the dual clock compare module (see Chapter 11) in order to determine the resultant frequency. This module allows for comparison of two clock frequencies. Once the HF LPO is determined to be in-range with the initial HFTRIM setting from the OTP, the crystal oscillator may be used as a reference against which the HF LPO and LF LPO may be further adjusted.

10.5 PLL

The following bit fields from PLLCTL1 and PLLCTL2 configure the PLL:

- REFCLKDIV[5:0]
- PLLMUL[15:0]
- ODPLL[2:0]
- PLLDIV[4:0]
- SPR_AMOUNT[8:0]
- SPREADINGRATE[8:0]
- FMENA

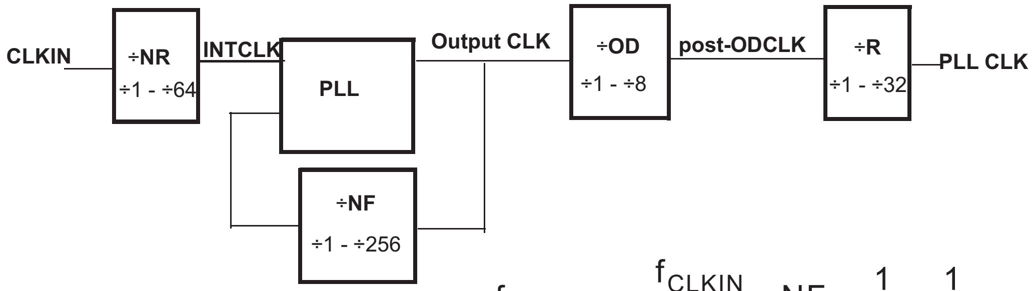
The PLL is responsible for synthesizing an output frequency from the input clock (from the oscillator); Figure 10-4 shows a simple block diagram of the PLL. The FM-PLL divides the reference input for a lower frequency input into the PLL ($f_{INTCLK} = f_{CLKIN}/NR$). The PLL multiplies this internal frequency by NF to get the VCO output clock frequency ($f_{Output\ CLK} = f_{INTCLK} \times NF$). The PLL output is subsequently divided by two prescale values (OD and R). The value of OD is an integer from 1 to 8 and R is an integer from 1 to 32. This output clock, PLL CLK, sources GCM clock source 1. Valid frequencies are shown in Table 10-1 while Table 10-2 shows how that encoding is generated from the PLL bit fields.

[$f_{(post\_ODCLK)}$ and $f_{(GCLK)}$ are data sheet parameters.]

Table 10-1. Valid Frequency Ranges for PLL

	Frequency Limit
f_{CLKIN}	$f_{(OSC\_Sqr)}$
f_{INTCLK}	1MHz - $f_{(OSC\_Sqr)}$
$f_{Output\ CLK}$	150MHz - 550MHz
$f_{post\text{-}ODCLK}$	$f_{(post\_ODCLK)}$
$f_{PLL\ CLK}$	$f_{(GCLK)}$

Figure 10-4. Operation of the FM-PLL Module



$$f_{PLLCLK} = \frac{f_{CLKIN}}{NR} \times NF \times \frac{1}{OD} \times \frac{1}{R}$$

Table 10-2. PLL Value Encoding

	PLL
NR	$NR = REFCLKDIV[5...0]+1$ (2)
NF	Non-modulated: $NF = \frac{(PLLMUL[15...0] + 256)}{256}$ (3) Modulated: $NF = \frac{(PLLMUL[15...0] + MULMOD[8...0] + 256)}{256}$ (4)
NV	$NV = \frac{(SPR\_AMOUNT[8...0]+1)}{2048}$ (5)
NS	$NS = SPRRATE[8...0]+1$ (6)
OD	$OD = ODPLL[2...0]+1$ (7)

NOTE: ODPLL change should occur prior to enabling asynchronous clock domains

Since changing the ODPLL bit-field causes the PLL CLK to be gated, these changes to ODPLL should be completed before configuring a clock domain for an asynchronous clock source. Some clock domains (for example, RTICLK, VCLK2) require a frequency relationship to the VCLK.

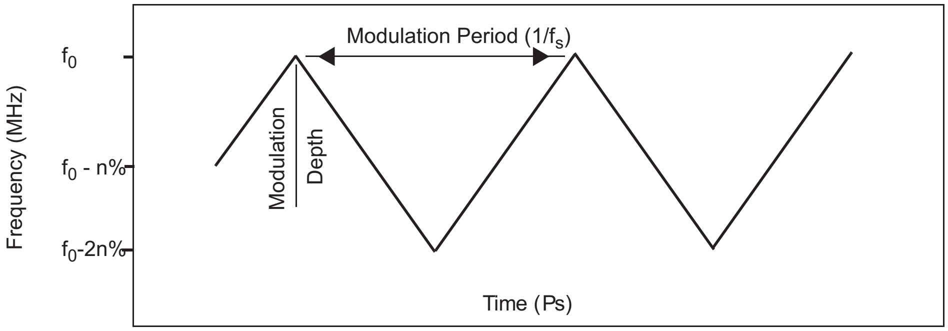
$$f_{VCLK} \geq 3 \times \frac{f_{RTISRC}}{RTIDIV}$$

If the PLL is clocking VCLK and it is stopped for some cycles, then the frequency relationship is temporarily violated.

Many asynchronous domains require frequency relationships between VCLK and the asynchronous domain. Therefore, if the PLL clock is the source for GCLK, HCLK, and VCLK, then the gating produces a short-term change in the PLL clock frequency (and hence also the VCLK frequency). As such, this frequency change could violate the requirements for an asynchronous clock domain.

10.5.1 Modulation

Optionally, the frequency can be modulated, that is, a controlled jitter is introduced onto the baseline frequency of the PLL. This modulation mechanism is not shown in Figure 10-4. When the PLL is used in the modulating mode, the programmable modulation block varies the PLL frequency from the baseline frequency ($f_{baseline}$ = (f_{CLKIN}/NR) × NF/(OD × R)) to $f_{baseline}$ × (1 - 2 × Depth) in a period defined by $1/f_s$; the modulation waveform is triangular and should be enabled after lock.



The modulation is digital and the spreading profile is triangular, down-spread which implies:

- the modulation waveform is composed of a series of frequency steps.
- the modulation frequency and modulation depth are both well controlled due to their digital character.
- the average frequency during modulation is lower than the average frequency prior to enabling modulation. The depth of modulation, however, sets the new average frequency.
- the modulation frequency must be selected slower than the loop bandwidth. From a practical perspective, NS should be near 20.

The modulation fields have a simple geometric meaning:

- the modulation step size is:

$$\frac{NV}{NF} \times f_{OutputCLK} \tag{8}$$

- the number of steps per modulation period is 2 × NS
- the modulation depth is given by:

$$\Delta f = \frac{NS}{2} \times \frac{NV}{NF} \times f_{OutputCLK}$$

$$Depth[\%] = \frac{NS}{2} \times \frac{NV}{NF} \tag{9}$$

- the modulation frequency is:

$$T_{\mathrm{mod}} = \frac{f_{OSC}}{2 \times NR \times NS} \tag{10}$$

- MULMOD minimizes frequency offset when programmed as:

$$\frac{(SPR\_AMOUNT[8...0]+1)(SPRRATE[8...0]+1)}{16} \tag{11}$$

NOTE: Modulation should be enabled after Lock

Enable modulation after the lock is completed.

10.5.2 PLL Output Control

The outputs from the PLL are the output clock, slip signals, and VALID.

- RFSLIP -- the RFSLIP signal indicates that the Output CLK is running *too fast* relative to INTCLK and sets a RFSLIP status flag in the Global Status Register (GLBSTAT), of the System and Peripheral Control Registers, if the slip signal is active during normal PLL operation; the RFSLIP flag is masked off while the PLL is not active and during the PLL's lock period.
- FBSLIP -- the FBSLIP signal indicates that the Output CLK is running *too slow* relative to INTCLK and sets a FBSLIP status flag in the Global Status Register (GLBSTAT), of the System and Peripheral Control Registers, if the slip signal is active during normal PLL operation; the FBSLIP flag is masked off while the PLL is not active and during the PLL's lock period.
- PLL Slip -- Logical-OR of the two PLL slip signals. Typically this signal is used to generate a consolidated slip signal to the device (for example, error logic or exception generation). Also used to gate VALID.

NOTE: Clearing Slip Bits

In order to clear any of the slip bits, it is necessary to disable the PLL first.

- VALID -- is driven based upon whether the output clock, PLL CLK, is gated or not. However, the VALID signal is dependent upon the PLL Slip signals so that VALID cannot be set if either slip signal is active.
- PLL Clock -- The PLL output clock runs at the programmed frequency. When enabled, it takes some time to acquire the programmed frequency (see Section 10.5.2.1). Similarly, the disable has some timing/constraints (see Section 10.5.2.2).

10.5.2.1 PLL Enable

After setting the PLL control registers, the clock source is enabled by clearing the appropriate bit in the Clock Source Disable Register (CSDIS) or setting the appropriate bit in the Clock Source Disable Clear Register (CSDISCLR) of the System and Peripheral Control Registers. The bit sends a signal to the PLL that starts the process of enabling the PLL.

1. The PLL checks to make sure that the oscillator is ON. If not, it turns the oscillator ON.
2. The PLL begins a locking process in which the PLL slews from a starting frequency point to the programmed frequency. During this lock period, the PLL slip signals are typically active, and the PLL masks off the signals during this phase. The lock phase takes the following length of time:

Parameter	Value
Lock	T_{Lock} = (512 × T_{OSCIN}) + (1024 × NR × T_{OSCIN})
Enable clocks after lock	T_{Enable} = 6 × T_{OSCIN}

3. After the lock phase is complete (when lock counters expire), the PLL releases the slip signals to the system.
4. Then, after the slip signals are released and a delay to enable the clocks, the clock is released to the system and the appropriate CLKSRnV bit for the PLL is set in the Clock Source Valid Status Register (CSVSTAT) of the System and Peripheral Control Registers.

10.5.2.2 PLL Disable

The clock sources (for example, OSC, PLL) are disabled by setting the appropriate bit in the Clock Source Disable Register (CSDIS) or setting the appropriate bit in the Clock Source Disable Set Register (CSDISSET) of the System and Peripheral Control Registers. These bits *allow* the clock to disable but do not force the behavior until the clock is no longer used as the source for a clock domain (for example, GCLK, VCLK, VCLK2, RTICLK).

The PLL receives a signal to disable after the clock is no longer used by any clock domain. Within the PLL, the clock is disabled and the appropriate CLKSRnV bit for the PLL in the Clock Source Valid Status Register (CSVSTAT), of the System and Peripheral Control Registers, becomes inactive. Then the PLL is placed into a low-power state after the following length of time: $T_{Enable} = 150 \times T_{OSCIN}$

10.5.2.3 OD-divider Change

The PLL gates the clock if the ODPLL bit-field is changed while the PLL is active. The output clock from the PLL is gated for 3 or 12 OSCIN clock cycles. As the post-ODCLK is gated in the low phase, the output clock to the device -- PLL CLK -- may be gated in a high or low phase though the transition is always glitchless: $T_{ODPLL} = 3 \times T_{OSCIN}$

NOTE: ODPLL change should occur prior to enabling asynchronous clock domains

Since changing the ODPLL bit-field causes the PLL CLK to be gated, these changes to ODPLL should be completed before configuring a clock domain for an asynchronous clock source. Some clock domains (for example, RTICLK, VCLK2) require a frequency relationship to the VCLK.

$$f_{VCLK} \geq 3 \times \frac{f_{RTISRC}}{RTIDIV}$$

If the PLL is clocking VCLK and it is stopped for some cycles, then the frequency relationship is temporarily violated.

Many asynchronous domains require frequency relationships between VCLK and the asynchronous domain. Therefore, if the PLL clock is the source for GCLK, HCLK, and VCLK, then the gating produces a short-term change in the PLL clock frequency (and hence also the VCLK frequency). As such, this frequency change could violate the requirements for an asynchronous clock domain.

10.5.2.4 Changing the PLL Operating Point While the PLL is Active

Once the valid bit (CLKSRnV bit in the Clock Source Valid Status Register (CSVSTAT) of the System and Peripheral Control Registers) is set, software may change values to the PLL. If the change of values results in a small percentage change to the VCO frequency ($\Delta f_{OutputCLK} < 0.1 \times f_{OutputCLK}$), then these changes can be done on-the-fly. In this mode, the values are updated into the PLL synchronously, and the PLL re-locks to the new value without gating the clocks or the slip bits. If the operating point change is too large, then the slip bits will be set.

Conversely, if the changes to the VCO frequency are large, then the PLL should be disabled prior to changing the values. Typically, any change to the REFCLKDIV field or large changes to the PLLMUL field in the PLL Control Register 1 (PLLCTL1) of the System and Peripheral Control Registers requires a complete disable-and-relock strategy.

10.5.2.5 Summary of PLL Timings

In addition to controlling the lock period and disabling the clock during an ODPLL change, the PLL also generates reset delays. When power-on reset is released (nPORRST 0 --> 1), that release is delayed by 1024 OSCIN cycles so that it is released at the same time that the oscillator valid is asserted. The system reset release is delayed by an additional 8 oscillator clock cycles.

Table 10-3. Summary of PLL Timings

Parameter	Value
nPORRST delay	$T_{nPORRST} = 1024 \times T_{OSCIN}$
nRST delay	$T_{nRST} = 1032 \times T_{OSCIN}$
OSC valid	$T_{OSCVALID} = 1024 \times T_{OSCIN}$
Lock	$T_{Lock} = (512 \times T_{OSCIN}) + (1024 \times NR \times T_{OSCIN})$
Enable clocks after lock	$T_{Enable} = 6 \times T_{OSCIN}$
Disable clocks after lock	$T_{Enable} = 150 \times T_{OSCIN}$
Change ODPLL	$T_{ODPLL} = 3 \times T_{OSCIN}$

10.5.3 Behavior on PLL Fail

The PLL allows flexible response to a PLL failure (slip). Like the oscillator, the PLL clock is configured by default to automatically switch-over to the oscillator in case of a PLL slip. (In this case, the oscillator sources GCM clock source 1 as well as GCM clock source 0. Also, if the oscillator fails, LPO HF is sourced to both GCM clock sources 0 and 1.)

The PLL slip outputs indicate that the PLL is running either too fast or too slow. These error output toggle when the PLL is locking and when the PLL is disabling. The PLL blocks these slip outputs during these times, leaving them active only while the PLL is active.

A slip after the PLL has locked and while it is active is an indication of a PLL failure. The PLL provides slip-filtering which enhances the flexibility of the PLL's response to failure. The slip-filtering circuit samples the slip based on HF LPO. The filter defines the number of consecutive HF LPO cycles for which the slip signal must be active before the slip is recognized. This slip is latched in the RFSLIP and FBSLIP status flags in the Global Status Register (GLBSTAT) of the System and Peripheral Control Registers.

The PLL may enable/disable the automatic switch over as well as the error signaling; if the error signaling is enabled, a PLL slip may be configured to generate a reset. The automatic switch-over and suppression of the error signals are controlled by the bypass on slip bit field -- BPOS[1:0] (PLLCTL1.(30:29)). When BPOS[1:0] is disabled (BPOS[1:0] = 10b):

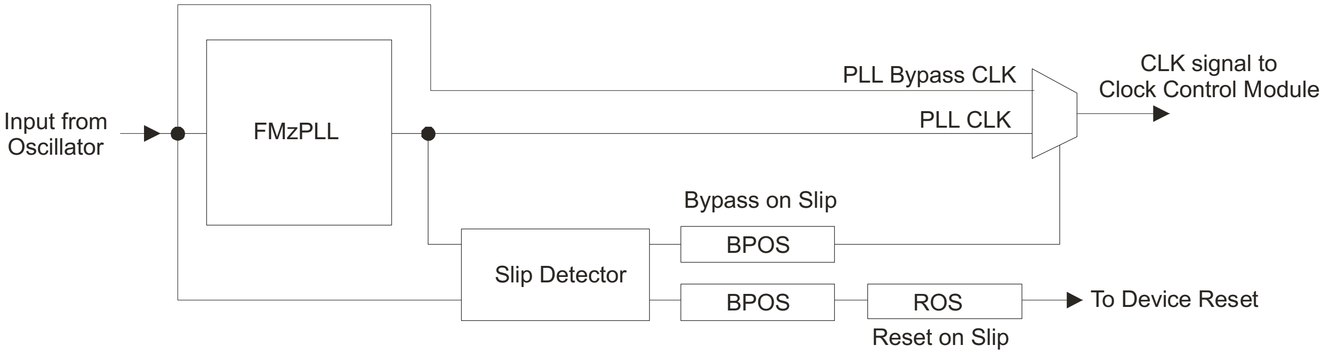
- automatic response to the PLL slip is prevented
- ESM/exception is NOT generated
- reset on slip is not generated regardless of the state of the ROS bit
- status bits are set on a PLL slip independent of BPOS[1:0]

When BPOS[1:0] is enabled (BPOS[1:0] = 00b OR 01b OR 11b):

- PLL slip causes the clock source into GCM clock source 1 to shift from the PLL to the oscillator
- ESM/exception is generated
- reset on slip is generated if ROS is set

The effect of BPOS[1:0] on the system is shown in Figure 10-5.

Figure 10-5. PLL Slip Detection and Reset/Bypass Block Diagram



10.5.4 Recovery from a PLL Failure

If PLL1 fails, the PLL’s slip causes the valid flag to be locked and causes the clock source into GCM clock source 1 to shift from the PLL to the oscillator. The RFSLIP or FBSLIP status flags in the Global Status Register (GLBSTAT) of the System and Peripheral Control Registers are also set. PLL1 may be re-enabled (though if the failure was caused by a hard-fault, the re-enable will fail) through the following procedure:

1. Switch all clock domains from PLL1 to the oscillator (for example, GHVSRC uses oscillator, VCLKAn uses oscillator or VCLK, and so on).
2. Disable PLL1 with CSDISSET. This action disables the PLL and causes the slip signal to no longer be driven. Valid is not released until the slip is cleared.
3. Clear the RFSLIP or FBSLIP status flags in the Global Status Register (GLBSTAT) of the System and Peripheral Control Registers by writing a 1 to the bit. After this step, the valid flag is unlocked and cleared if it was previously set.
4. Re-enable PLL1 with CSDISCLR.
5. Switch the clock domains back to PLL1.

If PLL2 fails, the PLL’s slip causes the valid flag to be locked. There is no autonomous change of clock source for PLL2. Neither the RFSLIP or FBSLIP status flags in the Global Status Register (GLBSTAT) of the System and Peripheral Control Registers are set. PLL2 may be re-enabled in a similar procedure to re-enabling PLL1 (though if the failure was caused by a hard-fault, the re-enable will fail):

1. Switch all clock domains from PLL2 to the oscillator (for example, GHVSRC uses oscillator, VCLKAn uses oscillator or VCLK, and so on).
2. Disable PLL2 with CSDISSET. This action disables the PLL and causes the slip signal to no longer be driven. Valid is not released until the slip is cleared.
3. Reset PLL2 Valid by writing a 1 to both RFSLIP and FBSLIP status flags in the Global Status Register (GLBSTAT) of the System and Peripheral Control Registers (even though they are not set by the slip). After this step, the valid flag is unlocked and cleared if it was previously set.
4. Re-enable PLL2 with CSDISCLR.
5. Switch the clock domains back to PLL2.

10.5.5 PLL Modulation Depth Measurement

The PLL contains a circuit for estimating the depth of the modulation. The circuit counts clock edges over a fixed window of the modulation waveform (SSW_CAPTURE_COUNT in SSWPLL2) and clock edges over the entire waveform (SSW_CLKOUT_COUNT in SSWPLL3). The capture ends after a pre-determined number of clock edges in SSW_CLKOUT_COUNTER as set in TAP_COUNTER_DIS. There are 2 × NR windows per modulation waveform. The procedure for estimating the modulation depth is:

1. While GCLK is sourced by the oscillator and the PLL is enabled with modulation, configure SSWPLL1 as follows:
   a. CAPTURE_WINDOW_INDEX is set equal to NR.
   b. COUNTER_RESET is set.
   c. TAP_COUNTER_DIS is set to disable the measurement after SSW_CLKOUT_COUNT captures this number of clocks. The measurement is disabled after the set tap is set AND the modulation cycle ends.
   d. Ensure that EXT_COUNTER_EN is cleared.
2. Ensure that both SSW_CAPTURE_COUNT and SSW_CLKOUT_COUNT are cleared (by the COUNTER_RESET).
3. Set COUNTER_EN and clear COUNTER_RESET. This step releases the reset and enables the counter to begin counting.
4. After a wait loop, poll for COUNTER_READ_READY to set. After the bit is set, read SSW_CAPTURE_COUNT and SSW_CLKOUT_COUNT.
5. Compute the modulation depth as:

$$Depth = abs\left(1 - \frac{2 \times NR \times SSW\_CAPTURE\_COUNT}{SSW\_CLKOUT\_COUNT}\right) \qquad (12)$$

10.5.6 PLL Frequency Measurement Circuit

The same circuit that is used to measure modulation depth is also available to measure the average frequency of the PLL. In this mode, the PLL output (before the R-divider) is captured in SSW_CLKOUT_COUNT while the oscillator is captured in SSW_CAPTURE_COUNT. The procedure for using the PLL frequency measurement circuit is:

1. While the PLL is enabled, set EXT_COUNTER_EN.
2. Set COUNTER_EN. This bit clears both SSW_CAPTURE_COUNT and SSW_CLKOUT_COUNT and then immediately enables for counting.
3. Wait for some software delay loop.
4. Clear COUNTER_EN. Wait for COUNTER_READ_READY to set. Read both SSW_CAPTURE_COUNT and SSW_CLKOUT_COUNT and compute the ratio of PLL multiplication as:

$$\frac{NF}{NR \times OD} = \frac{SSW\_CLKOUT\_COUNT}{SSW\_CAPTURE\_COUNT} \tag{13}$$

5. Note that CAPTURE_WINDOW_INDEX, COUNTER_RESET, TAP_COUNTER_DIS are not used in this procedure.

10.5.7 PLL2

PLL2 drives GCM clock source 6.

The PLL is identical to PLL1, except modulation is disabled on this instance of the PLL. Also, the PLL typically does not clock the system, there is no automatic switch over feature. Any PLL error can be handled by the CPU.

PLL2 is programmed through PLLCTL3.

10.6 PLL Control Registers

The clock module has two registers (PLLCTL1 and PLLCTL2) located within the System and Peripheral Control Registers, plus it has four bits located in other System and Peripheral Control Registers.

The FM-PLL is off at power-on. The clock source is enabled by clearing the appropriate bit in the Clock Source Disable Register (CSDIS) or setting the appropriate bit in the Clock Source Disable Clear Register (CSDISCLR) of the System and Peripheral Control Registers. [CSDISCLR and Clock Source Disable Set Register (CSDISSET) also enable/disable the PLL and oscillator (and other clock sources).]

The LPOCLKDET module generates the OSCFAIL flag in the Global Status Register (GLBSTAT), of the System and Peripheral Control Registers, if a problem with the reference oscillator is detected. The slip signals are also registered in the RFSLIP and FBSLIP status flags in the Global Status Register (GLBSTAT), of the System and Peripheral Control Registers, in order to indicate the source of a clock failure.

The appropriate CLKSRnV bit for the PLL is set in the Clock Source Valid Status Register (CSVSTAT) of the System and Peripheral Control Registers.

The following sections describe the PLL registers used in the system module. These registers support 8, 16, and 32-bit write accesses. The reset values for these registers are configured so that an input frequency in the range from 5MHz to 20MHz generates a valid clock.

Table 10-4. PLL Module Registers

Offset	Acronym	Register Description	Section
FFFF FF30h	CSDIS	Clock Source Disable Register	Section 2.5.1.10
FFFF FF34h	CSDISSET	Clock Source Disable Set Register	Section 2.5.1.11
FFFF FF38h	CSDISCLR	Clock Source Disable Clear Register	Section 2.5.1.12
FFFF FF54h	CSVSTAT	Clock Source Valid Status Register	Section 2.5.1.19
FFFF FF70h	PLLCTL1	PLL Control 1 Register	Section 2.5.1.26
FFFF FF74h	PLLCTL2	PLL Control 2 Register	Section 2.5.1.27
FFFF E100h	PLLCTL3	PLL Control 3 Register	Section 2.5.2.1
FFFF FFA0h	GPREG1	General Purpose Register	Section 2.5.1.35
FFFF FFECh	GLBSTAT	Global Status Register	Section 2.5.1.51
FFFF E170h	CLKSLIP	PLL Clock Slip Control Register	Section 2.5.2.5
FFFF FF24h	SSWPLL1	PLL Modulation Depth Measurement Control Register	Section 10.6.1
FFFF FF28h	SSWPLL2	SSW PLL BIST Control Register 2	Section 10.6.2
FFFF FF2Ch	SSWPLL3	SSW PLL BIST Control Register 3	Section 10.6.3

Table 10-5. LPOCLKDET Module Registers

Offset	Acronym	Register Description	Section
FFFF FF88h	LPOMONCTL	LPO/Clock Monitor Control Register	Section 2.5.1.31
FFFF FF8Ch	CLKTEST	Clock Test Register	Section 2.5.1.32

10.6.1 PLL Modulation Depth Measurement Control Register (SSWPLL1)

Figure 10-6 illustrates this register and Table 10-6 provides the bit descriptions. This register applies to PLL1, but does not apply to PLL2.

Figure 10-6. SSW PLL BIST Control Register 1 (SSWPLL1) [offset = FF24h]

31–16
Reserved
R-0

15–8
CAPTURE_WINDOW_INDEX
R/W-0

7	6	5	4	3–1	0
Reserved	COUNTER_READ_READY	COUNTER_RESET	COUNTER_EN	TAP_COUNTER_DIS	EXT_COUNTER_EN
R-0	R-0	R/W-1	R/W-0	R/W-0	R/W-0

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

Table 10-6. SSW PLL BIST Control Register 1 (SSWPLL1) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Read returns 0. Writes have no effect.
15-8	CAPTURE_WINDOW_INDEX	0-FFh	The capture counter present in the PLL wrapper will count the PLL clock edges when the current modulation phase capture window value is equal to these bits. Should be set equal to NR.
7	Reserved	0	Read returns 0. Writes have no effect.
6	COUNTER_READ_READY		Counter read ready. Indicates that SSW_CAPTURE_COUNT (SSWPLL2) and SSW_CLKOUT_COUNT (SSWPLL3) can be read.
		0	Counter registers in SSWPLL2 and SSWPLL3 are not ready to read.
		1	Counter registers in SSWPLL2 and SSWPLL3 are ready to read.
5	COUNTER_RESET		Counter reset. If EXT_COUNTER_EN = 0, COUNTER_RESET resets SSW_CAPTURE_COUNT (SSWPLL2) and SSW_CLKOUT_COUNT (SSWPLL3). If EXT_COUNTER_EN = 1, this bit is ignored.
		0	No impact to counters
		1	If the EXT_COUNTER_EN bit is 0, then counters SSW_CAPTURE_COUNT and SSW_CLKOUT_COUNT will be held in the reset state. If EXT_COUNTER_EN bit is 1, then this bit will be ignored by the PLL wrapper.
4	COUNTER_EN		Counter enable. If EXT_COUNTER_EN = 0, COUNTER_EN initializes the modulation depth measurement. (In this mode, the disable is set to occur automatically.) If EXT_COUNTER_EN = 1, the counters are enabled/disabled with COUNTER_EN.
		0	If EXT_COUNTER_EN = 0, COUNTER_EN = 0 indicates that the counters are inactive. If EXT_COUNTER_EN = 1, COUNTER_EN = 0 disables the counters.
		1	If EXT_COUNTER_EN = 0, COUNTER_EN = 1 indicates that the counters are still active. If EXT_COUNTER_EN = 1, COUNTER_EN = 1 enables the counters.

Table 10-6. SSW PLL BIST Control Register 1 (SSWPLL1) Field Descriptions (continued)

Bit	Field	Value	Description
3-1	TAP_COUNTER_DIS		The value in this register is used to program a particular bit in CLKOUT counter. When that particular bit in CLKOUT counter becomes 1, then both the CLKOUT counter and the CAPTURE counter will stop counting when EXT_COUNTER_EN = 0. When EXT_COUNTER_EN = 1, this bit field is not used.
		0	Bit 16 of CLKOUT counter is selected. When this bit is set and the modulation period finishes, the counters are disabled and READ_READY_FLAG is set.
		1h	Bit 18 of CLKOUT counter is selected.
		2h	Bit 20 of CLKOUT counter is selected.
		3h	Bit 22 of CLKOUT counter is selected.
		4h	Bit 24 of CLKOUT counter is selected.
		5h	Bit 26 of CLKOUT counter is selected.
		6h	Bit 28 of CLKOUT counter is selected.
		7h	Bit 30 of CLKOUT counter is selected.
0	EXT_COUNTER_EN	0	Modulation Depth Measurement mode
		1	Frequency Measurement mode

10.6.2 SSW PLL BIST Control Register 2 (SSWPLL2)

This is an observation register used to log counter value for the capture counter inside the PLL wrapper. This register applies to PLL1, but does not apply to PLL2. The SSWPLL2 register is shown in Figure 10-7 and described in Table 10-7.

Figure 10-7. SSW PLL BIST Control Register 2 (SSWPLL2) [offset = FF28h]

31	16
SSW_CAPTURE_COUNT	
R-0	

15	0
SSW_CAPTURE_COUNT	
R-0	

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

Table 10-7. SSW PLL BIST Control Register 2 (SSWPLL2) Field Descriptions

Bit	Field	Value	Description
31-0	SSW_CAPTURE_COUNT	0-FFFF FFFFh	Capture count. This register returns the value of the capture count. When EXT_COUNTER_EN = 0, this counter increments within a fixed modulation window. When EXT_COUNTER_EN = 1, this counter increments based upon the oscillator.

10.6.3 SSW PLL BIST Control Register 3 (SSWPLL3)

This is observation register used to log counter value for CLKOUT counter inside PLL wrapper. This register applies to PLL1, but does not apply to PLL2. The SSWPLL3 register is shown in Figure 10-8 and described in Table 10-8.

Figure 10-8. SSW PLL BIST Control Register 3 (SSWPLL3) [offset = FF2Ch]

31	16
SSW_CLKOUT_COUNT	
R-0	

15	0
SSW_CLKOUT_COUNT	
R-0	

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

Table 10-8. SSW PLL BIST Control Register 3 (SSWPLL3) Field Descriptions

Bit	Field	Value	Description
31-0	SSW_CAPTURE_COUNT	0-FFFF FFFFh	Value of CLKout count register. This counter increments based upon the PLL output (prior to the R-divider).

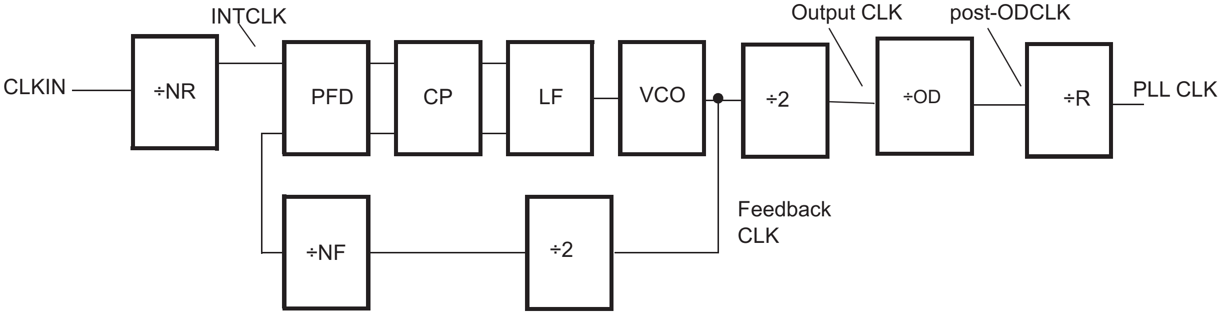
10.7 Phase-Locked Loop Theory of Operation

The PLL block consists of six logical sub-blocks:

- Phase-Frequency Detector (PFD)
- Charge Pump (CP)
- Loop Filter (LF)
- Voltage-Controlled Oscillator (VCO)
- Frequency Modulation
- Slip Detector

Figure 10-9 illustrates the sub-blocks in a basic PLL circuit. The VCO adjusts its frequency until the two signals into the PFD have the same phase and frequency. The feedback path (from VCO to PFD) divides the frequency of the feedback signal by 2 × NF; this feedback divider requires the VCO to generate a frequency 2 × NF times greater than the internal frequency (OSCIN/NR). In the forward path (from VCO to PLL CLK), the /2 block creates a clean duty cycle.

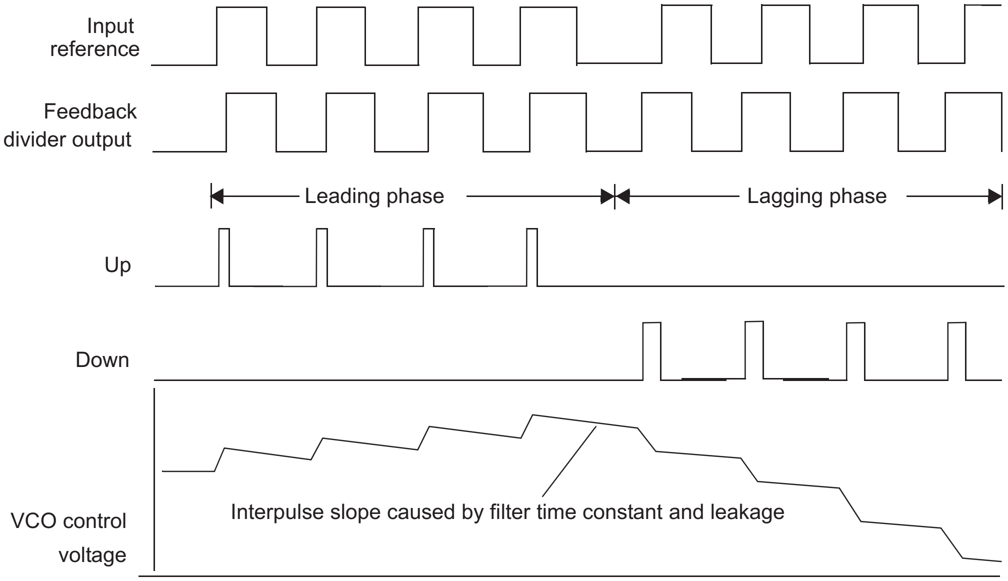
Figure 10-9. Basic PLL Circuit



10.7.1 Phase-Frequency Detector

The phase-frequency detector (PFD) compares the input reference phase/frequency to the phase/frequency of the feedback divider and generates two signals: an *up* pulse and a *down* pulse that drive a charge pump. The resulting charge, when integrated by the circuit at the LF pin, provides a VCO control voltage, as shown in Figure 10-10.

Figure 10-10. PFD Timing



The width of the up pulse and the down pulse depends on the difference in phase between the two inputs. For example, when the reference input leads the feedback input by 10 ns, then an up pulse of approximately 10 ns is generated (see Figure 10-10). On the other hand, when the reference input lags the feedback input by 10 ns, then a down pulse of approximately 10 ns is generated. When the two inputs are exactly in phase, the up pulse and down pulse become essentially zero-width. These pulses are fed to the charge pump block, which meters charge into the low-pass loop filter.

The advantage of a phase-frequency detector over a phase-only detector is that it cannot lock to a harmonic or subharmonic of the reference. This important property also ensures that the output frequency of the VCO is always exactly 2 × NF times the reference frequency.

The reference feedback frequency is based upon the VCO frequency and the feedback divider. Fractional multiplication is achieved by changing the feedback divider real-time in order to create the fractional multiplication. As an example, if a multiplier of 100.5 is selected, the feedback divider divides by 100 and 101 in equal proportions; in this case, the PLLMUL bit field would be programmed as 99.5 (0x6380). This fractional multiplication is useful when trying to achieve final frequencies that are non-integer to the input frequency (for example, a final frequency that is a prime number). The fractional portion of the divider should be small compared to the multiplier and so it is recommended that the fractional portion relate to parts in 16, implying that the last 4 bits should always be 0.

10.7.2 Charge Pump and Loop Filter

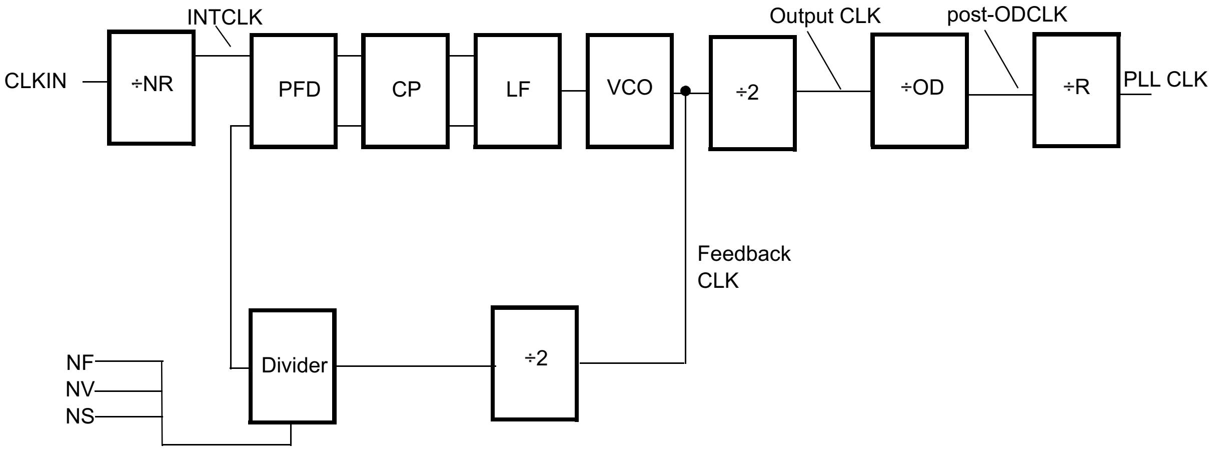
The charge pump (CP) add or remove charge from the loop filter based on the pulses coming from the phase-frequency detector (PFD).

Two components of the filter output signal are summed together: an integral component and a proportional component. The integral component maintains a DC level going to the VCO to set its frequency, and the proportional component makes the VCO track changes in phase to minimize jitter. The capacitors and resistors required for the filter are integrated in silicon.

10.7.3 Voltage-Controlled Oscillator

The output frequency of the VCO is proportional to its input control voltage, which is generated by the charge pump via the integrated loop filter. If the VCO oscillates too slowly, the feedback phase begins to lag the reference phase at the PFD, which increases the control voltage at the VCO. Conversely, if the VCO oscillates too fast, the feedback phase begins to lead the reference phase at the PFD, which decreases the control voltage at the VCO. These two actions keep the VCO running at the correct frequency multiple of the reference.

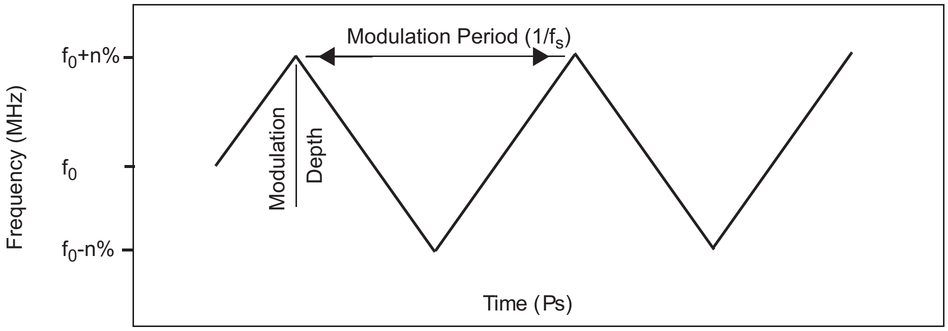
Figure 10-11. PLL Modulation Block Diagram



10.7.4 Frequency Modulation

The output clock of the PLL changes frequency in a controlled way, centered around the unmodulated output frequency. The modulation block directly modulates the VCO frequency at the loop filter, and creates the triangular frequency modulation (see Figure 10-12).

Figure 10-12. Frequency vs. Time



10.8 Programming Example

This section provides an example of how to program the PLL. For non-modulation settings, the PLLCTL1 and PLLCTL2 settings from 130nm process devices can be used without modification.

Suppose that, using a 20MHz crystal, the application requires:

- 180MHz GCLK (and HCLK) frequency
- 100 kHz spreading frequency
- 0.5% spreading depth

1. Choose an NR and NS such that:
   - $$\frac{f_{CLKIN}}{NR \times f_s} \geq 40 \quad (14)$$
   - $$f_s \equiv \frac{f_{CLKIN}}{2 \times NR \times NS} \quad (15)$$
   - $$2 \times NS = \frac{f_{CLKIN}}{NR \times f_s} \geq 40 \quad (16)$$
   - (NR,NS) = {(5,20), (4,25), (2,50), (1,100)}
   - Either NR = 5 and NS = 20 or NR = 4 and NS = 25 are reasonable. Another choice (NR = 3 and NS = 33) is possible, if the modulation frequency can vary from 100KHz.
2. Choose Output CLK frequency as integer divider of output frequency near to 330MHz. Output CLK frequency shall not exceed 550MHz or fall below 150MHz.

   The integer values for 180MHz are 360MHz or 540MHz. 360MHz is close to the target frequency of 330MHz and we use this frequency.
3. In this case, either of the following equations are suitable choices for getting to 360MHz. Choose NR = 5, NS = 20 and set NF = 90 or choose NR = 4, NS = 25 and set NF = 72.

$$\frac{f_{CLKIN}}{NR} = \frac{20[MHz]}{4} = 5[MHz] \ or \ \frac{f_{CLKIN}}{NR} = \frac{20[MHz]}{5} = 4[MHz] \quad (17)$$

4. Select the output divider OD so that the post-ODCLK frequency does not exceed the maximum frequency of output divider R (device specific frequency). In this case, choose OD = 2 and R = 1.
5. Compute the divider value NV:

$$Depth = \frac{0.5}{100} = \frac{NV}{NF} \times \frac{NS}{2} = \frac{NV}{90} \times \frac{20}{2} \tag{18}$$

   NV = 0.045
6. If it is important to maintain the same average frequency in modulation as in non-modulation, either NF should be modified OR program the MULMOD bit field. The modulation fields create a multiplier offset equal to:

$$\Delta NF = \frac{NV \times NS}{2} \tag{19}$$

   If using MULMOD[8:0], then:

$$\Delta NF = \frac{MULMOD[8...0]}{256} = \frac{NV \times NS}{2} = \frac{0.045 \times 20}{2} \tag{20}$$

$$MULMOD[8...0] = \frac{0.045 \times 20}{2} \times 256 = 115.2 \tag{21}$$

   MULMOD will be set to 115.
7. Convert the PLL parameters into bit field values:
   - NR = 5, implies that REFCLKDIV[5:0] = 4
   - NS = 20, implies that SPRATE[8:0] = 19 = 0x13
   - NF = 90, implies that PLLMUL[15:0] = 0x5900
   - OD = 2, implies that ODPLL[2:0] = 1
   - R = 1, implies that PLLDIV[4:0] = 0
   - NV = 0.045, implies that SPR_AMOUNT[8:0] = 91 = 0x5B
   - MULMOD[8:0] = 115 = 0x73
8. Setting only these fields (that is, not BPOS, ROF, or ROS) yields:

   PLLCTL1 = 0x00045900

   PLLCTL2 = 0x04C7325B

   When FM ENA is turned on, PLLCTL2 = 0x84C7325B

   The Output CLK is centered in the range from 150MHz to 550MHz at 360MHz.

   NF = 90 falls within the multiplier range from 1 to 256.

   OD is selected so that post-ODCLK meets the device specification.

Dual-Clock Comparator (DCC) Module

This chapter describes the dual-clock comparator (DCC) module.

Topic **Page**

11.1 Introduction 420
11.2 Module Operation 421
11.3 Clock Source Selection for Counter0 and Counter1 425
11.4 DCC Control Registers 426

11.1 Introduction

The primary purpose of a DCC module is to measure the frequency of a clock signal using a second known clock signal as a reference. This capability can be used to ensure the correct frequency range for several different device clock sources, thereby enhancing the system safety metrics.

11.1.1 Main Features

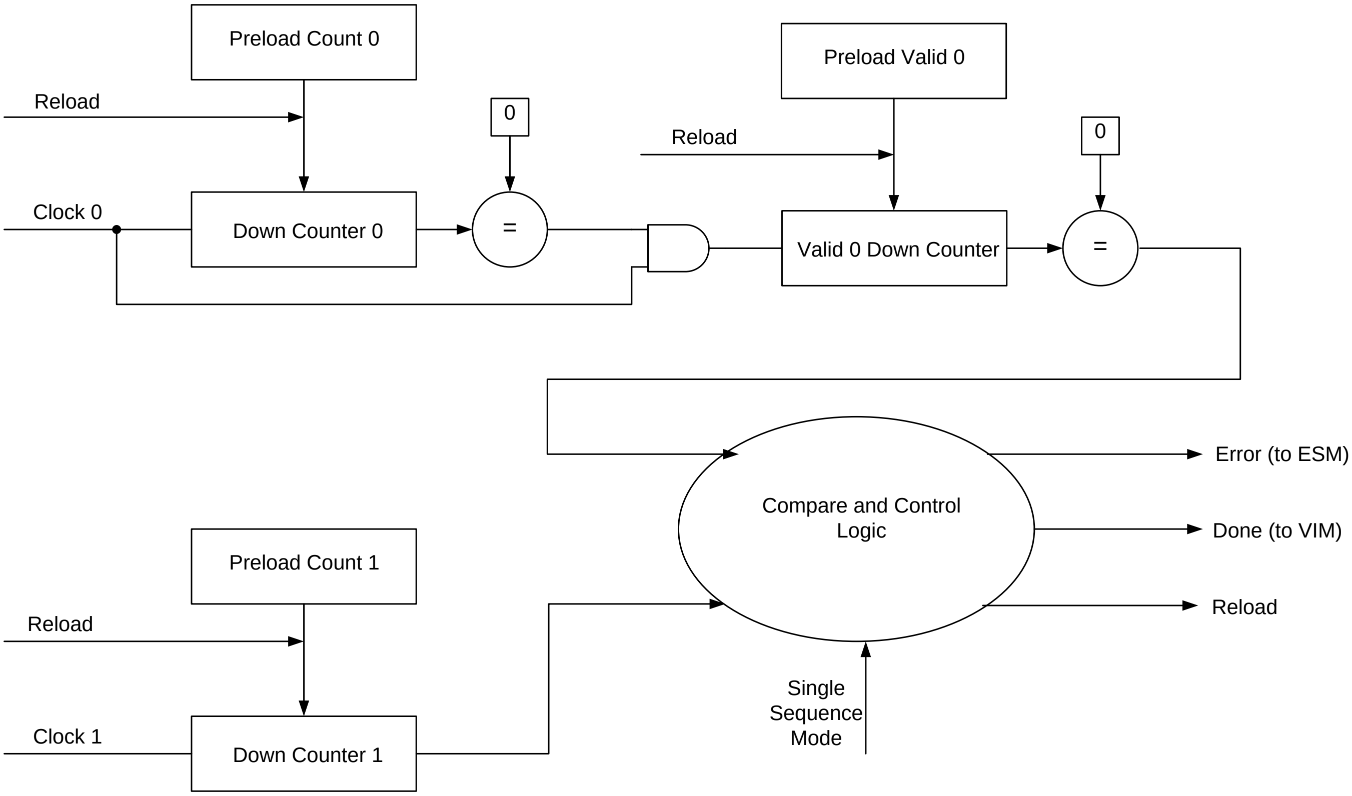
The main features of each of the DCC modules are:

- Allows application to ensure that a fixed ratio is maintained between frequencies of two clock signals
- Supports the definition of a programmable tolerance window in terms of number of reference clock cycles
- Supports continuous monitoring without requiring application intervention
- Also supports a single-sequence mode for spot measurements
- Allows selection of clock source for each of the counters resulting in several specific use cases

11.1.2 Block Diagram

Figure 11-1 illustrates the main concept of the DCC module.

Figure 11-1. DCC Operation



11.2 Module Operation

As shown in Figure 11-1, the DCC contains two counters – counter0 and counter1, which are driven by two signals – clock0 and clock1. The application programs the seed values for both these counters. The application also configures the tolerance window time by configuring the valid counter for clock0.

Counter0 and counter1 both start counting simultaneously once the DCC is enabled. When counter0 counts down to zero, this automatically triggers the count down of the tolerance window counter (valid0).

The DCC module can be used in two different operating modes:

11.2.1 Continuous Monitoring Mode

In this mode, the DCC is used by the application to ensure that two clock signals maintain the correct frequency ratio. Suppose the application wants to ensure that the PLL output signal (clock source # 1) always maintains a fixed frequency relationship with the main oscillator (clock source # 0).

- In this case, the application can use the main oscillator as the clock0 signal (for counter0 and valid0) and the PLL output as the clock1 (for counter1).
- The seed values of counter0, valid0 and counter1 are selected such that if the actual frequencies of clock0 and clock1 are equal to their expected frequencies, then the counter1 will reach zero either at the same time as counter0 or during the count down of the valid0 counter.
- If the counter1 reaches zero during the count down of the valid0 counter, then all the counters (counter0, valid0, counter1) are reloaded with their initial seed values once valid0 has also counted down to zero.
- This sequence of counting down and checking then continues as long as there is no error, or until the DCC module is disabled.
- The counters also all get reloaded if the application resets and restarts the DCC module.

Error Conditions:

An error condition is generated by any one of the following:

1. Counter1 counts down to 0 before Counter0 reaches 0. This means that clock1 is faster than expected, or clock0 is slower than expected. It includes the case when clock0 is stuck at 1 or 0.
2. Counter1 does not reach 0 even when Counter0 and Valid0 have both reached 0. This means that clock1 is slower than expected. It includes the case when clock1 is stuck at 1 or 0.

   Any error freezes the counters from counting. An application may then read out the counter values to help determine what caused the error.

11.2.1.1 Error Conditions

While operating in continuous mode, the counters get reloaded with the seed values and continue counting down under the following conditions:

- The module is reset or restarted by the application, OR
- Counter0, Valid 0 and Counter1 all reach 0 without any error

Figure 11-2. Counter Relationship

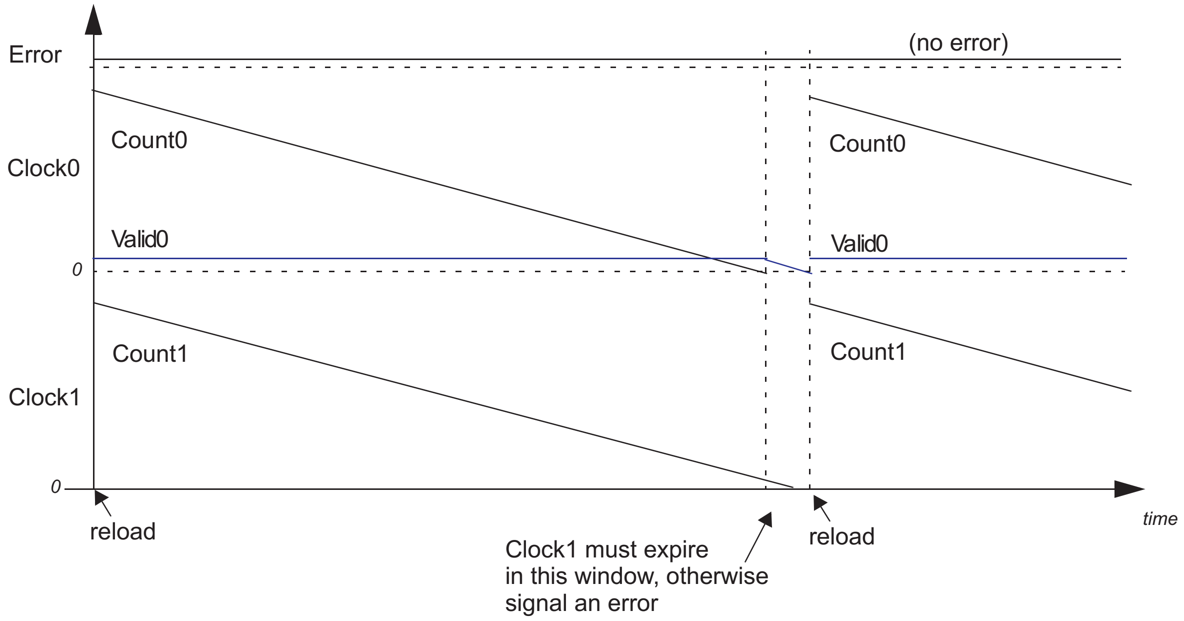


Figure 11-3. Clock1 Slower Than Clock0 - Results in an Error and Stops Counting

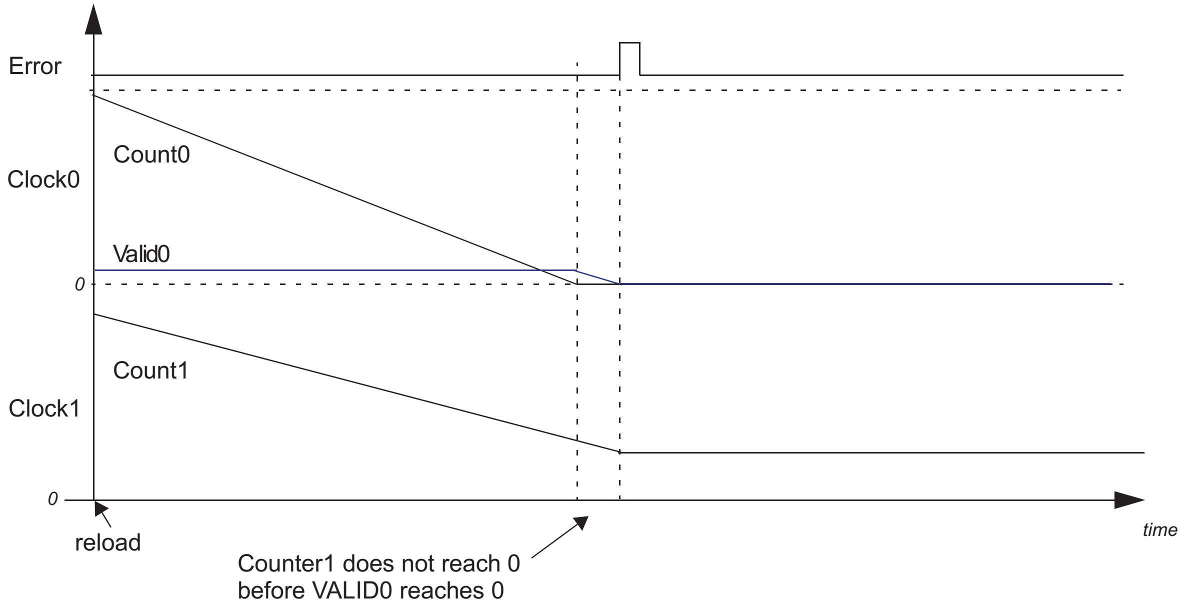


Figure 11-4. Clock1 Faster Than Clock0 - Results in an Error and Stops Counting

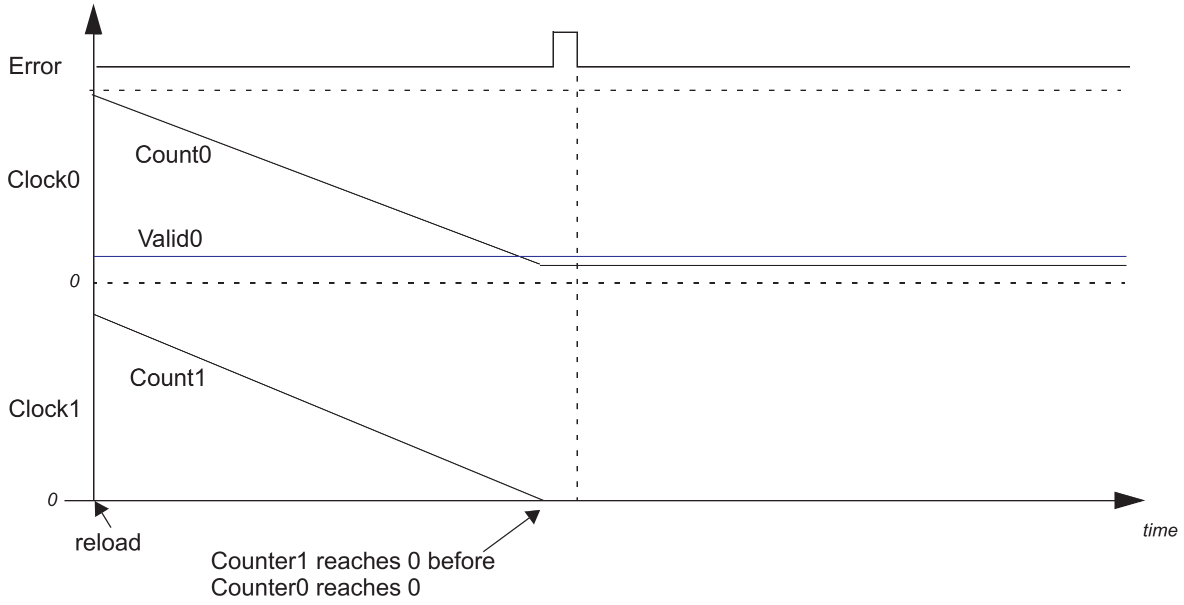


Figure 11-5. Clock1 Not Present - Results in an Error and Stops Counting

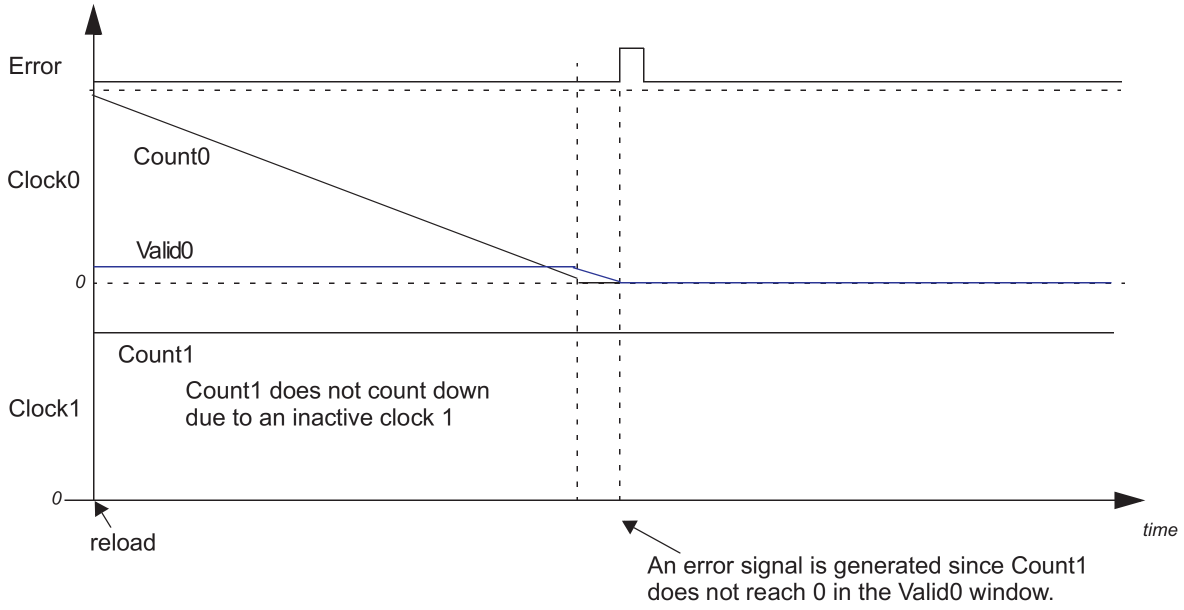
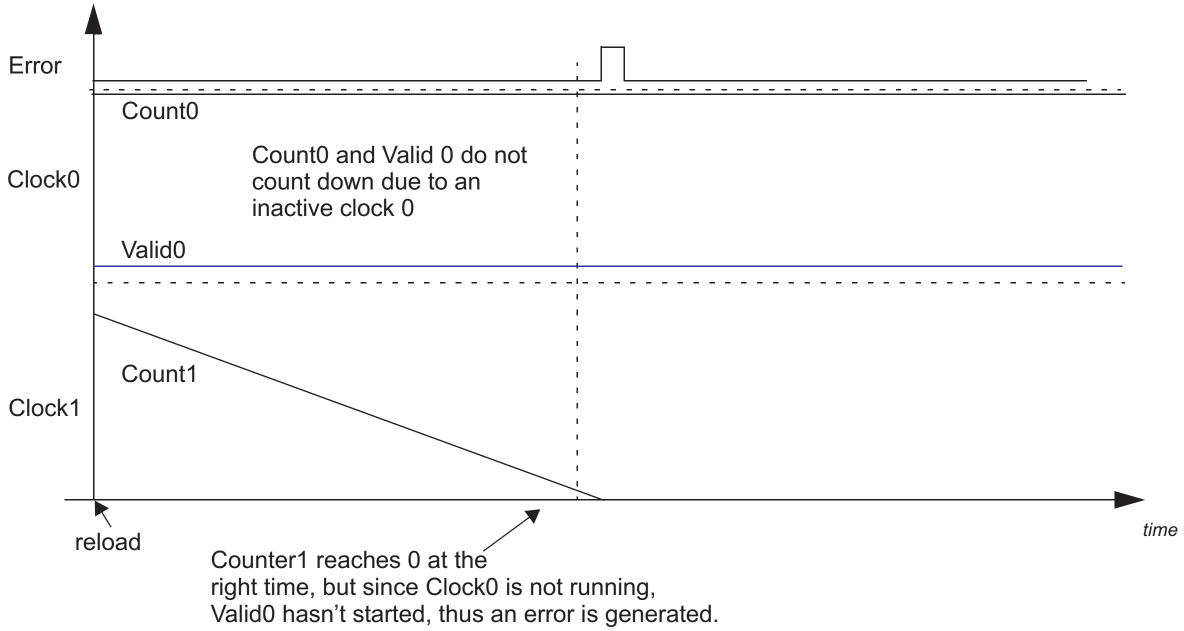

Error
Count0
Clock0
Valid0
0
Count1
Count1 does not count down
due to an inactive clock 1
Clock1
0
reload
time
An error signal is generated since Count1
does not reach 0 in the Valid0 window.

Figure 11-6. Clock0 Not Present - Results in an Error and Stops Counting


Error
Count0
Clock0
Count0 and Valid 0 do not count down due to an inactive clock 0
Valid0
Count1
Clock1
reload
time
Counter1 reaches 0 at the right time, but since Clock0 is not running, Valid0 hasn't started, thus an error is generated.

11.2.2 *Single-Shot Measurement Mode*

The DCC module can be programmed to count down one time by enabling the single-shot mode. In this mode, the DCC stops operating when the down counter0 and the valid counter0 reach 0. Alternatively, the DCC can be programmed to stop counting when the down counter1 reaches 0.

At the end of one sequence of counting down in this single-shot mode, the DCC gets disabled automatically, which prevents further counting. This mode is typically used for spot measurements of the frequency of a signal. This frequency could be an unknown for the application before the measurement.

Example Usage of Single-Shot Measurement Mode: Trimming the High-Frequency Low-Power Oscillator

A practical example of the usage of the spot measurement mode is in trimming the HF LPO (clock source # 5) using the main oscillator as a reference. This measurement sequence would proceed as follows:

- The application sets up the seed values for counter0 and valid0 for the duration of the measurement. Suppose the main oscillator frequency is 10MHz and the intended duration of the measurement is 500µs. The application needs to configure a seed value of 5000.
- These 5000 counts need to be divided between the counter0 and the valid0 counters. The minimum value for the valid0 seed is 4, so the application can configure counter0 seed value as 4996 and the valid0 seed value as 4.
- Suppose the HF LPO frequency is truly unknown. In this case the application can choose the maximum allowed seed value for counter1. This increases the probability of counter0 and valid0 counting down while the counter1 has still not fully counted down to zero. The maximum allowed seed value for counter1 is 1048575.
- Once the DCC is enabled, the counters counter0 and counter1 both start counting down from their seed values.
- When counter0 reaches zero, it automatically triggers the valid0 counter.
- When valid0 reaches zero, if counter1 is not zero as well, an ERROR status flag is set and a "DCC error" is sent to the ESM. Counter1 is also frozen so that it stops counting down any further. The application can enable an interrupt to be generated from the ESM whenever this DCC error is indicated. Refer the device datasheet to identify the ESM group and channel where the DCC error is connected.

- The DCC error interrupt service routine can then check the value of counter1 when the error was generated. Suppose that the counter1 now reads 1044575. This means that counter1 has counted 1048575 - 1044575, or 4000 cycles within the 500µs measurement period. This means that the average frequency of the HF LPO over this 500µs period was 4000 cycles / 500µs, or 8MHz.
- The application then needs to clear the ERROR status flag and restart the DCC module so that it is ready for the next spot measurement.

If there is no error generated at the end of the sequence, then the DONE status flag is set and a DONE interrupt is generated. The application must clear the DONE flag before restarting the DCC.

The conditions that cause a DCC error are identical between the continuous monitoring mode and the single-shot measurement mode.

Error Conditions:

An error condition is generated by any one of the following:

1. Counter1 counts down to 0 before Counter0 reaches 0. This means that clock1 is faster than expected, or clock0 is slower than expected. It includes the case when clock0 is stuck at 1 or 0.
2. Counter1 does not reach 0 even when Counter0 and Valid0 have both reached 0. This means that clock1 is slower than expected. It includes the case when clock1 is stuck at 1 or 0.

   Any error freezes the counters from counting. An application may then read out the counter values to help determine what caused the error.

Freezing Counters when Counter1 Reaches Zero:

The DCC module also allows the counters to be frozen when the counter1 reaches zero. This allows one of the clock sources for counter1 to be used as a reference for measuring one of the clock sources for counter0. The error conditions are the same as those where (counter0=0 and valid0=0) define the condition when the DCC counters are frozen. That is, an error is indicated if coutner0 and valid0 become zero while counter1 is still non-zero. In this case, however, the application would typically set up the seed values such that the counter1 will become zero before counter0. Essentially the measurement period is defined by the seed value of the counter1. Note that this is also an error condition, and the interrupt service routine can use the measurement period and the actual cycles counted by counter1 to determine the frequency of the clock0 signal.

11.3 Clock Source Selection for Counter0 and Counter1

Refer to the device datasheet to identify the available options for selecting the clock sources for both counters of the DCC module. Some microcontrollers may include multiple instances of the DCC module. This will also be identified in the device datasheet.

The selection of the clock sources for counter0 and coutner1 is done by a combination of the KEY, CNT0 CLKSRC, and CNT1 CLKSRC control fields of the CNT0CLKSRC and CNT1CLKSRC registers.

11.4 DCC Control Registers

This section describes the dual-clock comparator (DCC) module control and status registers. The registers support 8-bit, 16-bit or 32-bit writes and are aligned on a word (32-bit) boundary. Table 11-1 shows address offsets from the module base address. The base address for the control registers is FFFF EC00h for DCC1 and FFFF F400h for DCC2.

Table 11-1. DCC Control Registers

Offset	Acronym	Register Description	Section
00h	DCCGCTRL	DCC Global Control Register	Section 11.4.1
04h	DCCREV	DCC Revision Id Register	Section 11.4.2
08h	DCCCNT0SEED	DCC Counter0 Seed Register	Section 11.4.3
0Ch	DCCVALID0SEED	DCC Valid0 Seed Register	Section 11.4.4
10h	DCCCNT1SEED	DCC Counter1 Seed Register	Section 11.4.5
14h	DCCSTAT	DCC Status Register	Section 11.4.6
18h	DCCCNT0	DCC Counter0 Value Register	Section 11.4.7
1Ch	DCCVALID0	DCC Valid0 Value Register	Section 11.4.8
20h	DCCCNT1	DCC Counter1 Value Register	Section 11.4.9
24h	DCCCNT1CLKSRC	DCC Counter1 Clock Source Selection Register	Section 11.4.10
28h	DCCCNT0CLKSRC	DCC Counter0 Clock Source Selection Register	Section 11.4.11

11.4.1 DCC Global Control Register (DCCGCTRL)

Figure 11-7 and Table 11-2 describe the DCC Global Control register.

Figure 11-7. DCC Global Control Register (DCCGCTRL) [offset = 00]

31	16
Reserved	
R-0	

15	12	11	8	7	4	3	0
DONE INT ENA		SINGLE SHOT		ERR ENA		DCC ENA	
R/WP-5h		R/WP-5h		R/WP-5h		R/WP-5h	

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

Table 11-2. DCC Global Control Register (DCCGCTRL) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-12	DONE INT ENA		Done Interrupt Enable.
			Any operation mode read, privileged mode write:
		5h	No interrupt is generated when the DONE flag is set in the DCC Status (DCCSTAT) register.
		Others	DONE interrupt is generated when the DONE flag is set in the DCC Status (DCCSTAT) register.
11-8	SINGLE SHOT		Single-Shot Mode Enable.
			Any operation mode read, privileged mode write:
		Ah	DCC stops counting when counter0 and valid0 both reach zero.
		Bh	DCC stops counting when counter1 reaches zero.
		Others	DCC counts continuously and only stops when an error occurs.
7-4	ERR ENA		Error Interrupt Enable.
			Any operation mode read, privileged mode write:
		5h	No interrupt is generated when the ERR flag is set in the DCC Status (DCCSTAT) register.
		Others	ERROR interrupt is generated when the ERR flag is set in the DCC Status (DCCSTAT) register.
3-0	DCC ENA		DCC Enable.
			Any operation mode read, privileged mode write:
		5h	All DCC counters are stopped and error-checking is disabled. When an error occurs, the counters stop and this field is set to 5h automatically disabling the DCC counter in hardware.
		Others	Read: Counters are enabled.
			Write: Load counters with their seed values and begin counting. It is recommended to write Ah to enable counters to protect against single-bit errors.

11.4.2 DCC Revision Id Register (DCCREV)

Figure 11-8 and Table 11-3 describe the DCC Revision Id register.

Figure 11-8. DCC Revision Id Register (DCCREV) [offset = 4h]

31 30	29 28	27 16
SCHEME	Reserved	FUNC
R-01	R-0	R-0

15 11	10 8	7 6	5 0
RTL	MAJOR	CUSTOM	MINOR
R-0	R-2h	R-0	R-4h

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

Table 11-3. DCC Revision Id Register (DCCREV) Field Descriptions

Bit	Field	Value	Description
31-30	SCHEME	01	Reads return 01, writes have no effect.
29-28	Reserved	0	Reads return 0. Writes have no effect.
27-16	FUNC	0	Functional release number. Reads return 0x000, writes have no effect.
15-11	RTL	0	Design release number. Reads return 0x00, writes have no effect.
10-8	MAJOR	2h	Major revision number. Reads return 0x2, writes have no effect.
7-6	CUSTOM	0	Custom version number. Reads return 0x0, writes have no effect.
5-0	MINOR	4h	Minor revision number. Reads return 0x4, writes have no effect.

11.4.3 DCC Counter0 Seed Register (DCCCNT0SEED)

Figure 11-9 and Table 11-4 describe the DCC Counter0 Seed register.

Figure 11-9. DCC Counter0 Seed Register (DCCCNT0SEED) [offset = 8h]

31 20	19 16
Reserved	COUNT0 SEED
R-0	R/WP-0

15 0
COUNT0 SEED
R/WP-0

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

Table 11-4. DCC Counter0 Seed Register (DCCCNT0SEED) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reads return 0. Writes have no effect.
19-0	COUNT0 SEED		Seed value for DCC counter0. Reads in any operating mode return the current value of counter0. Writing in privileged mode only sets the current seed value for counter0.

NOTE: Seed for Counter0 must be non-zero

The DCC must only be enabled after programming a non-zero value in the COUNT0 SEED register.

11.4.4 DCC Valid0 Seed Register (DCCVALID0SEED)

Figure 11-10 and Table 11-5 describe the DCC Valid0 Seed register.

Figure 11-10. DCC Valid0 Seed Register (DCCVALID0SEED) [offset = Ch]

31	16
Reserved	
R-0	

15	0
VALID0 SEED	
R/WP-0	

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

Table 11-5. DCC Valid0 Seed Register (DCCVALID0SEED) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	VALID0 SEED		Seed value for DCC Valid0. This value defines the window within which the counter1 must reach 0. This window needs to be at least 4 cycles wide. Reads in any operating mode return the current value of seed for Valid0. Writing in privileged mode only sets the current seed value for Valid0. Writes in user mode are ignored.

NOTE: Seed for Valid0 must be at least 0x4

The DCC must only be enabled after programming a value greater than or equal to 0x4 in the VALID0 SEED register.

11.4.5 DCC Counter1 Seed Register (DCCCNT1SEED)

Figure 11-11 and Table 11-6 describe the DCC Counter1 Seed register.

Figure 11-11. DCC Counter1 Seed Register (DCCCNT1SEED) [offset = 10h]

31	20	19	16
Reserved		COUNT1 SEED	
R-0		R/WP-0	

15	0
COUNT1 SEED	
R/WP-0	

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

Table 11-6. DCC Counter1 Seed Register (DCCCNT0SEED) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reads return 0. Writes have no effect.
19-0	COUNT1 SEED		Seed value for DCC counter1. Reads in any operating mode return the current value of seed for counter1. Writing in privileged mode only sets the current seed value for counter1. Writes in user mode are ignored.

NOTE: Seed for Counter0 must be non-zero

The DCC must only be enabled after programming a non-zero value in the COUNT1 SEED register.

11.4.6 DCC Status Register (DCCSTAT)

Figure 11-7 and Table 11-2 describe the DCC Status register.

Figure 11-12. DCC Status Register (DCCSTAT) [offset = 14h]

31	16
Reserved	
R-0	

15	2	1	0
Reserved		DONE	ERR
R-0		R/W1CP-0	R/W1CP-0

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset

Table 11-7. DCC Status Register (DCCSTAT) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reads return 0. Writes have no effect.
1	DONE		Single-Shot Sequence Done flag. Indicates that a single-shot DCC sequence is done without any error.
		0	Read: Single-shot sequence is not done. Write: Writing 0 has no effect.
		1	Read: Single-shot sequence is done without any error. Write: Writing 1 in privileged mode clears the DONE flag.
0	ERR		Error flag. Indicates that a DCC error has occurred.
		0	Read: DCC error has not occurred. Write: Writing 0 has no effect.
		1	Read: An error has occurred. Write: Writing 1 in privileged mode clears the ERR flag.

11.4.7 DCC Counter0 Value Register (DCCCNT0)

Figure 11-13 and Table 11-8 describe the DCC Counter0 Value register.

Figure 11-13. DCC Counter0 Value Register (DCCCNT0) [offset = 18h]

31 - 20	19 - 16
Reserved	COUNT0
R-0	R-0

15 - 0
COUNT0
R-0

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

Table 11-8. DCC Counter0 Value Register (DCCCNT0) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reads return 0. Writes have no effect.
19-0	COUNT0		Current value of DCC counter0. Reads in any operating mode return the current value of counter0. Writes have no effect.

NOTE: Reads may not return exact current value of counter

Reading the counter0 value while counting is enabled may not return the exact value of the counter0.

11.4.8 DCC Valid0 Value Register (DCCVALID0)

Figure 11-14 and Table 11-9 describe the DCC Valid0 Value register.

Figure 11-14. DCC Valid0 Value Register (DCCVALID0) [offset = 1Ch]

31 16
Reserved
R-0

15 0
VALID0
R-0

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

Table 11-9. DCC Valid0 Value Register (DCCVALID0) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	VALID0		Current value for DCC Valid0. Reads in any operating mode return the current value of Valid0. Writes have no effect.

NOTE: Reads may not return exact current value of Valid0

Reading the Valid0 value while counting is enabled may not return the exact value of the Valid0.

11.4.9 DCC Counter1 Value Register (DCCCNT1)

Figure 11-15 and Table 11-10 describe the DCC Counter1 Value register.

Figure 11-15. DCC Counter1 Value Register (DCCCNT1) [offset = 20h]

31 20	19 16
Reserved	COUNT1
R-0	R/WP-0

15 0
COUNT1
R/WP-0

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

Table 11-10. DCC Counter1 Value Register (DCCCNT1) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reads return 0. Writes have no effect.
19-0	COUNT1		Current value for DCC counter1. Reads in any operating mode return the current value of counter1. Writes have no effect.

NOTE: Reads may not return exact current value of counter

Reading the counter1 value while counting is enabled may not return the exact value of the counter1.

11.4.10 DCC Counter1 Clock Source Selection Register (DCCCNT1CLKSRC)

Figure 11-15 and Table 11-10 describe the DCC Counter1 Clock Source Selection register.

Figure 11-16. DCC Counter1 Clock Source Selection Register (DCCCNT1CLKSRC) [offset = 24h]

31 16
Reserved
R-0

15 12	11 4	3 0
KEY	Reserved	CNT1 CLKSRC
R/WP-5h	R-0	R/WP-0

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

Table 11-11. DCC Counter1 Clock Source Selection Register (DCCCNT1CLKSRC) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-12	KEY		Key to enable clock source selection for counter1. Reads in any operating mode return the current value of the key.
			Writes in privileged mode set the key value.
		Ah	Writing Ah as the key enables the CNT1 CLKSRC field to define the clock source for counter1.
		Any other value	Writing any other value as the key disables the clock source selection for counter1. In this case, the N2HET signal is used as the source for counter1.
			Refer to the device datasheet for available clock source options and the KEY required to enable these options for counter1.
11-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	CNT1 CLKSRC		Clock source for counter1 when KEY is programmed to Ah. Reads in any operating mode return the current value of CLKSRC. Writes in privileged mode select the clock source for counter1. Refer to the device datasheet for available clock source options and the KEY required to enable these options for counter1.

11.4.11 DCC Counter0 Clock Source Selection Register (DCCCNT0CLKSRC)

Figure 11-15 and Table 11-10 describe the DCC Counter0 Clock Source Selection register.

Figure 11-17. DCC Counter0 Clock Source Selection Register (DCCCNT0CLKSRC) [offset = 28h]

31	16
Reserved	
R-0	

15	4	3	0
Reserved		CNT0 CLKSRC	
R-0		R/WP-5h	

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

Table 11-12. DCC Counter0 Clock Source Selection Register (DCCCNT0CLKSRC) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	CNT0 CLKSRC		Clock source for counter0 . Reads in any operating mode return the current value of CLKSRC. Writes in privileged mode select the clock source for counter0. Refer to the device datasheet for available clock source options for counter0.

Chapter 12

SPNU514C–March 2018

Error Signaling Module (ESM)

This chapter provides the details of the error signaling module (ESM) that aggregates device errors and provides the capability to define internal and external error response based on error severity.

Topic **Page**

12.1 Overview **436**
12.2 Module Operation **438**
12.3 Recommended Programming Procedure **442**
12.4 Control Registers **443**

12.1 Overview

The Error Signaling Module (ESM) collects and reports the various error conditions on the microcontroller. The error condition is categorized based on a severity level. Error response is then generated based on the category of the error. Possible error responses include a low-priority interrupt, high-priority interrupt, and an external pin action.

12.1.1 Features

- Up to 128 error channels are supported, divided into 3 different groups:
  - 64 Group1 (low-severity) channels with configurable interrupt generation and configurable $\overline{\text{ERROR}}$ pin behavior
  - 32 Group2 (high-severity) channels with predefined interrupt generation and predefined $\overline{\text{ERROR}}$ pin behavior
  - 32 Group3 (high-severity) channels with no interrupt generation and predefined $\overline{\text{ERROR}}$ pin behavior. These channels have no interrupt response as they are reserved for CPU based diagnostics that generate aborts directly to the CPU.
- Dedicated device $\overline{\text{ERROR}}$ pin to signal an external observer
- Configurable timebase for $\overline{\text{ERROR}}$ pin output
- Error forcing capability for latent fault testing

12.1.2 Block Diagram

As shown in Figure 12-1, the ESM channels are divided into three groups. Group1 channels are considered to be low-severity. Group1 errors have a configurable interrupt response and configurable $\overline{\text{ERROR}}$ pin behavior. Note that the ESM Status Register 1 (ESMSR1) for error group1 gets updated, regardless of whether an ESM interrupt for that Group1 channel is enabled or not. Group2 channels are connected to higher-severity error signals. Group2 errors generate a non-maskable high-priority interrupt to the CPU and assert the $\overline{\text{ERROR}}$ pin. Group3 channels indicate errors of the highest severity. Check the specific part's datasheet for identifying group3 errors and their expected responses. Group3 errors always generate an $\overline{\text{ERROR}}$ pin output.

The ESM interrupt and $\overline{\text{ERROR}}$ pin behavior are also summarized in Table 12-1.

Figure 12-1. Block Diagram

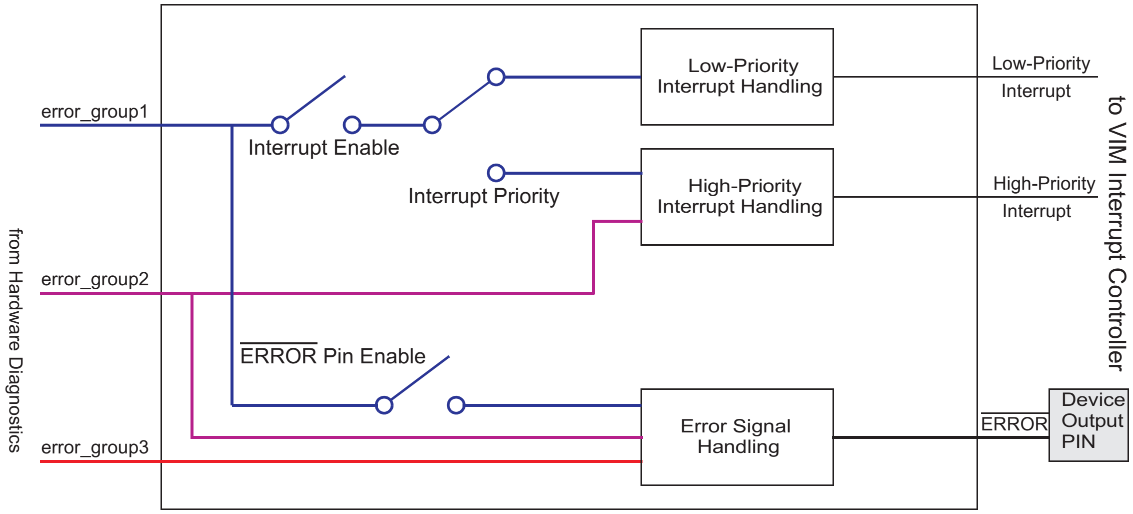


Table 12-1. ESM Interrupt and ERROR Pin Behavior

Error Group	Interrupt to CPU	Interrupt Priority	ERROR Pin Response
1	Can be enabled or disabled for each channel	Can be selected as low/high-priority for each channel	ERROR pin action can be selected for each channel separately
2	Cannot be disabled	High priority	ERROR pin is asserted
3	No interrupt	NA	ERROR pin is asserted

Figure 12-2 and Figure 12-3 show the interrupt response handling and ERROR pin response handling with register configuration. The total active time of the ERROR pin is controlled by the Low-Time Counter Preload register (LTCP) and the key register (ESMEPSR) as shown in Figure 12-3. See Section 12.2.2 for details.

Figure 12-2. Interrupt Response Handling

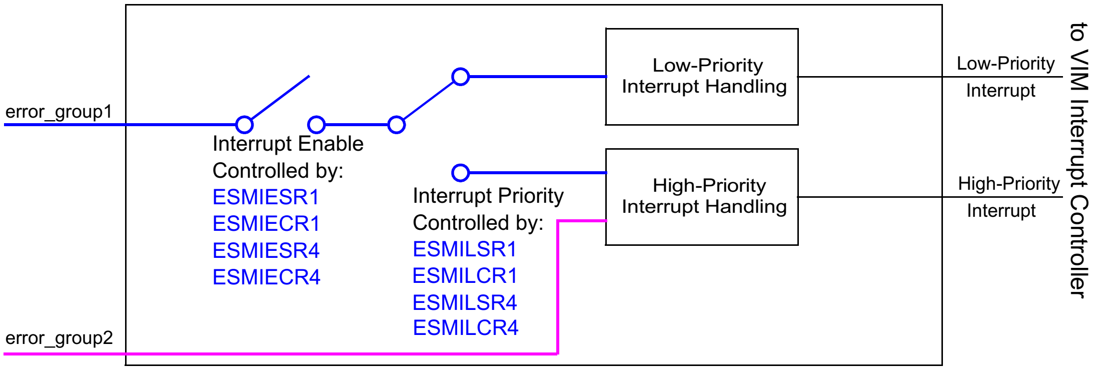
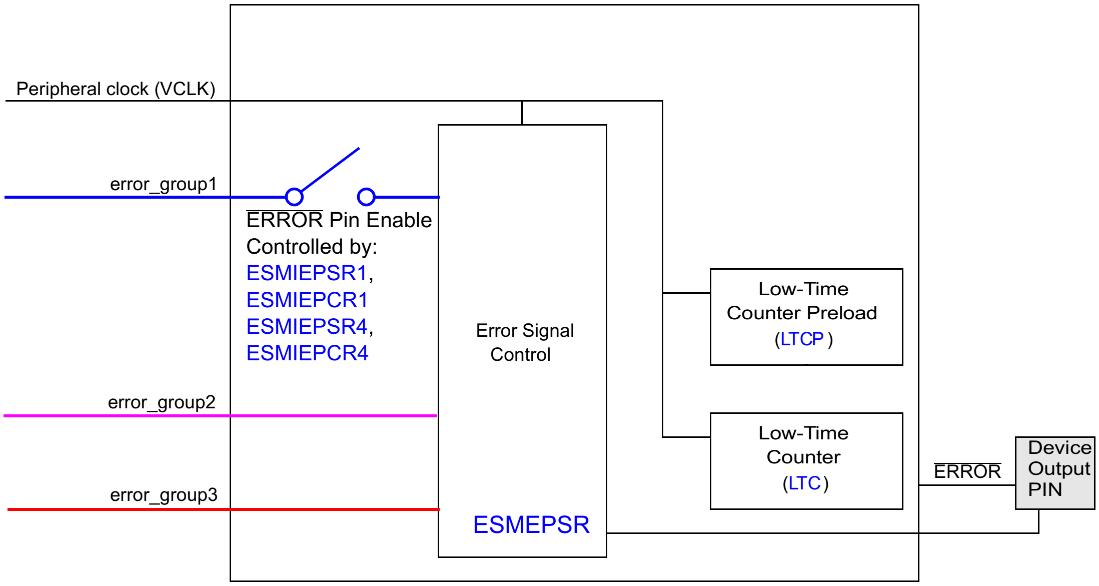


Figure 12-3. ERROR Pin Response Handling



12.2 Module Operation

This device has 128 error channels, divided into 3 different error groups. Please refer to the device datasheet for ESM channel assignment details.

The ESM module has error flags for each error channel. The error status registers ESMSR1, ESMSR4, ESMSR2, ESMSR3 provide status information on a pending error of Group1 (Channel 0-31), Group1 (Channel 32-63), Group2, and Group3, respectively. The ESMEPSR register provides the current ERROR status. The module also provides a status shadow register, ESMSSR2, which maintains the error flags of Group2 until power-on reset (PORRST) is asserted. See Section 12.2.1 for details of their behavior during power on reset and warm reset.

Once an error occurs, the ESM module will set the corresponding error flags. In addition, it can trigger an interrupt, ERROR pin outputs low depending on the ESM settings. Once the ERROR pin outputs low, a power on reset or a write of 5h to ESMEKR is required to release the ESM error pin back to normal state. See Section 12.2.2 for details. The application can read the error status registers (ESMSR1, ESMSR4, ESMSR2, and ESMSR3) to debug the error. If an RST is triggered or the error interrupt has been served, the error flag of Group2 should be read from ESMSSR2 because the error flag in ESMSR2 will be cleared by RST.

The user can also test the functionality of the ERROR pin by forcing an error. See Section 12.2.3 for details.

12.2.1 Reset Behavior

Power on reset:

- ERROR pin behavior

  When PORRST is active, the ERROR pin is in a high impedance state (output drivers disabled).
- Register behavior

  After PORRST, all registers in ESM module will be re-initialized to the default value. All the error status registers are cleared to zero.

Warm reset (RST):

- ERROR pin behavior

  During RST, the ERROR pin is in “output active” state with pull-down disabled. The ERROR pin remains unchanged after RST.
- Register behavior

  After RST, ESMSR1, ESMSR4, ESMSSR2, ESMSR3 and ESMEPSR register values remains unchanged. Since RST does not clear the critical failure registers, the user can read those registers to debug the failures after RST pin goes back to high.

  After RST, if one of the flags in ESMSR1 and ESMSR4 is set, the interrupt service routine will be called once the corresponding interrupt is enabled.

NOTE: ESMSR2 is cleared after RST. The flag in ESMSR2 gets cleared when reading the appropriate vector in the ESMIOFFHR offset register. Reading ESMIOFFHR will not clear the ESMSR1, ESMSR4, and the shadow register ESMSSR2. Reading ESMIOFFLR will also not clear the ESMSR1 and ESMSR4.

12.2.2 ERROR Pin Timing

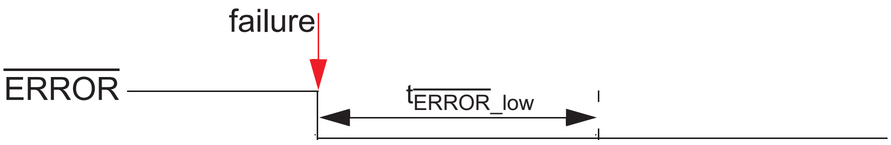
The ERROR pin is an active-low function. The state of the pin is also readable from ERROR Pin Status Register (ESMEPSR). The pin is in a high-impedance state during power-on reset. Once the ESM module drives the ERROR pin low, it remains in this state for the time specified by the Low-Time Counter Preload register (LTCPR). Based on the time period of the peripheral clock (VCLK), the total active time of the ERROR pin can be calculated as:

$$t_{\overline{ERROR}\_low} = t_{VCLK} \times (LTCP + 1) \qquad (22)$$

Once this period expires, the ERROR pin is set to high in case the reset of the ERROR pin was requested. This request is done by writing an appropriate key (5h) to the key register (ESMEKR) during the ERROR pin low time. Here are a few examples:

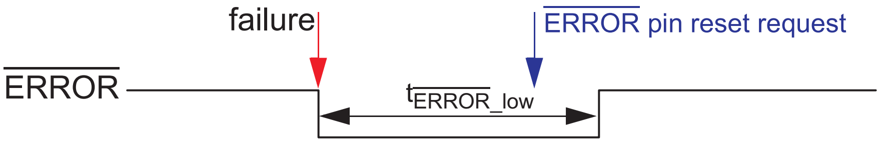
Example 1: ESM detects a failure and drives the ERROR pin low. No ERROR pin reset is requested. The ERROR pin continues outputting low until power on reset occurs.

Figure 12-4. ERROR Pin Timing - Example 1



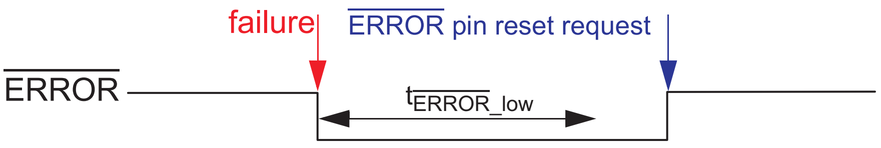
Example 2: ESM detects a failure and drives the ERROR pin low. An ERROR pin reset request is received before $t_{\overline{\text{ERROR}}\_\text{low}}$ expires. In this case, the ERROR pin is set to high immediately after $t_{\overline{\text{ERROR}}\_\text{low}}$ expires.

Figure 12-5. ERROR Pin Timing - Example 2



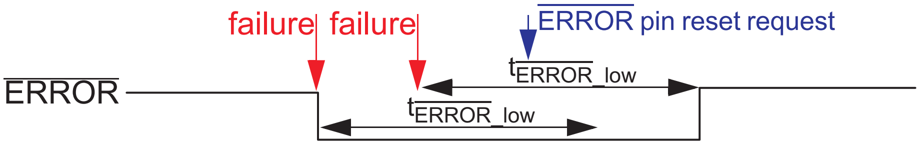
Example 3: ESM detects a failure and drives the ERROR pin low. An ERROR pin reset request is received after $t_{\overline{\text{ERROR}}\_\text{low}}$ expires. In this case, the ERROR pin is set to high immediately after ERROR pin reset request is received.

Figure 12-6. ERROR Pin Timing - Example 3



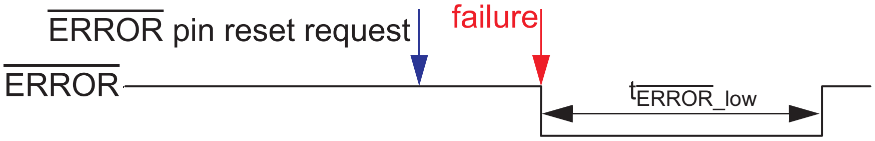
Example 4: ESM detects a failure and drives the $\overline{\text{ERROR}}$ pin low. Another failure occurs within the time the pin stays low. In this case, the low-time counter will be reset when the other failure occurs. In other words, $t_{\overline{\text{ERROR}}\_\text{low}}$ should be counted from whenever the most recent failure occurs.

Figure 12-7. $\overline{\text{ERROR}}$ Pin Timing - Example 4


failure
failure
$\overline{\text{ERROR}}$ pin reset request
$\overline{\text{ERROR}}$
$t_{\overline{\text{ERROR}}\_\text{low}}$
$t_{\overline{\text{ERROR}}\_\text{low}}$

Example 5: The reset of the $\overline{\text{ERROR}}$ pin was requested by the software even before the failure occurs. In this case, the $\overline{\text{ERROR}}$ pin is set to high immediately after $t_{\overline{\text{ERROR}}\_\text{low}}$ expires. This case is not recommended and should be avoided by the application.

Figure 12-8. $\overline{\text{ERROR}}$ Pin Timing - Example 5


$\overline{\text{ERROR}}$ pin reset request
failure
$\overline{\text{ERROR}}$
$t_{\overline{\text{ERROR}}\_\text{low}}$

Example 6: Failure1, then $\overline{\text{ERROR}}$ pin reset request, then Failure2 occurs before $\overline{\text{ERROR}}$ pin gets reset. In this case, the $\overline{\text{ERROR}}$ pin low-time is just extended (restarted) when the failure2 occurs and goes high when this count-down expires. There now is a scenario where the $\overline{\text{ERROR}}$ pin is high and the group2/3 status flag is set. To avoid this scenario, the application must write 5h followed by 0 to the ESM Error Key Register (ESMEKR). In this case, the $\overline{\text{ERROR}}$ pin will go high and then go low again to indicate the second failure.

12.2.3 Forcing an Error Condition

The error response generation mechanism is testable by software by forcing an error condition. This allows testing the ERROR pin functionality. By writing a dedicated key to the ESM Error Key Register (ESMEKR), the ERROR pin is set to low for the specified time. The following steps describe how to force an error condition:

1. Check ERROR Pin Status Register (ESMEPSR). This register must be 1 to switch into the error forcing mode.

   The ESM module cannot be switched into the error forcing mode if a failure has already been detected in functional mode. The application command to switch to error forcing mode is ignored.

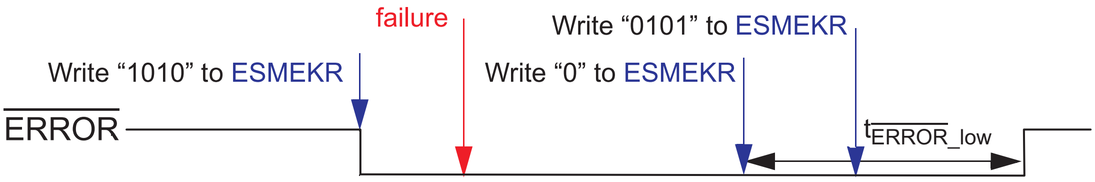
2. Write 5h to the ESM Error Key Register (ESMEKR). After that, the ERROR pin should output low (error force mode).

   Once the application puts the ESM module in the error forcing mode, the ERROR pin cannot indicate the normal error functionality. If a failure occurs during this time, it gets still latched and the LTC is reset and stopped. The error output pin is already driven low on account of the error forcing mode. When the ESM is forced back to normal functional mode, the LTC becomes active and forces the ERROR pin low until the expiration of the LTC (see Figure 12-9).

3. Write 0 to the ESM Error Key Register (ESMEKR) back to the active normal mode.

   If there are no errors detected while the ESM module is in the error forcing mode, the ERROR pin goes high immediately after exiting the error forcing mode.

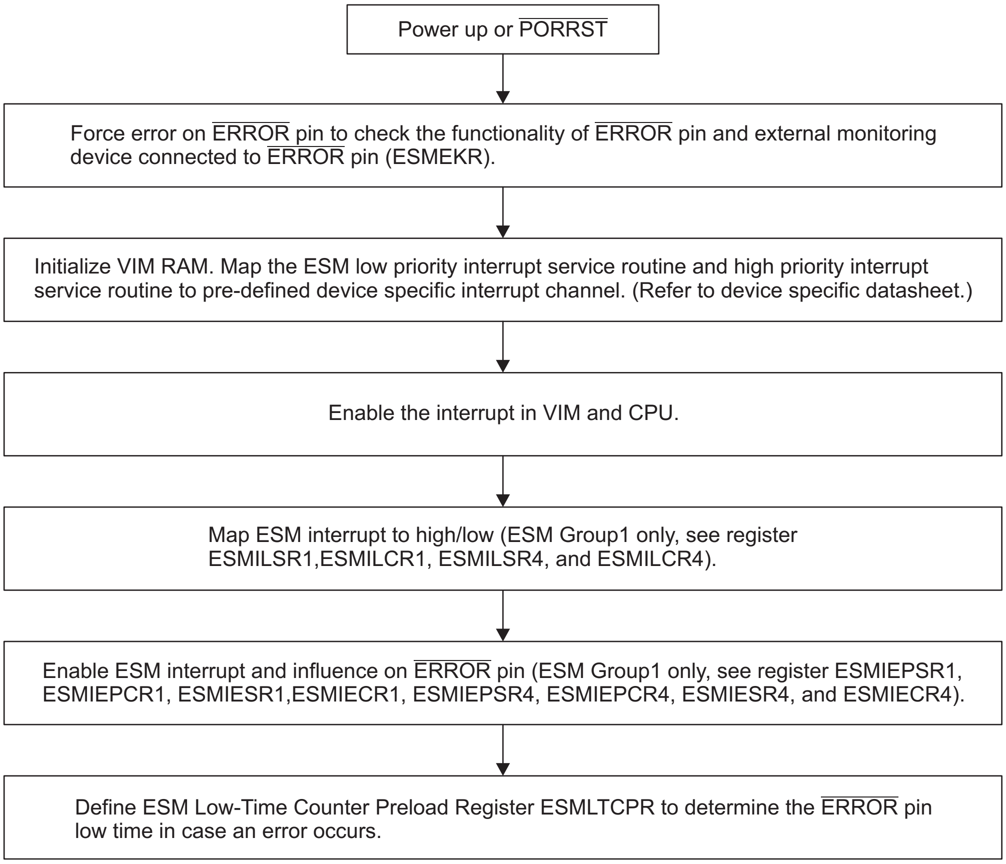
Figure 12-9. ERROR Pin Timing - Example 7



12.3 Recommended Programming Procedure

During the initialization stage, the application code should follow the recommendations in Figure 12-10 to initialize the ESM.

Figure 12-10. ESM Initialization


Power up or $\overline{\text{PORRST}}$
Force error on $\overline{\text{ERROR}}$ pin to check the functionality of $\overline{\text{ERROR}}$ pin and external monitoring device connected to $\overline{\text{ERROR}}$ pin (ESMEKR).
Initialize VIM RAM. Map the ESM low priority interrupt service routine and high priority interrupt service routine to pre-defined device specific interrupt channel. (Refer to device specific datasheet.)
Enable the interrupt in VIM and CPU.
Map ESM interrupt to high/low (ESM Group1 only, see register ESMILSR1,ESMILCR1, ESMILSR4, and ESMILCR4).
Enable ESM interrupt and influence on $\overline{\text{ERROR}}$ pin (ESM Group1 only, see register ESMIEPSR1, ESMIEPCR1, ESMIESR1,ESMIECR1, ESMIEPSR4, ESMIEPCR4, ESMIESR4, and ESMIECR4).
Define ESM Low-Time Counter Preload Register ESMLTCPR to determine the $\overline{\text{ERROR}}$ pin low time in case an error occurs.

Once an error occurs, it can trigger an interrupt, $\overline{\text{ERROR}}$ pin outputs low depending on the ESM settings. Once the $\overline{\text{ERROR}}$ pin outputs low, a power on reset or a write of 5h to ESMEKR is required to release the ESM back to normal state. The application can read the error status registers (ESMSR1, ESMSR4, ESMSR2, and ESMSR3) to debug the error. If an $\overline{\text{RST}}$ is triggered or the error interrupt has been served, the error flag of Group2 should be read from ESMSSR2 because the error flag in ESMSR2 will be cleared by $\overline{\text{RST}}$.

12.4 Control Registers

This section describes the ESM registers. Each register begins on a 32-bit word boundary. The registers support 32-bit, 16-bit, and 8-bit accesses. The base address for the registers is FFFF F500h.

Table 12-2. ESM Module Registers

Address	Acronym	Register Description	Section
FFFF F500h	ESMEEPAPR1	ESM Enable $\overline{\text{ERROR}}$ Pin Action/Response Register 1	Section 12.4.1
FFFF F504h	ESMDEPAPR1	ESM Disable $\overline{\text{ERROR}}$ Pin Action/Response Register 1	Section 12.4.2
FFFF F508h	ESMIESR1	ESM Interrupt Enable Set Register 1	Section 12.4.3
FFFF F50Ch	ESMIECR1	ESM Interrupt Enable Clear Register 1	Section 12.4.4
FFFF F510h	ESMILSR1	Interrupt Level Set Register 1	Section 12.4.5
FFFF F514h	ESMILCR1	Interrupt Level Clear Register 1	Section 12.4.6
FFFF F518h	ESMSR1	ESM Status Register 1	Section 12.4.7
FFFF F51Ch	ESMSR2	ESM Status Register 2	Section 12.4.8
FFFF F520h	ESMSR3	ESM Status Register 3	Section 12.4.9
FFFF F524h	ESMEPSR	ESM $\overline{\text{ERROR}}$ Pin Status Register	Section 12.4.10
FFFF F528h	ESMIOFFHR	ESM Interrupt Offset High Register	Section 12.4.11
FFFF F52Ch	ESMIOFFLR	ESM Interrupt Offset Low Register	Section 12.4.12
FFFF F530h	ESMLTCR	ESM Low-Time Counter Register	Section 12.4.13
FFFF F534h	ESMLTCPR	ESM Low-Time Counter Preload Register	Section 12.4.14
FFFF F538h	ESMEKR	ESM Error Key Register	Section 12.4.15
FFFF F53Ch	ESMSSR2	ESM Status Shadow Register 2	Section 12.4.16
FFFF F540h	ESMIEPSR4	ESM Influence $\overline{\text{ERROR}}$ Pin Set Register 4	Section 12.4.17
FFFF F544h	ESMIEPCR4	ESM Influence $\overline{\text{ERROR}}$ Pin Clear Register 4	Section 12.4.18
FFFF F548h	ESMIESR4	ESM Interrupt Enable Set Register 4	Section 12.4.19
FFFF F54Ch	ESMIECR4	ESM Interrupt Enable Clear Register 4	Section 12.4.20
FFFF F550h	ESMILSR4	Interrupt Level Set Register 4	Section 12.4.21
FFFF F554h	ESMILCR4	Interrupt Level Clear Register 4	Section 12.4.22
FFFF F558h	ESMSR4	ESM Status Register 4	Section 12.4.23

12.4.1 ESM Enable $\overline{\text{ERROR}}$ Pin Action/Response Register 1 (ESMEEPAPR1)

This register is dedicated for Group1.

Figure 12-11. ESM Enable $\overline{\text{ERROR}}$ Pin Action/Response Register 1 (ESMEEPAPR1) [address = FFFF F500h]

31	16
IEPSET	
R/WP-0	

15	0
IEPSET	
R/WP-0	

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

Table 12-3. ESM Enable $\overline{\text{ERROR}}$ Pin Action/Response Register 1 (ESMEEPAPR1) Field Descriptions

Bit	Field	Value	Description
31-0	IEPSET		Enable $\overline{\text{ERROR}}$ Pin Action/Response on Group 1. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: Failure on channel x has no influence on $\overline{\text{ERROR}}$ pin. Write: Leaves the bit and the corresponding clear bit in the ESMIEPCR1 register unchanged.
		1	Read: Failure on channel x has influence on $\overline{\text{ERROR}}$ pin. Write: Enables failure influence on $\overline{\text{ERROR}}$ pin and sets the corresponding clear bit in the ESMIEPCR1 register.

12.4.2 ESM Disable $\overline{\text{ERROR}}$ Pin Action/Response Register 1 (ESMDEPAPR1)

This register is dedicated for Group1.

Figure 12-12. ESM Disable $\overline{\text{ERROR}}$ Pin Action/Response Register 1 (ESMDEPAPR1) [address = FFFF F504h]

31	16
IEPCLR	
R/WP-0	

15	0
IEPCLR	
R/WP-0	

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

Table 12-4. ESM Disable $\overline{\text{ERROR}}$ Pin Action/Response Register 1 (ESMDEPAPR1) Field Descriptions

Bit	Field	Value	Description
31-0	IEPCLR		Disable $\overline{\text{ERROR}}$ Pin Action/Response on Group 1. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: Failure on channel x has no influence on $\overline{\text{ERROR}}$ pin. Write: Leaves the bit and the corresponding set bit in the ESMIEPSR1 register unchanged.
		1	Read: Failure on channel x has influence on $\overline{\text{ERROR}}$ pin. Write: Disables failure influence on $\overline{\text{ERROR}}$ pin and clears the corresponding set bit in the ESMIEPSR1 register.

12.4.3 ESM Interrupt Enable Set Register 1 (ESMIESR1)

This register is dedicated for Group1.

Figure 12-13. ESM Interrupt Enable Set Register 1 (ESMIESR1) [address = FFFF F508h]

31	16
INTENSET	
R/WP-0	

15	0
INTENSET	
R/WP-0	

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

Table 12-5. ESM Interrupt Enable Set Register 1 (ESMIESR1) Field Descriptions

Bit	Field	Value	Description
31-0	INTENSET		Set interrupt enable. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: Interrupt is disabled. Write: Leaves the bit and the corresponding clear bit in the ESMIECR1 register unchanged.
		1	Read: Interrupt is enabled. Write: Enables interrupt and sets the corresponding clear bit in the ESMIECR1 register.

12.4.4 ESM Interrupt Enable Clear Register 1 (ESMIECR1)

This register is dedicated for Group1.

Figure 12-14. ESM Interrupt Enable Clear Register 1 (ESMIECR1) [address = FFFF F50Ch]

31	16
INTENCLR	
R/WP-0	

15	0
INTENCLR	
R/WP-0	

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

Table 12-6. ESM Interrupt Enable Clear Register 1 (ESMIECR1) Field Descriptions

Bit	Field	Value	Description
31-0	INTENCLR		Clear interrupt enable. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: Interrupt is disabled. Write: Leaves the bit and the corresponding set bit in the ESMIESR1 register unchanged.
		1	Read: Interrupt is enabled. Write: Disables interrupt and clears the corresponding set bit in the ESMIESR1 register.

12.4.5 ESM Interrupt Level Set Register 1 (ESMILSR1)

This register is dedicated for Group1.

Figure 12-15. ESM Interrupt Level Set Register 1 (ESMILSR1) [address = FFFF F510h]

31	16
INTLVLSET	
R/WP-0	

15	0
INTLVLSET	
R/WP-0	

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

Table 12-7. ESM Interrupt Level Set Register 1 (ESMILSR1) Field Descriptions

Bit	Field	Value	Description
31-0	INTLVLSET		Set interrupt priority.
			Read in User and Privileged mode. Write in Privileged mode only.
		0	Read: Interrupt of channel x is mapped to low-level interrupt line.
			Write: Leaves the bit and the corresponding clear bit in the ESMILCR1 register unchanged.
		1	Read: Interrupt of channel x is mapped to high-level interrupt line.
			Write: Maps interrupt of channel x to high-level interrupt line and sets the corresponding clear bit in the ESMILCR1 register.

12.4.6 ESM Interrupt Level Clear Register 1 (ESMILCR1)

This register is dedicated for Group1.

Figure 12-16. ESM Interrupt Level Clear Register 1 (ESMILCR1) [address = FFFF F514h]

31	16
INTLVLCLR	
R/WP-0	

15	0
INTLVLCLR	
R/WP-0	

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

Table 12-8. ESM Interrupt Level Clear Register 1 (ESMILCR1) Field Descriptions

Bit	Field	Value	Description
31-0	INTLVLCLR		Clear interrupt priority.
			Read in User and Privileged mode. Write in Privileged mode only.
		0	Read: Interrupt of channel x is mapped to low-level interrupt line.
			Write: Leaves the bit and the corresponding set bit in the ESMILSR1 register unchanged.
		1	Read: Interrupt of channel x is mapped to high-level interrupt line.
			Write: Maps interrupt of channel x to low-level interrupt line and clears the corresponding set bit in the ESMILSR1 register.

12.4.7 ESM Status Register 1 (ESMSR1)

This register is dedicated for Group1. Note that the ESMSR1 status register will get updated if an error condition occurs, regardless if the corresponding interrupt enable flag is set or not.

Figure 12-17. ESM Status Register 1 (ESMSR1) [address = FFFF F518h]

31	16
ESF	
R/W1CP-X/0	

15	0
ESF	
R/W1CP-X/0	

LEGEND: R/W = Read/Write; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset/$\overline{\text{PORRST}}$; *X* = Value unchanged

Table 12-9. ESM Status Register 1 (ESMSR1) Field Descriptions

Bit	Field	Value	Description
31-0	ESF		Error Status Flag. Provides status information on a pending error. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: No error occurred; no interrupt is pending. Write: Leaves the bit unchanged.
		1	Read: Error occurred; interrupt is pending. Write: Clears the bit. **Note:** After $\overline{\text{RST}}$, if one of these flags are set and the corresponding interrupt are enabled, the interrupt service routine will be called.

12.4.8 ESM Status Register 2 (ESMSR2)

This register is dedicated for Group2.

Figure 12-18. ESM Status Register 2 (ESMSR2) [address = FFFF F51Ch]

31	16
ESF	
R/W1CP-0	

15	0
ESF	
R/W1CP-0	

LEGEND: R/W = Read/Write; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset

Table 12-10. ESM Status Register 2 (ESMSR2) Field Descriptions

Bit	Field	Value	Description
31-0	ESF		Error Status Flag. Provides status information on a pending error. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: No error occurred; no interrupt is pending. Write: Leaves the bit unchanged.
		1	Read: Error occurred; interrupt is pending. Write: Clears the bit. ESMSSR2 is not impacted by this action. **Note:** In normal operation, the flag gets cleared when reading the appropriate vector in the ESMIOFFHR offset register. Reading ESMIOFFHR will not clear the ESMSR1 and the shadow register ESMSSR2.

12.4.9 ESM Status Register 3 (ESMSR3)

This register is dedicated for Group3.

Figure 12-19. ESM Status Register 3 (ESMSR3) [address = FFFF F520h]

31 ... 16
ESF
R/W1CP-X/0

15 ... 0
ESF
R/W1CP-X/0

LEGEND: R/W = Read/Write; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset/$\overline{\text{PORRST}}$; *X* = Value unchanged

Table 12-11. ESM Status Register 3 (ESMSR3) Field Descriptions

Bit	Field	Value	Description
31-0	ESF		Error Status Flag. Provides status information on a pending error. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: No error occurred. Write: Leaves the bit unchanged.
		1	Read: Error occurred. Write: Clears the bit.

12.4.10 ESM $\overline{\text{ERROR}}$ Pin Status Register (ESMEPSR)

Figure 12-20. ESM $\overline{\text{ERROR}}$ Pin Status Register (ESMEPSR) [address = FFFF F524h]

31 ... 16
Reserved
R-0

15 ... 1	0
Reserved	EPSF
R-0	R-X/1

LEGEND: R = Read only; -*n* = value after reset/$\overline{\text{PORRST}}$; *X* = Value unchanged

Table 12-12. ESM $\overline{\text{ERROR}}$ Pin Status Register (ESMEPSR) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	EPSF		$\overline{\text{ERROR}}$ Pin Status Flag. Provides status information for the $\overline{\text{ERROR}}$ pin. **Read/Write in User and Privileged mode.**
		0	Read: $\overline{\text{ERROR}}$ pin is low (active) if any error has occurred. Write: Writes have no effect.
		1	Read: $\overline{\text{ERROR}}$ pin is high if no error has occurred. Write: Writes have no effect. **Note:** This flag will be set to 1 after $\overline{\text{PORRST}}$. The value will be unchanged after $\overline{\text{RST}}$. The $\overline{\text{ERROR}}$ pin status remains unchanged after $\overline{\text{RST}}$.

12.4.11 ESM Interrupt Offset High Register (ESMIOFFHR)

Figure 12-21. ESM Interrupt Offset High Register (ESMIOFFHR) [address = FFFF F528h]

31	16
Reserved	
R-0	

15	7	6	0
Reserved		INTOFFH	
R-0		R-0	

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

Table 12-13. ESM Interrupt Offset High Register (ESMIOFFHR) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reads return 0. Writes have no effect.
6-0	INTOFFH		Offset High-Level Interrupt. This vector gives the channel number of the highest-pending interrupt request for the high-level interrupt line. Interrupts of error Group2 have higher priority than interrupts of error Group1. Inside a group, channel 0 has highest priority and channel 31 has lowest priority.
			User and privileged mode (read):
			Returns number of pending interrupt with the highest priority for the high-level interrupt line.
		0	No pending interrupt.
		1h	Interrupt pending for channel 0, error Group1.
		:	:
		20h	Interrupt pending for channel 31, error Group1.
		21h	Interrupt pending for channel 0, error Group2.
		:	:
		40h	Interrupt pending for channel 31, error Group2.
		41h	Interrupt pending for channel 32, error Group1.
		:	:
		60h	Interrupt pending for channel 63, error Group1.
			Note: Reading the interrupt vector will clear the corresponding flag in the ESMSR2 register; will **not** clear ESMSR1 and ESMSSR2 and the offset register gets updated.
			User and privileged mode (write): Writes have no effect.

12.4.12 ESM Interrupt Offset Low Register (ESMIOFFLR)

Figure 12-22. ESM Interrupt Offset Low Register (ESMIOFFLR) [address = FFFF F52Ch]

31 16
Reserved
R-0

15 7	6 0
Reserved	INTOFFL
R-0	R-0

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

Table 12-14. ESM Interrupt Offset Low Register (ESMIOFFLR) Field Descriptions

Bit	Field	Value	Description
31-7	Reserved	0	Reads return 0. Writes have no effect.
6-0	INTOFFL		Offset Low-Level Interrupt. This vector gives the channel number of the highest-pending interrupt request for the low-level interrupt line. Inside a group, channel 0 has highest priority and channel 31 has lowest priority.
			User and privileged mode (read):
			Returns number of pending interrupt with the highest priority for the low-level interrupt line.
		0	No pending interrupt.
		1h	Interrupt pending for channel 0, error Group1.
		:	:
		20h	Interrupt pending for channel 31, error Group1.
		21h-40h	Reserved
		41h	Interrupt pending for channel 32, error Group1.
		:	:
		60h	Interrupt pending for channel 63, error Group1.
			Note: Reading the interrupt vector will **not** clear the corresponding flag in the ESMSR1 register. Group2 interrupts are fixed to the high-level interrupt line only.
			User and privileged mode (write):
			Writes have no effect.

12.4.13 ESM Low-Time Counter Register (ESMLTCR)

Figure 12-23. ESM Low-Time Counter Register (ESMLTCR) [address = FFFF F530h]

31	16
Reserved	
R-0	

15	0
LTC	
R-3FFFh	

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

Table 12-15. ESM Low-Time Counter Register (ESMLTCR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	LTC		ERROR Pin Low-Time Counter 16-bit preloadable down-counter to control low-time of $\overline{\text{ERROR}}$ pin. The low-time counter is triggered by the peripheral clock (VCLK). **Note:** Low-time counter is set to the default preload value of the ESMLTCPR in the following cases: 1. Reset (power on reset or warm reset) 2. An error occurs 3. User forces an error

12.4.14 ESM Low-Time Counter Preload Register (ESMLTCPR)

Figure 12-24. ESM Low-Time Counter Preload Register (ESMLTCPR) [address = FFFF F534h]

31	16
Reserved	
R-0	

15 14	13 0
LTCP	LTCP
R/WP-0	R-3FFFh

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

Table 12-16. ESM Low-Time Counter Preload Register (ESMLTCPR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	LTCP	0-FFFFh	$\overline{\text{ERROR}}$ Pin Low-Time Counter Pre-load Value 16-bit preload value for the $\overline{\text{ERROR}}$ pin low-time counter. Defines the minimum period for which the $\overline{\text{ERROR}}$ pin will be driven to 16384 VCLK cycles. **Note:** Only LTCP[15] and LTCP[14] are configurable (privileged mode write).

12.4.15 ESM Error Key Register (ESMEKR)

Figure 12-25. ESM Error Key Register (ESMEKR) [address = FFFF F538h]

31	16
Reserved	
R-0	

15	4	3	0
Reserved		EKEY	
R-0		R/WP-0	

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

Table 12-17. ESM Error Key Register (ESMEKR) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	EKEY		Error Key. The key to reset the $\overline{\text{ERROR}}$ pin or to force an error on the $\overline{\text{ERROR}}$ pin. **User and privileged mode (read):** Returns current value of the EKEY.
			Privileged mode (write):
		0	Activates normal mode (recommended default mode).
		5h	The $\overline{\text{ERROR}}$ pin set to high when the low-time counter (LTC) has completed; then the EKEY bit will switch back to normal mode (EKEY = 0000).
		Ah	Forces error on $\overline{\text{ERROR}}$ pin.
		All other values	Activates normal mode.

12.4.16 ESM Status Shadow Register 2 (ESMSSR2)

This register is dedicated for Group2.

Figure 12-26. ESM Status Shadow Register 2 (ESMSSR2) [address = FFFF F53Ch]

31	16
ESF	
R/W1CP-X/0	

15	0
ESF	
R/W1CP-X/0	

LEGEND: R/W = Read/Write; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset/$\overline{\text{PORRST}}$; *X* = Value unchanged

Table 12-18. ESM Status Shadow Register 2 (ESMSSR2) Field Descriptions

Bit	Field	Value	Description
31-0	ESF		Error Status Flag. Shadow register for status information on pending error. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: No error occurred. Write: Leaves the bit unchanged.
		1	Read: Error occurred. Write: Clears the bit. ESMSR2 is not impacted by this action. **Note:** Errors are stored until they are cleared by the software or at power-on reset ($\overline{\text{PORRST}}$).

12.4.17 ESM Influence $\overline{\text{ERROR}}$ Pin Set Register 4 (ESMIEPSR4)

This register is dedicated for Group1.

Figure 12-27. ESM Influence $\overline{\text{ERROR}}$ Pin Set Register 4 (ESMIEPSR4) [address = FFFF F540h]

31	16
IEPSET[63:48]	
R/WP-0	

15	0
IEPSET[47:32]	
R/WP-0	

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

Table 12-19. ESM Influence $\overline{\text{ERROR}}$ Pin Set Register 4 (ESMIEPSR4) Field Descriptions

Bit	Field	Value	Description
63-32	IEPSET		Set influence on $\overline{\text{ERROR}}$ pin. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: Failure on channel x has no influence on $\overline{\text{ERROR}}$ pin. Write: Leaves the bit and the corresponding clear bit in the ESMIEPCR4 register unchanged.
		1	Read: Failure on channel x has influence on $\overline{\text{ERROR}}$ pin. Write: Enables failure influence on $\overline{\text{ERROR}}$ pin and sets the corresponding clear bit in the ESMIEPCR4 register.

12.4.18 ESM Influence $\overline{\text{ERROR}}$ Pin Clear Register 4 (ESMIEPCR4)

This register is dedicated for Group1.

Figure 12-28. ESM Influence $\overline{\text{ERROR}}$ Pin Clear Register 4 (ESMIEPCR4) [address = FFFF F544h]

31	16
IEPCLR[63:48]	
R/WP-0	

15	0
IEPCLR[47:32]	
R/WP-0	

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

Table 12-20. ESM Influence $\overline{\text{ERROR}}$ Pin Clear Register 4 (ESMIEPCR4) Field Descriptions

Bit	Field	Value	Description
63-32	IEPCLR		Clear influence on $\overline{\text{ERROR}}$ pin. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: Failure on channel x has no influence on $\overline{\text{ERROR}}$ pin. Write: Leaves the bit and the corresponding set bit in the ESMIEPSR4 register unchanged.
		1	Read: Failure on channel x has influence on $\overline{\text{ERROR}}$ pin. Write: Disables failure influence on $\overline{\text{ERROR}}$ pin and clears the corresponding set bit in the ESMIEPSR4 register.

12.4.19 ESM Interrupt Enable Set Register 4 (ESMIESR4)

This register is dedicated for Group1.

Figure 12-29. ESM Interrupt Enable Set Register 4 (ESMIESR4) [address = FFFF F548h]

31		16
	INTENSET[63:48]	
	R/WP-0	

15		0
	INTENSET[47:32]	
	R/WP-0	

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

Table 12-21. ESM Interrupt Enable Set Register 4 (ESMIESR4) Field Descriptions

Bit	Field	Value	Description
63-32	INTENSET		Set interrupt enable. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: Interrupt is disabled. Write: Leaves the bit and the corresponding clear bit in the ESMIECR4 register unchanged.
		1	Read: Interrupt is enabled. Write: Enables interrupt and sets the corresponding clear bit in the ESMIECR4 register.

12.4.20 ESM Interrupt Enable Clear Register 4 (ESMIECR4)

This register is dedicated for Group1.

Figure 12-30. ESM Interrupt Enable Clear Register 4 (ESMIECR4) [address = FFFF F54Ch]

31		16
	INTENCLR[63:48]	
	R/WP-0	

15		0
	INTENCLR[47:32]	
	R/WP-0	

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

Table 12-22. ESM Interrupt Enable Clear Register 4 (ESMIECR4) Field Descriptions

Bit	Field	Value	Description
63-32	INTENCLR		Clear interrupt enable. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: Interrupt is disabled. Write: Leaves the bit and the corresponding set bit in the ESMIESR4 register unchanged.
		1	Read: Interrupt is enabled. Write: Disables interrupt and clears the corresponding set bit in the ESMIESR4 register.

12.4.21 ESM Interrupt Level Set Register 4 (ESMILSR4)

This register is dedicated for Group1.

Figure 12-31. ESM Interrupt Level Set Register 4 (ESMILSR4) [address = FFFF F550h]

31	16
INTLVLSET[63:48]	
R/WP-0	

15	0
INTLVLSET[47:32]	
R/WP-0	

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

Table 12-23. ESM Interrupt Level Set Register 4 (ESMILSR4) Field Descriptions

Bit	Field	Value	Description
63-32	INTLVLSET		Set interrupt level. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: Read: Interrupt of channel x is mapped to low-level interrupt line. Write: Leaves the bit and the corresponding clear bit in the ESMILCR4 register unchanged.
		1	Read: Interrupt of channel x is mapped to high-level interrupt line. Write: Maps interrupt of channel x to high-level interrupt line and sets the corresponding clear bit in the ESMILCR4 register.

12.4.22 ESM Interrupt Level Clear Register 4 (ESMILCR4)

This register is dedicated for Group1.

Figure 12-32. ESM Interrupt Level Clear Register 4 (ESMILCR4) [address = FFFF F554h]

31	16
INTLVLCLR[63:48]	
R/WP-0	

15	0
INTLVLCLR[47:32]	
R/WP-0	

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

Table 12-24. ESM Interrupt Level Clear Register 4 (ESMILCR4) Field Descriptions

Bit	Field	Value	Description
63-32	INTLVLCLR		Clear interrupt level. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: Interrupt of channel x is mapped to low-level interrupt line. Write: Leaves the bit and the corresponding set bit in the ESMILSR4 register unchanged.
		1	Read: Interrupt of channel x is mapped to high-level interrupt line. Write: Maps interrupt of channel x to low-level interrupt line and clears the corresponding set bit in the ESMILSR4 register.

12.4.23 ESM Status Register 4 (ESMSR4)

This register is dedicated for Group1.

**Figure 12-33. ESM Status Register 4 (ESMSR4)
[address = FFFF F558h]**

31	16
ESF[63:48]	
R/W1CP-X/0	

15	0
ESF[47:32]	
R/W1CP-X/0	

LEGEND: R/W = Read/Write; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset/$\overline{\text{PORRST}}$; *X* = Value unchanged

Table 12-25. ESM Status Register 4 (ESMSR4) Field Descriptions

Bit	Field	Value	Description
63-32	ESF		Error Status Flag. Provides status information on a pending error. **Read in User and Privileged mode. Write in Privileged mode only.**
		0	Read: No error occurred; no interrupt is pending. Write: Leaves the bit unchanged.
		1	Read: Error occurred; interrupt is pending. Write: Clears the bit. **Note:** After $\overline{\text{RST}}$, if one of these flags are set and the corresponding interrupt are enabled, the interrupt service routine will be called.

Real-Time Interrupt (RTI) Module

This chapter describes the functionality of the real-time interrupt (RTI) module. The RTI is designed as an operating system timer to support a real time operating system (RTOS).

NOTE: This chapter describes a superset implementation of the RTI module that includes features and functionality related to DMA, and Timbase control. These features are dependent on the device-specific feature content. Consult your device-specific datasheet to determine the applicability of these features to your device being used.

Topic **Page**

13.1 Overview 458
13.2 Module Operation 459
13.3 RTI Control Registers 469

13.1 Overview

The real-time interrupt (RTI) module provides timer functionality for operating systems and for benchmarking code. The RTI module can incorporate several counters that define the timebases needed for scheduling in the operating system.

The timers also allow you to benchmark certain areas of code by reading the values of the counters at the beginning and the end of the desired code range and calculating the difference between the values.

13.1.1 Features

The RTI module has the following features:

- Two independent 64 bit counter blocks
- Four configurable compares for generating operating system ticks or DMA requests. Each event can be driven by either counter block 0 or counter block 1.
- Fast enabling/disabling of events
- Two time stamp (capture) functions for system or peripheral interrupts, one for each counter block
- Digital windowed watchdog

13.1.2 Industry Standard Compliance Statement

This module is specifically designed to fulfill the requirements for OSEK (**O**ffene **S**ysteme und deren **S**chnittstellen für die **E**lektronik im **K**raftfahrzeug, or Open Systems and the Corresponding Interfaces for Automotive Electronics) as well as OSEK/time-compliant operating systems, but is not limited to it.

13.2 Module Operation

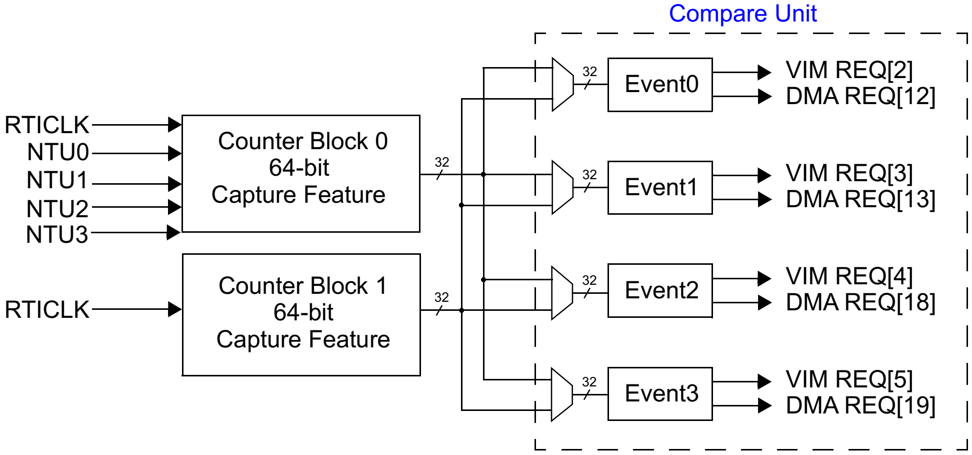
Figure 13-1 illustrates the high level block diagram of the RTI module.

The RTI module has two independent counter blocks for generating different timebases: counter block 0 and counter block 1.

A compare unit compares the counters with programmable values and generates four independent interrupt or DMA requests on compare matches. Each of the compare registers can be programmed to be compared to either counter block 0 or counter block 1.

The following sections describe the individual functions in more detail.

Figure 13-1. RTI Block Diagram



13.2.1 Counter Operation

Each counter block consists of the following (see Figure 13-2):

- One 32-bit prescale counter (RTIUC0 or RTIUC1)
- One 32-bit free running counter (RTIFRC0 or RTIFRC1)

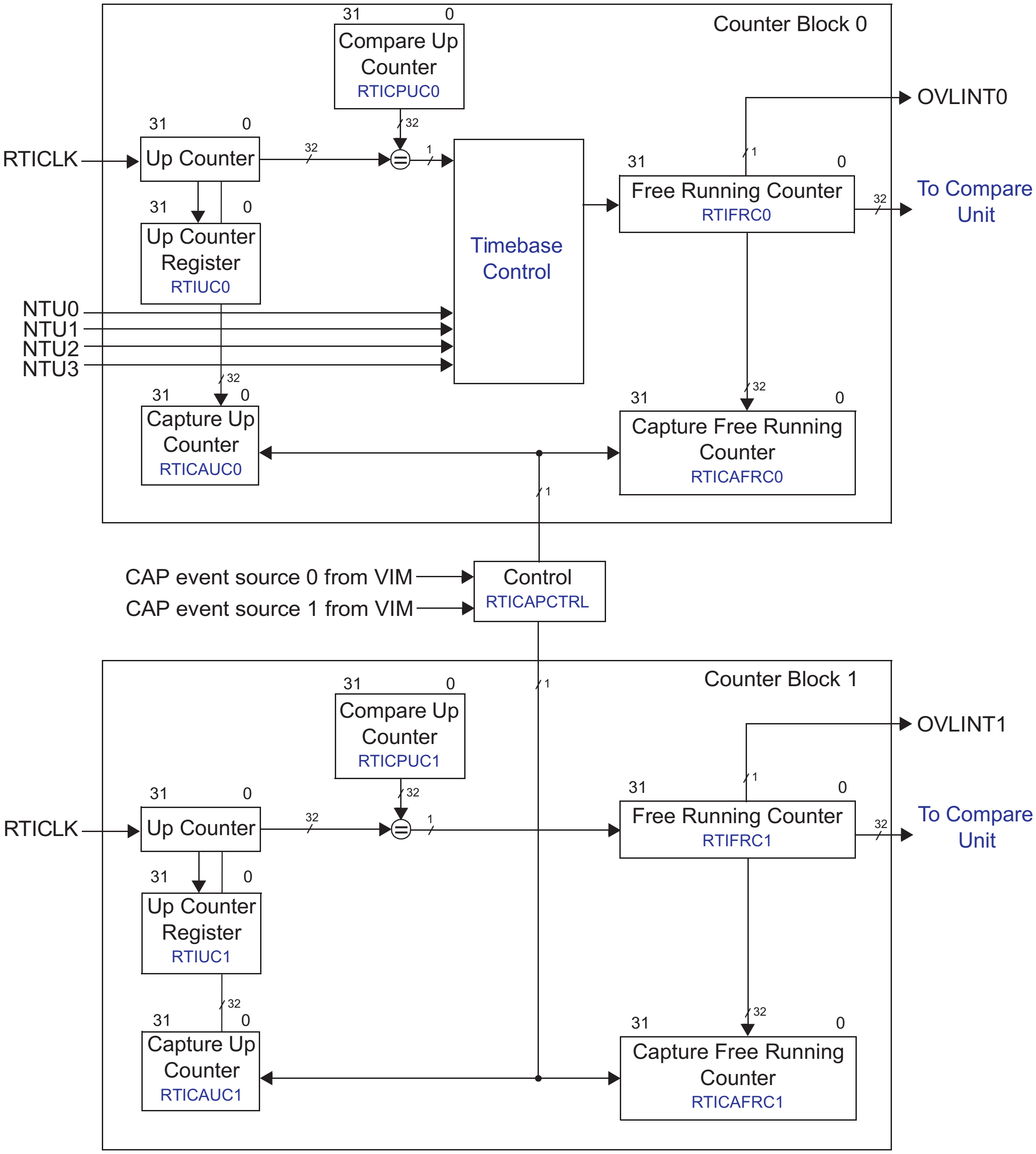
The RTIUC0/1 is driven by the RTICLK and counts up until the compare value in the compare up counter register (RTICPUC0 or RTICPUC1) is reached. When the compare matches, RTIFRC0/1 is incremented and RTIUC0/1 is reset to 0. If RTIFRC0/1 overflows, an interrupt is generated to the vectored interrupt manager (VIM). The overflow interrupt is not intended to generate the timebase for the operating system. See Section 13.2.2 for the timebase generation. The up counter together with the compare up counter value prescale the RTI clock. The resulting formula for the frequency of the free running counter (RTIFRC0/1) is:

$$f_{RTIFRCx} = \begin{cases} \dfrac{f_{RTICLK}}{RTICPUCx + 1} & \text{when RTICPUCx} \neq 0 \\ \dfrac{f_{RTICLK}}{2^{32}+1} & \text{when RTICPUCx = 0} \end{cases} \quad (23)$$

NOTE: Setting RTICPUCx equal to zero is not recommended. Doing so will hold the Up Counter at zero for two RTICLK cycles after it overflows from 0xFFFFFFFF to zero.

The counter values can be determined by reading the respective counter registers or by generating a hardware event which captures the counter value into the respective capture register. Both functions are described in the following sections.

Figure 13-2. Counter Block Diagram



13.2.1.1 Counter and Capture Read Consistency

Portions of the device internal databus are 32-bits wide. If the application wants to read the 64-bit counters or the 64-bit capture values, a certain order of 32-bit read operations needs to be followed. This is to prevent one counter incrementing in between the two separate read operations to both counters.

Reading the Counters

The free running counter (RTIFRCx) must be read first. This priority will ensure that in the cycle when the CPU reads RTIFRCx, the up counter value is stored in its counter register (RTIUCx). The second read has to access the up counter register (RTIUCx), which then holds the value which corresponds to the number of RTICLK cycles that have elapsed at the time reading the free running counter register (RTIFRCx).

NOTE: The up counters are implemented as shadow registers. Reading RTIUCx without having read RTIFRCx first will return always the same value. RTIUCx will only be updated when RTIFRCx is read.

Reading the Capture Values

The free running counter capture register (RTICAFRCx) must be read first. This priority will ensure that in the cycle when the CPU reads RTICAFRCx, the up counter value is stored in its counter register (RTICAUCx). The second read has to access the up counter register (RTICAUCx), which then holds the value captured at the time when reading the capture free running counter register (RTICAFRCx).

NOTE: The capture up counter registers are implemented as shadow registers. Reading RTICAUCx without having read RTICAFRCx first will return always the same value. RTICAUCx will only be updated when RTICAFRCx is read.

13.2.1.2 Capture Feature

Both counter blocks also provide a capture feature on external events. Two capture sources can trigger the capture event. The source triggering the block is configurable (RTICAPCTRL). The sources originate from the Vectored Interrupt Manager (VIM) and allow the generation of capture events when a peripheral modules has generated an interrupt. Any of the peripheral interrupts can be selected as the capture event in the VIM.

When an event is detected, RTIUCx and RTIFRCx are stored in the capture up counter (RTICAUCx) and capture free running counter (RTICAFRCx) registers. The read order of the captured values must be the same as the read order of the actual counters (see Section 13.2.1.1).

13.2.2 Interrupt/DMA Requests

There are four compare registers (RTICOMPy) to generate interrupt requests to the VIM or DMA requests to the DMA controller. The interrupts can be used to generate different timebases for the operating system. Each of the compare registers can be configured to be compared to either RTIFRC0 or RTIFRC1. When the counter value matches the compare value, an interrupt is generated. To allow periodic interrupts, a certain value can be added to the compare value in RTICOMPy automatically. This value is stored in the update compare register (RTIUDCPy) and will be added after a compare is matched. The period of the generated interrupt/DMA request can be calculated with:

$$t_{COMPx} = t_{RTICLK} \text{ x } (RTICPUCy + 1) \text{ x } RTIUDCPy$$

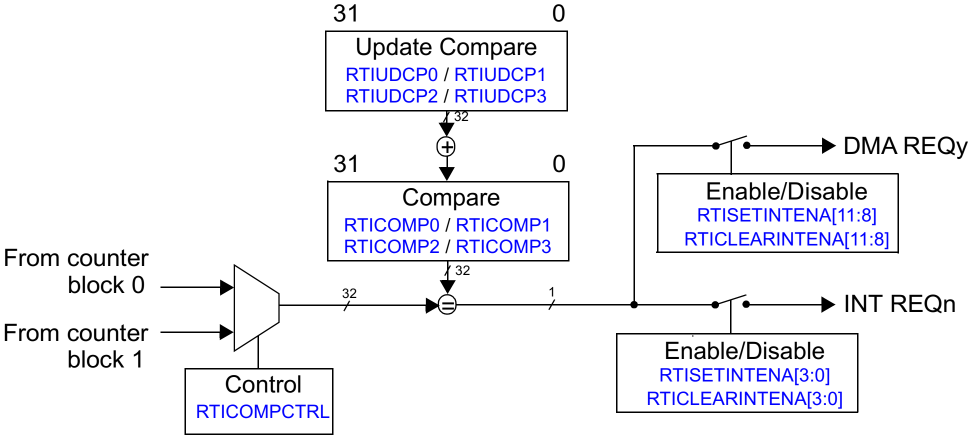
if RTICPUCy ≠ 0,

$$t_{COMPx} = t_{RTICLK} \text{ x } (2^{32}+1) \text{ x } RTIUDCPy$$

if RTIUDCPy = 0,

$$t_{COMPx} = t_{RTICLK} \text{ x } (RTICPUCy + 1) \text{ x } 2^{32} \qquad (24)$$

Figure 13-3. Compare Unit Block Diagram (shows only 1 of 4 blocks for simplification)



Another interrupt that can be generated is the overflow interrupt (OVLINTx) in case the RTIFRCx counter overflows.

The interrupts/DMA requests can be enabled in the RTISETINTENA register and disabled in the RTICLEARINTENA register. The RTIINTFLAG register shows the pending interrupts.

13.2.3 RTI Clocking

The counter blocks are clocked with RTICLK (for definition see Section 2.4.2).

A clock supervision for the NTUx clocking scheme is implemented to avoid missing operating system ticks.

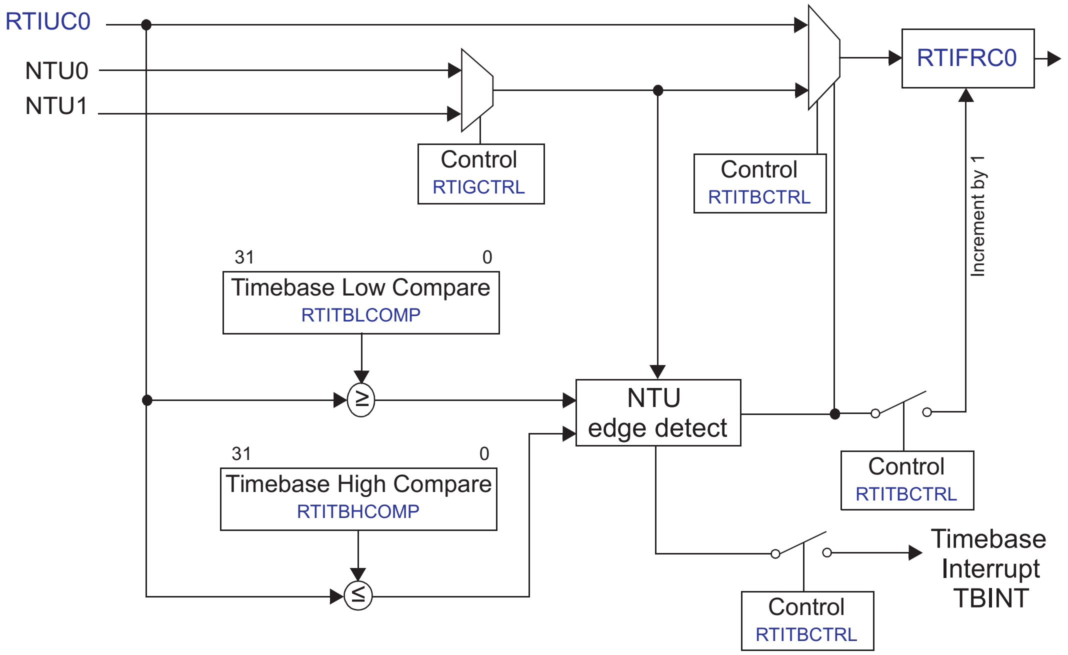
13.2.4 Synchronizing Timer Events to Network Time (NTU)

For applications which are participating on a time-triggered communication bus, it is often beneficial to synchronize the application or operating system to the network time. The RTI provides a feature to increment Free Running Counter 0 (RTIFRC0) by a periodic clock provided by the communication module. In this case two different clocks can be chosen.

The application has control over which clock (RTICLK, NTU0, NTU1) should be used for clocking RTIFRC0. If NTUx is used, a clock supervision circuit allows to monitor this clock and provides a fallback solution, should the clock be non-functional (missing). A too fast running NTUx cannot be detected.

RTIUC0 is utilized to monitor the NTUx signal. A detection window can be programmed in which a valid NTU clock pulse needs to occur. If no pulse is detected, the RTI automatically switches back to clock the Free Running Counter 0 with RTIUC0. In order to avoid a big jitter in the operating ticks, in case a switch back to RTIUC0 happens, RTICPUC0 should be set to a value so the clock frequency RTIUC0 outputs is approximately the same as the NTUx frequency.

Figure 13-4. Timebase Control



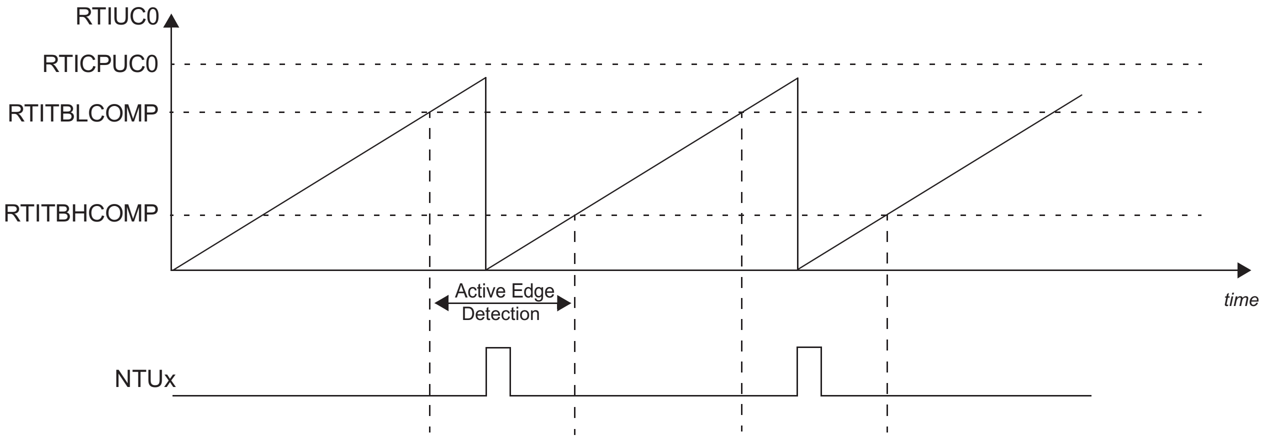
13.2.4.1 Detecting Clock Edges

To detect clock edges on the NTUx signal, the timebase low compare has to be set lower or equal than the value stored in the RTICPUC0 register and the timebase high compare has to be set higher than 0 and lower than the timebase low compare value. This effectively opens a window in which an edge of the NTUx signal is expected (see Figure 13-5). Outside this window, no edges will be detected. If no edge will occur inside the detection window, the multiplexer is switched to internal timebase. The application can select to generate a timebase interrupt (TBINT) and if the INC bit is set, also will automatically increment RTIFRC0 by one to compensate for the missed clock cycle of NTUx. If an edge occurs inside the window, RTIUC0 will be reset to synchronize the two timebases.

In order to make the edge detection work properly, the value in RTICPUC0 needs to be adapted so that RTIUC0 has a similar period as NTUx.

NOTE: To ensure the NTUx signal is properly detected, the NTUx period must be at least twice as long as the RTICLK period.

Figure 13-5. Clock Detection Scheme



13.2.4.2 Switching from Internal Source to External Source

If the application switches from an internal source to an external source, the two signals must be synchronized (see Figure 13-6). The synchronization will occur when the TBEXT bit is set. RTIUC0 will be reset and the edge detection circuit will be active for one (RTICPUC0 + RTITBHCOMP) period or until an edge is detected. If there is no pulse during this period, the source will be reset from an external clock source to an internal clock source. If an edge is detected, the windowed edge detection behavior will take place. Setting the TBEXT bit will not increment free running counter 0.

NOTE: If an external timebase is used, then the software must ensure that timebase low compare and timebase high compare are programmed to a valid state before switching TBEXT to an external source. This state is necessary to allow the timebase control circuit to operate correctly. The following condition must be met:

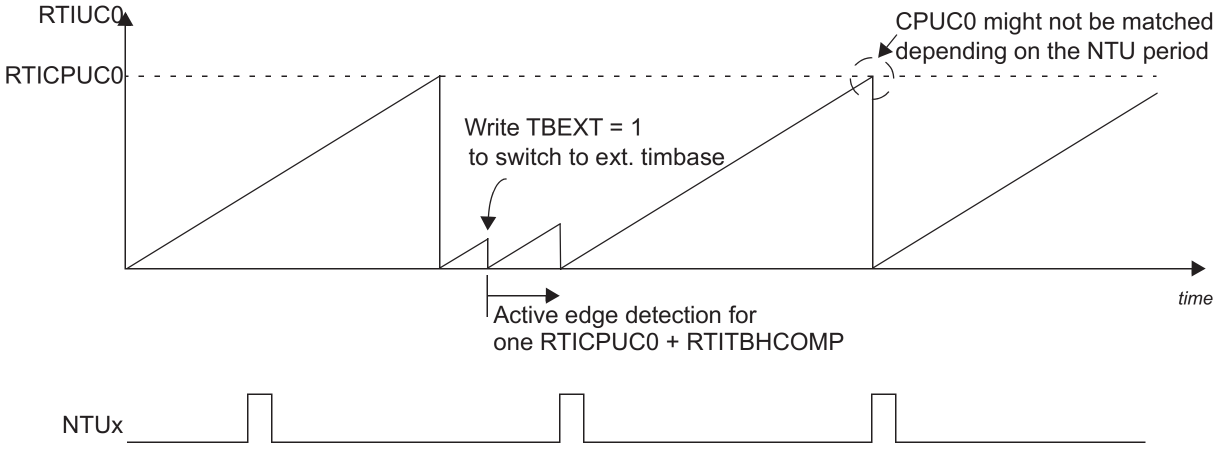
- RTITBHCOMP < RTITBLCOMP + RTICPUC0

RTITBHCOMP must be lower than RTICPUC0 because RTIUC0 will be reset if RTICPUC0 is reached. RTITBHCOMP will represent the number of RTICLK cycles from RTICPUC0 until the circuit switches to the internal timebase when no NTU edge is detected.

If an external timebase is used, RTIGCTRL[0] must be set to 1 (enable RTIUC0) to ensure that the timebase control circuit does not wait indefinitely for an incoming signal.

Figure 13-6 shows a timing example for the synchronization phase when the TBEXT bit is set.

Figure 13-6. Switch to NTUx

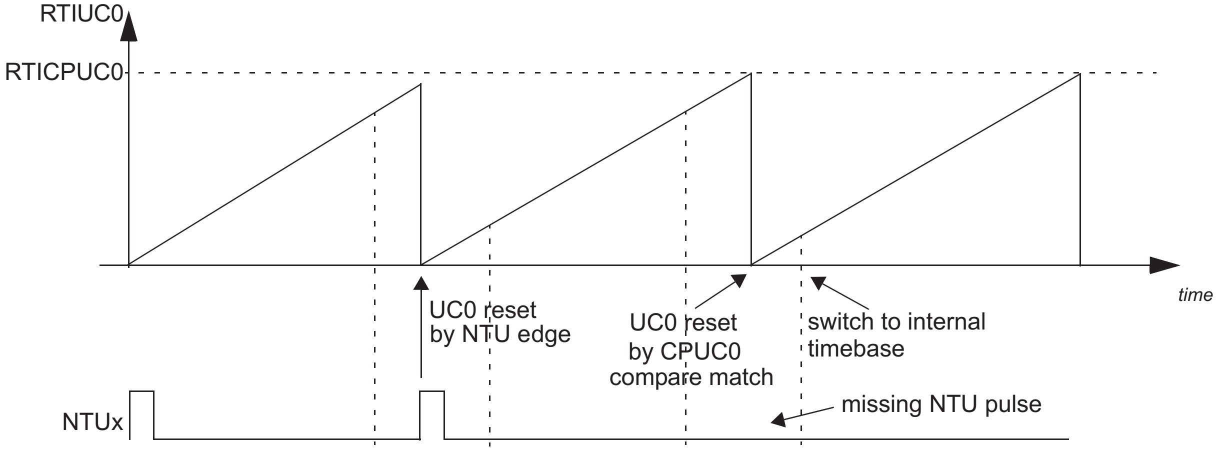

RTIUC0
RTICPUC0
CPUC0 might not be matched depending on the NTU period
Write TBEXT = 1 to switch to ext. timbase
Active edge detection for one RTICPUC0 + RTITBHCOMP
time
NTUx

13.2.4.3 Switching from External Source to Internal Source

When the edge detection is active (TBEXT = 1) and no clock edge of NTUx is detected inside the programmed detection window, the RTI will automatically switch the timebase to RTIUC0. Figure 13-7 shows a timing example for a missing NTU signal. In the case where the INC bit is set, RTIFRC0 will automatically be incremented by one to compensate for the missed NTU pulse.

Setting TBEXT = 0 will also switch the clock source for RTIFRC0 to RTIUC0.

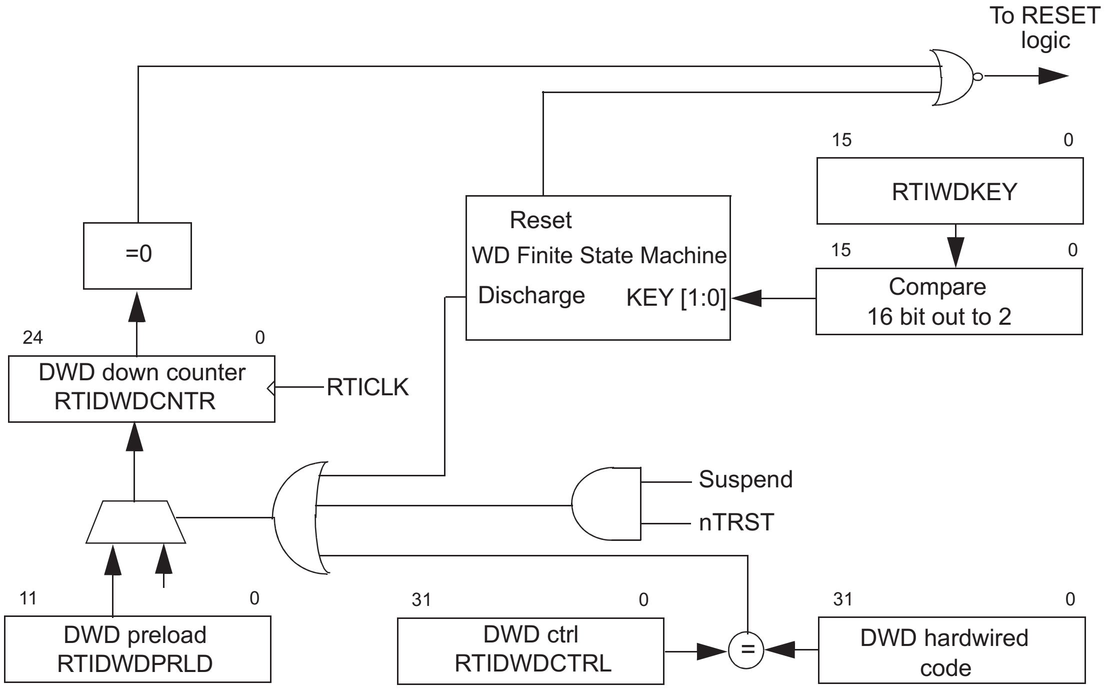
Figure 13-7. Missing NTUx Signal Example



13.2.5 Digital Watchdog (DWD)

The digital watchdog (DWD) is an optional safety diagnostic which can detect a runaway CPU and generate either a reset or NMI (non-maskable interrupt) response. It generates resets or NMIs after a programmable period, or if no correct key sequence was written to the RTIWDKEY register. Figure 13-8 illustrates the DWD.

Figure 13-8. Digital Watchdog



13.2.5.1 Digital Watchdog (DWD)

The DWD is disabled by default. If it should be used, it must be enabled by writing a 32-bit value to the RTIDWDCTRL register.

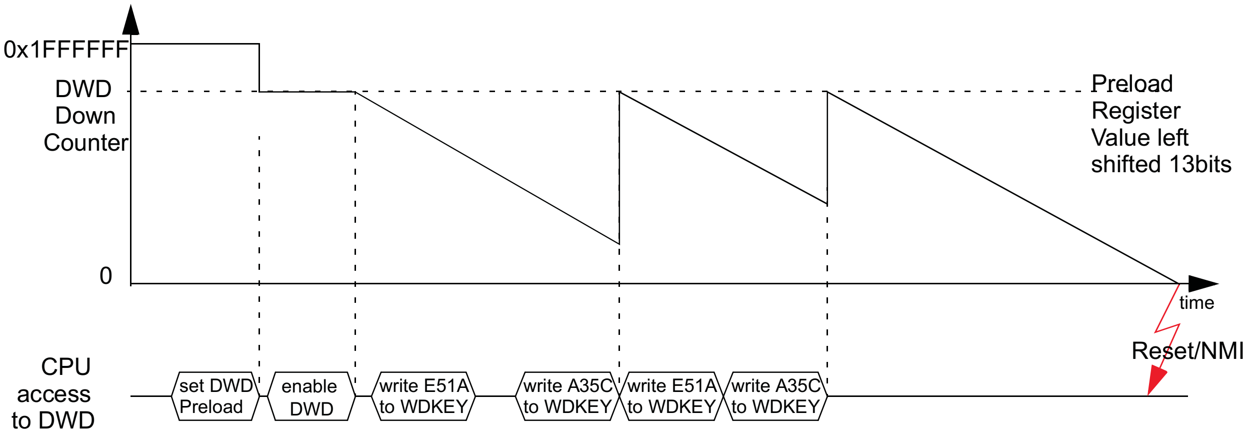
NOTE: Once the DWD is enabled, it cannot be disabled except by system reset or power on reset.

If the correct key sequence is written to the RTIWDKEY register (0xE51A followed by 0xA35C), the 25-bit DWD down counter is reloaded with the left justified 12-bit preload value stored in RTIDWDPRLD. If an incorrect value is written, a watchdog reset or NMI will occur immediately. A reset or NMI will also be generated when the DWD down counter is decremented to 0.

While the device is in suspend mode (halting debug mode), the DWD down counter keeps the value it had when entering suspend mode.

The DWD down counter will be decremented with the RTICLK frequency.

Figure 13-9. DWD Operation



The expiration time of the DWD down counter can be determined with the following equation:

texp = (DWDPRLD + 1) × 2^{13}/RTICLK

where

DWDPRLD = 0...4095

NOTE: Care should be taken to ensure that the CPU write to the watchdog register is made allowing time for the write to propagate to the RTI.

13.2.5.2 Digital Windowed Watchdog (DWWD)

In addition to the time-out boundary configurable via the digital watchdog discussed in Section 13.2.5.1, for enhanced safety metrics it is desirable to check for a watchdog "pet" within a time window rather than using a single time threshold. This is enabled by the digital windowed watchdog (DWWD) feature.

- Functional Behavior

The DWWD opens a configurable time window in which the watchdog must be serviced. Any attempt to service the watchdog outside this time window, or a failure to service the watchdog in this time window, will cause the watchdog to generate either a reset or a NMI to the CPU. This is controlled by configuring the RTIWWDRXNCTRL register. As with the DWD, the DWWD is disabled after power on reset. When the DWWD is configured to generate a non-maskable interrupt on a window violation, the watchdog counter continues to count down. The NMI handler needs to clear the watchdog violation status flag(s) and then service the watchdog by writing the correct sequence in the watchdog key register. This service will cause the watchdog counter to get reloaded from the preload value and start counting down. If the NMI handler does not service the watchdog in time, it could count down all the way to zero and wrap around. If the NMI Handler does not service the watchdog in time, the NMI gets generated continuously, each time the counter counts to '0'.

The DWWD uses the Digital Watchdog (DWD) preload register (RTIDWDPRLD) setting to define the end-time of the window. The start-time of the window is defined by a window size configuration register(RTIWWDSIZECTRL).

The default window size is set to 100%, which corresponds to the DWD functionality of a time-out-only watchdog. The window size can be selected (through register RTIWWDSIZECTRL) from among 100%, 50%, 25%, 12.5%, 6.25% and 3.125% as shown in Figure 13-10. The window with the respective size will be opened before the end of the DWD expiration. The user has to serve the watchdog in the window. Otherwise, a reset or NMI will generate. Figure 13-11 shows an DWWD operation example (25% window).

- Configuration of DWWD

The DWWD preload value (same as DWD preload) can only be configured when the DWWD counter is disabled. The window size and watchdog reaction to a violation can be configured even after the watchdog has been enabled. Any changes to the window size and watchdog reaction configurations will only take effect after the next servicing of the DWWD. This feature can be utilized to dynamically set windows of different sizes based on task execution time, adding a program sequence element to the diagnostic which can improve fault coverage.

Figure 13-10. Digital Windowed Watchdog Timing Example

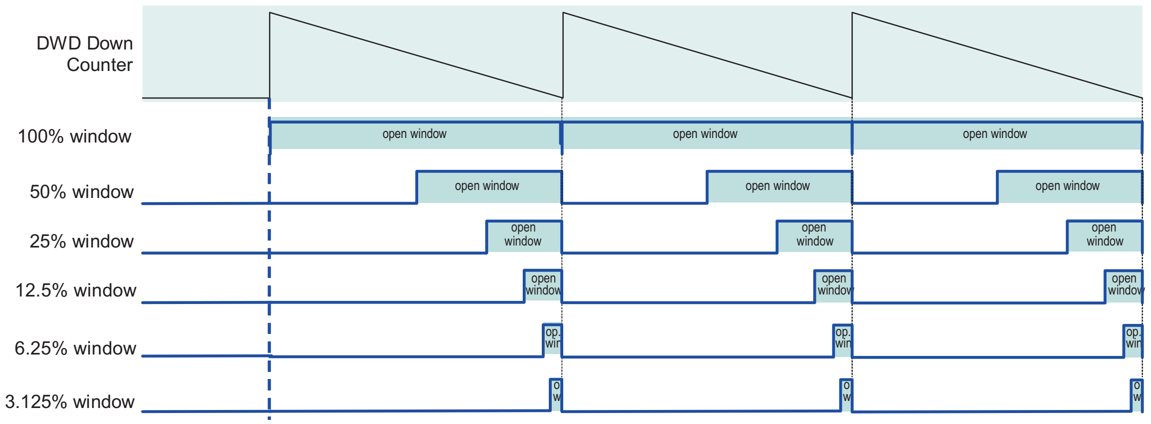
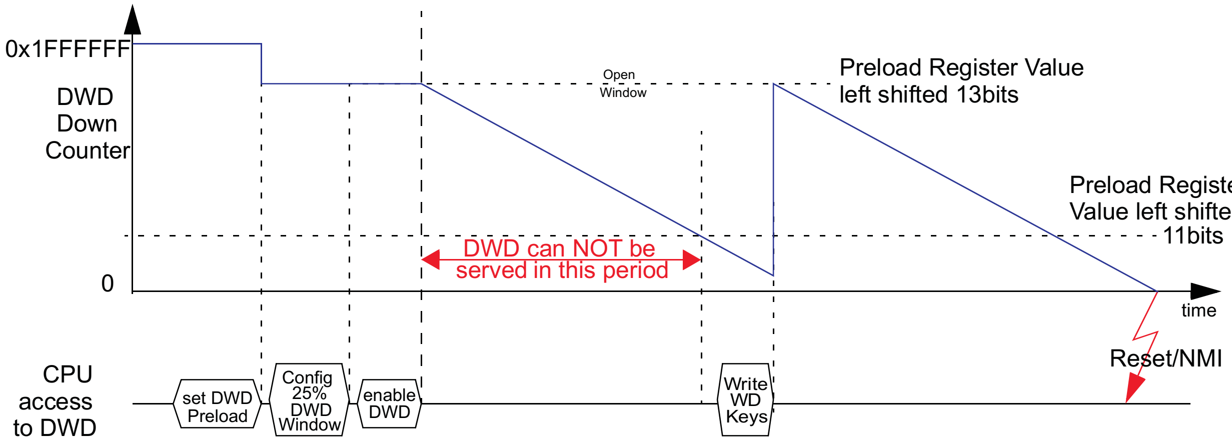


Figure 13-11. Digital Windowed Watchdog Operation Example (25% Window)



13.2.6 Low Power Modes

Low power modes allow the trade off of the current used during low power versus functionality and fast wakeup response. All low power modes have the following characteristics:

- CPU and system clocks are disabled.
- Flash banks and pump are in sleep mode.
- All peripheral modules are in low power modes and the clocks are disabled (exceptions to this may occur and would be documented in the specific device data sheet).

Flexibility in enabling and disabling clocks allows for many different low-power modes (see Section 2.4.3).

The operation of the RTI Module is guaranteed in Run, Doze and Snooze modes. In Sleep mode, all clocks will be switched off and the RTI will not work.

In Doze and Snooze modes, the RTI is active and is able to wake up the device with compare, timebase and overflow interrupts. The compare interrupts can be used to periodically wake up the device. The overflow interrupt can be used to notify the operating system that a counter overflow has occurred. Capturing events generated by the Vectored Interrupt Module (VIM) is also possible since, in both of these low power modes, the peripheral modules are able to generate interrupts that can trigger capture events. Capturing events while in Sleep mode is not supported as the clock to the RTI is not active.

When the device is put into low power mode, the peripheral which is generating the external clock NTU is no longer active, and the timebase control circuitry has to switch to an internal clocking scheme when it detects a missing clock on NTU. The timebase interrupt will wake up the device and the application software has to adapt the periodic interrupt generation to the internal clock source.

DMA transfers will be disabled, and DMA requests will not be generated after device wakeup since the DMA controller will be powered down.

NOTE: RTICLK in Doze Mode

In the special case of Doze Mode with PLL off, RTICLK might have a different period than with PLL enabled since RTICLK will be derived from the oscillator output. It has to be ensured that the VCLK to RTICLK ratio is at least 3:1.

13.2.7 Halting Debug Mode Behaviour

Once the system enters halting debug mode, the behavior of the RTI depends on the COS (continue on suspend) bit. If the bit is cleared and halting debug mode is active, all counters will stop operation. If the bit is set to one, all counters will be clocked normally and the RTI will work like in normal mode. However, if the external timebase (NTU) is used and the system is in halting debug mode, the timebase control circuit will switch to internal timebase once it detects the missing NTU signal of the suspended communication controller. This will be signaled with an TBINT interrupt so that software can resynchronize after the device exits halting debug mode.

13.3 RTI Control Registers

Table 13-1 provides a summary of the registers. The registers support 8-bit, 16-bit, and 32-bit writes. The offset is relative to the associated peripheral select. See the following sections for detailed descriptions of the registers. The base address for the control registers is FFFF FC00h. The address locations not listed are reserved.

Table 13-1. RTI Registers

Offset	Acronym	Register Description	Section
00h	RTIGCTRL	RTI Global Control Register	Section 13.3.1
04h	RTITBCTRL	RTI Timebase Control Register	Section 13.3.2
08h	RTICAPCTRL	RTI Capture Control Register	Section 13.3.3
0Ch	RTICOMPCTRL	RTI Compare Control Register	Section 13.3.4
10h	RTIFRC0	RTI Free Running Counter 0 Register	Section 13.3.5
14h	RTIUC0	RTI Up Counter 0 Register	Section 13.3.6
18h	RTICPUC0	RTI Compare Up Counter 0 Register	Section 13.3.7
20h	RTICAFRC0	RTI Capture Free Running Counter 0 Register	Section 13.3.8
24h	RTICAUC0	RTI Capture Up Counter 0 Register	Section 13.3.9
30h	RTIFRC1	RTI Free Running Counter 1 Register	Section 13.3.10
34h	RTIUC1	RTI Up Counter 1 Register	Section 13.3.11
38h	RTICPUC1	RTI Compare Up Counter 1 Register	Section 13.3.12
40h	RTICAFRC1	RTI Capture Free Running Counter 1 Register	Section 13.3.13
44h	RTICAUC1	RTI Capture Up Counter 1 Register	Section 13.3.14
50h	RTICOMP0	RTI Compare 0 Register	Section 13.3.15
54h	RTIUDCP0	RTI Update Compare 0 Register	Section 13.3.16
58h	RTICOMP1	RTI Compare 1 Register	Section 13.3.17
5Ch	RTIUDCP1	RTI Update Compare 1 Register	Section 13.3.18
60h	RTICOMP2	RTI Compare 2 Register	Section 13.3.19
64h	RTIUDCP2	RTI Update Compare 2 Register	Section 13.3.20
68h	RTICOMP3	RTI Compare 3 Register	Section 13.3.21
6Ch	RTIUDCP3	RTI Update Compare 3 Register	Section 13.3.22
70h	RTITBLCOMP	RTI Timebase Low Compare Register	Section 13.3.23
74h	RTITBHCOMP	RTI Timebase High Compare Register	Section 13.3.24
80h	RTISETINTENA	RTI Set Interrupt Enable Register	Section 13.3.25
84h	RTICLEARINTENA	RTI Clear Interrupt Enable Register	Section 13.3.26
88h	RTIINTFLAG	RTI Interrupt Flag Register	Section 13.3.27
90h	RTIDWDCTRL	Digital Watchdog Control Register	Section 13.3.28
94h	RTIDWDPRLD	Digital Watchdog Preload Register	Section 13.3.29
98h	RTIWDSTATUS	Watchdog Status Register	Section 13.3.30
9Ch	RTIWDKEY	RTI Watchdog Key Register	Section 13.3.31
A0h	RTIDWDCNTR	RTI Digital Watchdog Down Counter Register	Section 13.3.32
A4h	RTIWWDRXNCTRL	Digital Windowed Watchdog Reaction Control Register	Section 13.3.33
A8h	RTIWWDSIZECTRL	Digital Windowed Watchdog Window Size Control Register	Section 13.3.34
ACh	RTIINTCLRENABLE	RTI Compare Interrupt Clear Enable Register	Section 13.3.35
B0h	RTICOMP0CLR	RTI Compare 0 Clear Register	Section 13.3.36
B4h	RTICOMP1CLR	RTI Compare 1 Clear Register	Section 13.3.37
B8h	RTICOMP2CLR	RTI Compare 2 Clear Register	Section 13.3.38
BCh	RTICOMP3CLR	RTI Compare 3 Clear Register	Section 13.3.39

NOTE: Writes to Reserved registers may clear the pending RTI interrupt.

13.3.1 RTI Global Control Register (RTIGCTRL)

The global control register starts/stops the counters and selects the signal compared with the timebase control circuit. This register is shown in Figure 13-12 and described in Table 13-2.

Figure 13-12. RTI Global Control Register (RTIGCTRL) [offset = 00]

31	20	19	16
Reserved		NTUSEL	
R-0		R/WP-0	

15	14	2	1	0
COS	Reserved		CNT1EN	CNT0EN
R/WP-0	R-0		R/WP-0	R/WP-0

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

Table 13-2. RTI Global Control Register (RTIGCTRL) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	NTUSEL		Select NTU signal. These bits determine which NTU input signal is used as external timebase
		0h	NTU0
		5h	NTU1
		Ah	NTU2
		Fh	NTU3
		All other values	Tied to 0
15	COS		Continue on suspend. This bit determines if both counters are stopped when the device goes into halting debug mode or if they continue counting.
		0	Counters are stopped while in halting debug mode.
		1	Counters are running while in halting debug mode.
14-2	Reserved	0	Reads return 0. Writes have no effect.
1	CNT1EN		Counter 1 enable. This bit starts and stops counter block 1 (RTIUC1 and RTIFRC1).
		0	Counter block 1 is stopped.
		1	Counter block 1 is running.
0	CNT0EN		Counter 0 enable. This bit starts and stops counter block 0 (RTIUC0 and RTIFRC0).
		0	Counter block 0 is stopped.
		1	Counter block 0 is running.

NOTE: If the application uses the timebase circuit for synchronization between the communications controller and the operating system and the device enters halting debug mode, the synchronization may be lost depending on the COS setting in the RTI module and the halting debug mode behavior of the communications controller.

13.3.2 RTI Timebase Control Register (RTITBCTRL)

The timebase control register selects if the free running counter 0 is incremented by RTICLK or NTU. This register is shown in Figure 13-13 and described in Table 13-3.

Figure 13-13. RTI Timebase Control Register (RTITBCTRL) [offset = 04h]

31	8
Reserved	
R-0	

7	2	1	0
Reserved		INC	TBEXT
R-0		R/WP-0	R/WP-0

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

Table 13-3. RTI Timebase Control Register (RTITBCTRL) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reads return 0. Writes have no effect.
1	INC		Increment free running counter 0. This bit determines whether the free running counter 0 (RTIFRC0) is automatically incremented if a failing clock on the NTU signal is detected.
		0	RTIFRC0 will not be incremented on a failing external clock.
		1	RTIFRC0 will be incremented on a failing external clock.
0	TBEXT		Timebase external. This bit selects whether the free running counter 0 (RTIFRC0) is clocked by the internal up counter 0 (RTIUC0) or from the external signal NTU. Setting the TBEXT bit from 0 to 1 will not increment RTIFRC0, since RTIUC0 is reset. When the timebase supervisor circuit detects a missing clock edge, then the TBEXT bit is reset. Only the software can select whether the external signal should be used.
		0	RTIUC0 clocks RTIFRC0.
		1	NTU clocks RTIFRC0.

13.3.3 RTI Capture Control Register (RTICAPCTRL)

The capture control register controls the capture source for the counters. This register is shown in Figure 13-14 and described in Table 13-4.

Figure 13-14. RTI Capture Control Register (RTICAPCTRL) [offset = 08h]

31 … 8
Reserved
R-0

7 … 2	1	0
Reserved	CAPCNTR1	CAPCNTR0
R-0	R/WP-0	R/WP-0

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

Table 13-4. RTI Capture Control Register (RTICAPCTRL) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reads return 0. Writes have no effect.
1	CAPCNTR1		Capture counter 1. This bit determines which external interrupt source triggers a capture event of RTIUC1 and RTIFRC1.
		0	Capture of RTIUC1/ RTIFRC1 is triggered by capture event source 0.
		1	Capture of RTIUC1/ RTIFRC1 is triggered by capture event source 1.
0	CAPCNTR0		Capture counter 0. This bit determines which external interrupt source triggers a capture event of RTIUC0 and RTIFRC0.
		0	Capture of RTIUC0/ RTIFRC0 is triggered by capture event source 0.
		1	Capture of RTIUC0/ RTIFRC0 is triggered by capture event source 1.

13.3.4 RTI Compare Control Register (RTICOMPCTRL)

The compare control register controls the source for the compare registers. This register is shown in Figure 13-15 and described in Table 13-5.

Figure 13-15. RTI Compare Control Register (RTICOMPCTRL) [offset = 0Ch]

31–16
Reserved
R-0

15–13	12	11–9	8
Reserved	COMPSEL3	Reserved	COMPSEL2
R-0	R/WP-0	R-0	R/WP-0

7–5	4	3–1	0
Reserved	COMPSEL1	Reserved	COMPSEL0
R-0	R/WP-0	R-0	R/WP-0

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

Table 13-5. RTI Compare Control Register (RTICOMPCTRL) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reads return 0. Writes have no effect.
12	COMPSEL3		Compare select 3. This bit determines the counter with which the compare value held in compare register 3 (RTICOMP3) is compared.
		0	Value will be compared with RTIFRC0.
		1	Value will be compared with RTIFRC1.
11-9	Reserved	0	Reads return 0. Writes have no effect.
8	COMPSEL2		Compare select 2. This bit determines the counter with which the compare value held in compare register 2 (RTICOMP2) is compared.
		0	Value will be compared with RTIFRC0.
		1	Value will be compared with RTIFRC1.
7-5	Reserved	0	Reads return 0. Writes have no effect.
4	COMPSEL1		Compare select 1. This bit determines the counter with which the compare value held in compare register 1 (RTICOMP1) is compared.
		0	Value will be compared with RTIFRC0.
		1	Value will be compared with RTIFRC1.
3-1	Reserved	0	Reads return 0. Writes have no effect.
0	COMPSEL0		Compare select 0. This bit determines the counter with which the compare value held in compare register 0 (RTICOMP0) is compared.
		0	Value will be compared with RTIFRC0.
		1	Value will be compared with RTIFRC1.

13.3.5 RTI Free Running Counter 0 Register (RTIFRC0)

The free running counter 0 register holds the current value of free running counter 0. This register is shown in Figure 13-16 and described in Table 13-6.

Figure 13-16. RTI Free Running Counter 0 Register (RTIFRC0) [offset = 10h]

31	16
FRC0	
R/WP-0	

15	0
FRC0	
R/WP-0	

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

Table 13-6. RTI Free Running Counter 0 Register (RTIFRC0) Field Descriptions

Bit	Field	Value	Description
31-0	FRC0	0-FFFF FFFFh	Free running counter 0. This registers holds the current value of the free running counter 0. A read of this counter returns the current value of the counter. The counter can be preset by writing (in privileged mode only) to this register. The counter increments then from this written value upwards. **Note: If counters must be preset, they must be disabled in the RTIGCTRL register to ensure consistency between RTIUC0 and RTIFRC0.**

13.3.6 RTI Up Counter 0 Register (RTIUC0)

The up counter 0 register holds the current value of prescale counter. This register is shown in Figure 13-17 and described in Table 13-7.

Figure 13-17. RTI Up Counter 0 Register (RTIUC0) [offset = 14h]

31	16
UC0	
R/WP-0	

15	0
UC0	
R/WP-0	

LLEGEND: R/W = Read/Write; WP = Write in privileged mode only; -*n* = value after reset

Table 13-7. RTI Up Counter 0 Register (RTIUC0) Field Descriptions

Bit	Field	Value	Description
31-0	UC0	0-FFFF FFFFh	Up counter 0. This register holds the current value of the up counter 0 and prescales the RTI clock. It will be only updated by a previous read of free running counter 0 (RTIFRC0). This method of updating effectively gives a 64-bit read of both counters, without having the problem of a counter being updated between two consecutive reads on up counter 0 (RTIUC0) and free running counter 0 (RTIFRC0). A read of this counter returns the value of the counter at the time RTIFRC0 was read. A write to this counter presets it with a value. The counter then increments from this written value upwards. Note: If counters must be preset, they must be disabled in the RTIGCTRL register to ensure consistency between RTIUC0 and RTIFRC0. **Note: If the preset value is bigger than the compare value stored in register RTICPUC0, then it can take a long time until a compare matches, since RTIUC0 has to count up until it overflows.**

13.3.7 RTI Compare Up Counter 0 Register (RTICPUC0)

The compare up counter 0 register holds the value to be compared with prescale counter 0 (RTIUC0). This register is shown in Figure 13-18 and described in Table 13-8.

Figure 13-18. RTI Compare Up Counter 0 Register (RTICPUC0) [offset = 18h]

31 ... 16
CPUC0
R/WP-0

15 ... 0
CPUC0
R/WP-0

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

Table 13-8. RTI Compare Up Counter 0 Register (RTICPUC0) Field Descriptions

Bit	Field	Value	Description
31-0	CPUC0	0-FFFF FFFFh	Compare up counter 0. This register holds the value that is compared with the up counter 0. When the compare shows a match, the free running counter 0 (RTIFRC0) is incremented. RTIUC0 is set to 0 when the counter value matches the RTICPUC0 value. The value set in this register prescales the RTI clock. If CPUC0 = 0, then f_{FRC0} = RTICLK/(2^{32}+1) (Setting CPUC0 equal to 0 is not recommended. Doing so will hold the Up Counter at 0 for 2 RTICLK cycles after it overflows from FFFF FFFFh to 0.) If CPUC0 ≠ 0, then f_{FRC0} = RTICLK/(RTICPUC0+1) A read of this register returns the current compare value. A write to this register: • If TBEXT = 0, the compare value is updated. • If TBEXT = 1, the compare value is unchanged.

13.3.8 RTI Capture Free Running Counter 0 Register (RTICAFRC0)

The capture free running counter 0 register holds the free running counter 0 on external events. This register is shown in Figure 13-19 and described in Table 13-9.

Figure 13-19. RTI Capture Free Running Counter 0 Register (RTICAFRC0) [offset = 20h]

31 ... 16
CAFRC0
R-0

15 ... 0
CAFRC0
R-0

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

Table 13-9. RTI Capture Free Running Counter 0 Register (RTICAFRC0) Field Descriptions

Bit	Field	Value	Description
31-0	CAFRC0	0-FFFF FFFFh	Capture free running counter 0. This register captures the current value of the free running counter 0 (RTIFRC0) when an event occurs, controlled by the external capture control block. A read of this register returns the value of RTIFRC0 on a capture event.

13.3.9 RTI Capture Up Counter 0 Register (RTICAUC0)

The capture up counter 0 register holds the current value of prescale counter 0 on external events. This register is shown in Figure 13-20 and described in Table 13-10.

Figure 13-20. RTI Capture Up Counter 0 Register (RTICAUC0) [offset = 24h]

31	16
CAUC0	
R-0	

15	0
CAUC0	
R-0	

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

Table 13-10. RTI Capture Up Counter 0 Register (RTICAUC0) Field Descriptions

Bit	Field	Value	Description
31-0	CAUC0	0-FFFF FFFFh	Capture up counter 0. This register captures the current value of the up counter 0 (RTIUC0) when an event occurs, controlled by the external capture control block. **Note: The read sequence must be the same as with RTIUC0 and RTIFRC0. Therefore, the RTICAFRC0 register must be read before the RTICAUC0 register is read. This sequence ensures that the value of the RTICAUC0 register is the corresponding value to the RTICAFRC0 register, even if another capture event happens in between the two reads.** A read of this register returns the value of RTIUC0 on a capture event.

13.3.10 RTI Free Running Counter 1 Register (RTIFRC1)

The free running counter 1 register holds the current value of the free running counter 1. This register is shown in Figure 13-21 and described in Table 13-11.

Figure 13-21. RTI Free Running Counter 1 Register (RTIFRC1) [offset = 30h]

31	16
FRC1	
R/WP-0	

15	0
FRC1	
R/WP-0	

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

Table 13-11. RTI Free Running Counter 1 Register (RTIFRC1) Field Descriptions

Bit	Field	Value	Description
31-0	FRC1	0-FFFF FFFFh	Free running counter 1. This register holds the current value of the free running counter 1 and will be updated continuously. A read of this register returns the current value of the counter. A write to this register presets the counter. The counter increments then from this written value upwards. **Note: If counters must be preset, they must be disabled in the RTIGCTRL register to ensure consistency between RTIUC1 and RTIFRC1.**

13.3.11 RTI Up Counter 1 Register (RTIUC1)

The up counter 1 register holds the current value of the prescale counter 1. This register is shown in Figure 13-22 and described in Table 13-12.

Figure 13-22. RTI Up Counter 1 Register (RTIUC1) [offset = 34h]

31	16
UC1	
R/WP-0	

15	0
UC1	
R/WP-0	

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

Table 13-12. RTI Up Counter 1 Register (RTIUC1) Field Descriptions

Bit	Field	Value	Description
31-0	UC1	0-FFFF FFFFh	Up counter 1. This register holds the current value of the up counter 1 and prescales the RTI clock. It will be only updated by a previous read of free running counter 1 (RTIFRC1). This method of updating effectively gives a 64-bit read of both counters, without having the problem of a counter being updated between two consecutive reads on RTIUC1 and RTIFRC1. A read of this register will return the value of the counter when the RTIFRC1 was read. A write to this register presets the counter. The counter then increments from this written value upwards. **Note: If counters must be preset, they must be disabled in the RTIGCTRL register to ensure consistency between RTIUC1 and RTIFRC1.** **Note: If the preset value is bigger than the compare value stored in register RTICPUC1, then it can take a long time until a compare matches, since RTIUC1 has to count up until it overflows.**

13.3.12 RTI Compare Up Counter 1 Register (RTICPUC1)

The compare up counter 1 register holds the value compared with prescale counter 1. This register is shown in Figure 13-23 and described in Table 13-13.

Figure 13-23. RTI Compare Up Counter 1 Register (RTICPUC1) [offset = 38h]

31 … 16
CPUC1
R/WP-0

15 … 0
CPUC1
R/WP-0

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

Table 13-13. RTI Compare Up Counter 1 Register (RTICPUC1) Field Descriptions

Bit	Field	Value	Description
31-0	CPUC1	0-FFFF FFFFh	Compare up counter 1. This register holds the compare value, which is compared with the up counter 1. When the compare matches, the free running counter 1 (RTIFRC1) is incremented. The up counter is cleared to 0 when the counter value matches the CPUC1 value. The value set in this prescales the RTI clock according to the following formula: If CPUC1 = 0, then f_{FRC1} = RTICLK/(2^{32}+1) (Setting CPUC1 equal to 0 is not recommended. Doing so will hold the Up Counter at 0 for 2 RTICLK cycles after it overflows from FFFF FFFFh to 0.) If CPUC1 ≠ 0, then f_{FRC1} = RTICLK/(RTICPUC1+1) A read of this register returns the current compare value. A write to this register updates the compare value.

13.3.13 RTI Capture Free Running Counter 1 Register (RTICAFRC1)

The capture free running counter 1 register holds the current value of free running counter 1 on external events. This register is shown in Figure 13-24 and described in Table 13-14.

Figure 13-24. RTI Capture Free Running Counter 1 Register (RTICAFRC1) [offset = 40h]

31	16
CAFRC1	
R-0	

15	0
CAFRC1	
R-0	

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

Table 13-14. RTI Capture Free Running Counter 1 Register (RTICAFRC1) Field Descriptions

Bit	Field	Value	Description
31-0	CAFRC1	0-FFFF FFFFh	Capture free running counter 1. This register captures the current value of the free running counter 1 (RTIFRC1) when an event occurs, controlled by the external capture control block. A read of this register returns the value of RTIFRC1 on a capture event.

13.3.14 RTI Capture Up Counter 1 Register (RTICAUC1)

The capture up counter 1 register holds the current value of prescale counter 1 on external events. This register is shown in Figure 13-25 and described in Table 13-15.

Figure 13-25. RTI Capture Up Counter 1 Register (RTICAUC1) [offset = 44h]

31	16
CAUC1	
R-0	

15	0
CAUC1	
R-0	

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

Table 13-15. RTI Capture Up Counter 1 Register (RTICAUC1) Field Descriptions

Bit	Field	Value	Description
31-0	CAUC1	0-FFFF FFFFh	Capture up counter 1. This register captures the current value of the up counter 1 (RTIUC1) when an event occurs, controlled by the external capture control block. **Note: The RTICAFRC1 register must be read before the RTICAUC1 register is read. This sequence ensures that the value of the RTICAUC1 register is the corresponding value to the RTICAFRC1 register, even if another capture event happens in between the two reads.** A read of this register returns the value of RTIUC1 on a capture event.

13.3.15 RTI Compare 0 Register (RTICOMP0)

The compare 0 register holds the value to be compared with the counters. This register is shown in Figure 13-26 and described in Table 13-16.

Figure 13-26. RTI Compare 0 Register (RTICOMP0) [offset = 50h]

31	16
COMP0	
R/WP-0	

15	0
COMP0	
R/WP-0	

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

Table 13-16. RTI Compare 0 Register (RTICOMP0) Field Descriptions

Bit	Field	Value	Description
31-0	COMP0	0-FFFF FFFFh	Compare 0. This registers holds a value that is compared with the counter selected in the compare control logic. If RTIFRC0 or RTIFRC1, depending on the counter selected, matches the compare value, an interrupt is flagged. With this register it is also possible to initiate a DMA request. A read of this register will return the current compare value. A write to this register (in privileged mode only) will update the compare register with a new compare value.

13.3.16 RTI Update Compare 0 Register (RTIUDCP0)

The update compare 0 register holds the value to be added to the compare register 0 value on a compare match. This register is shown in Figure 13-27 and described in Table 13-17.

Figure 13-27. RTI Update Compare 0 Register (RTIUDCP0) [offset = 54h]

31	16
UDCP0	
R/WP-0	

15	0
UDCP0	
R/WP-0	

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

Table 13-17. RTI Update Compare 0 Register (RTIUDCP0) Field Descriptions

Bit	Field	Value	Description
31-0	UDCP0	0-FFFF FFFFh	Update compare 0. This register holds a value that is added to the value in the compare 0 (RTICOMP0) register each time a compare matches. This function allows periodic interrupts to be generated without software intervention. A read of this register will return the value to be added to the RTICOMP0 register on the next compare match. A write to this register will provide a new update value.

13.3.17 RTI Compare 1 Register (RTICOMP1)

The compare 1 register holds the value to be compared to the counters. This register is shown in Figure 13-28 and described in Table 13-18.

Figure 13-28. RTI Compare 1 Register (RTICOMP1) [offset = 58h]

31	16
COMP1	
R/WP-0	

15	0
COMP1	
R/WP-0	

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

Table 13-18. RTI Compare 1 Register (RTICOMP1) Field Descriptions

Bit	Field	Value	Description
31-0	COMP1	0-FFFF FFFFh	Compare 1. This register holds a value that is compared with the counter selected in the compare control logic. If RTIFRC0 or RTIFRC1, depending on the counter selected, matches this compare value, an interrupt is flagged. With this register, it is possible to initiate a DMA request. A read of this register will return the current compare value. A write to this register will update the compare register with a new compare value.

13.3.18 RTI Update Compare 1 Register (RTIUDCP1)

The update compare 1 register holds the value to be added to the compare register 1 value on a compare match. This register is shown in Figure 13-29 and described in Table 13-19.

Figure 13-29. RTI Update Compare 1 Register (RTIUDCP1) [offset = 5Ch]

31	16
UDCP1	
R/WP-0	

15	0
UDCP1	
R/WP-0	

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

Table 13-19. RTI Update Compare 1 Register (RTIUDCP1) Field Descriptions

Bit	Field	Value	Description
31-0	UDCP1	0-FFFF FFFFh	Update compare 1. This register holds a value that is added to the value in the RTICOMP1 register each time a compare matches. This process allows periodic interrupts to be generated without software intervention. A read of this register will return the value to be added to the RTICOMP1 register on the next compare match. A write to this register will provide a new update value.

13.3.19 RTI Compare 2 Register (RTICOMP2)

The compare 2 register holds the value to be compared to the counters. This register is shown in Figure 13-30 and described in Table 13-20.

Figure 13-30. RTI Compare 2 Register (RTICOMP2) [offset = 60h]

31	16
COMP2	
R/WP-0	

15	0
COMP2	
R/WP-0	

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

Table 13-20. RTI Compare 2 Register (RTICOMP2) Field Descriptions

Bit	Field	Value	Description
31-0	COMP2	0-FFFF FFFFh	Compare 2. This register holds a value that is compared with the counter selected in the compare control logic. If RTIFRC0 or RTIFRC1, depending on the counter selected, matches this compare value, an interrupt is flagged. With this register, it is possible to initiate a DMA request. A read of this register will return the current compare value. A write to this register (in privileged mode only) will provide a new compare value.

13.3.20 RTI Update Compare 2 Register (RTIUDCP2)

The update compare 2 register holds the value to be added to the compare register 2 value on a compare match. This register is shown in Figure 13-31 and described in Table 13-21.

Figure 13-31. RTI Update Compare 2 Register (RTIUDCP2) [offset = 64h]

31	16
UDCP2	
R/WP-0	

15	0
UDCP2	
R/WP-0	

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

Table 13-21. RTI Update Compare 2 Register (RTIUDCP2) Field Descriptions

Bit	Field	Value	Description
31-0	UDCP2	0-FFFF FFFFh	Update compare 2. This register holds a value that is added to the value in the RTICOMP2 register each time a compare matches. This process makes it possible to generate periodic interrupts without software intervention. A read of this register will return the value to be added to the RTICOMP2 register on the next compare match. A write to this register will provide a new update value.

13.3.21 RTI Compare 3 Register (RTICOMP3)

The compare 3 register holds the value to be compared to the counters. This register is shown in Figure 13-32 and described in Table 13-22.

Figure 13-32. RTI Compare 3 Register (RTICOMP3) [offset = 68h]

31	16
COMP3	
R/WP-0	

15	0
COMP3	
R/WP-0	

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

Table 13-22. RTI Compare 3 Register (RTICOMP3) Field Descriptions

Bit	Field	Value	Description
31-0	COMP3	0-FFFF FFFFh	Compare 3. This register holds a value that is compared with the counter selected in the compare control logic. If RTIFRC0 or RTIFRC1, depending on the counter selected, matches this compare value, an interrupt is flagged. With this register, it is possible to initiate a DMA request. A read of this register will return the current compare value. A write to this register will provide a new compare value.

13.3.22 RTI Update Compare 3 Register (RTIUDCP3)

The update compare 3 register holds the value to be added to the compare register 3 value on a compare match. This register is shown in Figure 13-33 and described in Table 13-23.

Figure 13-33. RTI Update Compare 3 Register (RTIUDCP3) [offset = 6Ch]

31	16
UDCP3	
R/WP-0	

15	0
UDCP3	
R/WP-0	

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

Table 13-23. RTI Update Compare 3 Register (RTIUDCP3) Field Descriptions

Bit	Field	Value	Description
31-0	UDCP3	0-FFFF FFFFh	Update compare 3. This register holds a value that is added to the value in the RTICOMP3 register each time a compare matches. This process makes it possible to generate periodic interrupts without software intervention. A read of this register will return the value to be added to the RTICOMP3 register on the next compare match. A write to this register will provide a new update value.

13.3.23 RTI Timebase Low Compare Register (RTITBLCOMP)

The timebase low compare register holds the value to activate the edge detection circuit. This register is shown in Figure 13-34 and described in Table 13-24.

Figure 13-34. RTI Timebase Low Compare Register (RTITBLCOMP) [offset = 70h]

31	16
TBLCOMP	
R/WP-0	

15	0
TBLCOMP	
R/WP-0	

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

Table 13-24. RTI Timebase Low Compare Register (RTITBLCOMP) Field Descriptions

Bit	Field	Value	Description
31-0	TBLCOMP	0-FFFF FFFFh	Timebase low compare value. This value determines when the edge detection circuit starts monitoring the NTU signal. It will be compared with RTIUC0. A read of this register will return the current compare value. A write to this register has the following effects: If TBEXT = 0: The compare value is updated. If TBEXT = 1: The compare value is not changed.

13.3.24 RTI Timebase High Compare Register (RTITBHCOMP)

The timebase high compare register holds the value to deactivate the edge detection circuit. This register is shown in Figure 13-35 and described in Table 13-25.

Figure 13-35. RTI Timebase High Compare Register (RTITBHCOMP) [offset = 74h]

31	16
TBHCOMP	
R/WP-0	

15	0
TBHCOMP	
R/WP-0	

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

Table 13-25. RTI Timebase High Compare Register (RTITBHCOMP) Field Descriptions

Bit	Field	Value	Description
31-0	TBHCOMP	0-FFFF FFFFh	Timebase high compare value. This value determines when the edge detection circuit will stop monitoring the NTU signal. It will be compared with RTIUC0. RTITBHCOMP must be less than RTICPUC0 because RTIUC0 will be reset when RTICPUC0 is reached. Example: The NTU edge detection circuit should be active ± 10 RTICLK cycles around RTICPUC0. • RTICPUC0 = 0050h • RTITBLCOMP = 0046h • RTITBHCOMP = 0009h A read of this register will return the current compare value. A write to this register has the following effects: If TBEXT = 0: The compare value is updated. If TBEXT = 1: The compare value is not changed.

13.3.25 RTI Set Interrupt Enable Register (RTISETINTENA)

This register prevents the necessity of a read-modify-write operation if a particular interrupt should be enabled. This register is shown in Figure 13-36 and described in Table 13-26.

Figure 13-36. RTI Set Interrupt Control Register (RTISETINTENA) [offset = 80h]

31-24
Reserved
R-0

23-19	18	17	16
Reserved	SETOVL1INT	SETOVL0INT	SETTBINT
R-0	R/WP-0	R/WP-0	R/WP-0

15-12	11	10	9	8
Reserved	SETDMA3	SETDMA2	SETDMA1	SETDMA0
R-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

7-4	3	2	1	0
Reserved	SETINT3	SETINT2	SETINT1	SETINT0
R-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

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

Table 13-26. RTI Set Interrupt Control Register (RTISETINTENA) Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reads return 0. Writes have no effect.
18	SETOVL1INT		Set free running counter 1 overflow interrupt.
		0	*Read:* Interrupt is disabled. *Write:* Corresponding bit is unchanged.
		1	*Read or Write:* Interrupt is enabled.
17	SETOVL0INT		Set free running counter 0 overflow interrupt.
		0	*Read:* Interrupt is disabled. *Write:* Corresponding bit is unchanged.
		1	*Read or Write:* Interrupt is enabled.
16	SETTBINT		Set timebase interrupt.
		0	*Read:* Interrupt is disabled. *Write:* Corresponding bit is unchanged.
		1	*Read or Write:* Interrupt is enabled.
15-12	Reserved	0	Reads return 0. Writes have no effect.
11	SETDMA3		Set compare DMA request 3.
		0	*Read:* DMA request is disabled. *Write:* DMA request is unchanged.
		1	*Read or Write:* DMA request is enabled.
10	SETDMA2		Set compare DMA request 2.
		0	*Read:* DMA request is disabled. *Write:* DMA request is unchanged.
		1	*Read or Write:* DMA request is enabled.
9	SETDMA1		Set compare DMA request 1.
		0	*Read:* DMA request is disabled. *Write:* DMA request is unchanged.
		1	*Read or Write:* DMA request is enabled.

Table 13-26. RTI Set Interrupt Control Register (RTISETINTENA) Field Descriptions (continued)

Bit	Field	Value	Description
8	SETDMA0		Set compare DMA request 0.
		0	*Read:* DMA request is disabled. *Write:* DMA request is unchanged.
		1	*Read or Write:* DMA request is enabled.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3	SETINT3		Set compare interrupt 3.
		0	*Read:* Interrupt is disabled. *Write:* Corresponding bit is unchanged.
		1	*Read or Write:* Interrupt is enabled.
2	SETINT2		Set compare interrupt 2.
		0	*Read:* Interrupt is disabled. *Write:* Corresponding bit is unchanged.
		1	*Read or Write:* Interrupt is enabled.
1	SETINT1		Set compare interrupt 1.
		0	*Read:* Interrupt is disabled. *Write:* Corresponding bit is unchanged.
		1	*Read or Write:* Interrupt is enabled.
0	SETINT0		Set compare interrupt 0.
		0	*Read:* Interrupt is disabled. *Write:* Corresponding bit is unchanged.
		1	*Read or Write:* Interrupt is enabled.

13.3.26 RTI Clear Interrupt Enable Register (RTICLEARINTENA)

This register prevents the necessity of a read-modify-write operation if a particular interrupt should be disabled. This register is shown in Figure 13-37 and described in Table 13-27.

Figure 13-37. RTI Clear Interrupt Control Register (RTICLEARINTENA) [offset = 84h]

31-24
Reserved
R-0

23-19	18	17	16
Reserved	CLEAROVL1INT	CLEAROVL0INT	CLEARTBINT
R-0	R/WP-0	R/WP-0	R/WP-0

15-12	11	10	9	8
Reserved	CLEARDMA3	CLEARDMA2	CLEARDMA1	CLEARDMA0
R-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

7-4	3	2	1	0
Reserved	CLEARINT3	CLEARINT2	CLEARINT1	CLEARINT0
R-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

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

Table 13-27. RTI Clear Interrupt Control Register (RTICLEARINTENA) Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reads return 0. Writes have no effect.
18	CLEAROVL1INT		Clear free running counter 1 overflow interrupt.
		0	*Read:* Interrupt is disabled. *Write:* Corresponding bit is unchanged.
		1	*Read:* Interrupt is enabled. *Write:* Interrupt is disabled.
17	CLEAROVL0INT		Clear free running counter 0 overflow interrupt.
		0	*Read:* Interrupt is disabled. *Write:* Corresponding bit is unchanged.
		1	*Read:* Interrupt is enabled. *Write:* Interrupt is disabled.
16	CLEARTBINT		Clear timebase interrupt.
		0	*Read:* Interrupt is disabled. *Write:* Corresponding bit is unchanged.
		1	*Read:* Interrupt is enabled. *Write:* Interrupt is disabled.
15-12	Reserved	0	Reads return 0. Writes have no effect.
11	CLEARDMA3		Clear compare DMA request 3.
		0	*Read:* DMA request is disabled. *Write:* Corresponding bit is unchanged.
		1	*Read:* DMA request is enabled. *Write:* DMA request is disabled.
10	CLEARDMA2		Clear compare DMA request 2.
		0	*Read:* DMA request is disabled. *Write:* Corresponding bit is unchanged.
		1	*Read:* DMA request is enabled. *Write:* DMA request is disabled.

Table 13-27. RTI Clear Interrupt Control Register (RTICLEARINTENA) Field Descriptions (continued)

Bit	Field	Value	Description
9	CLEARDMA1	0	Clear compare DMA request 1. *Read:* DMA request is disabled. *Write:* Corresponding bit is unchanged.
		1	*Read:* DMA request is enabled. *Write:* DMA request is disabled.
8	CLEARDMA0	0	Clear compare DMA request 0. *Read:* DMA request is disabled. *Write:* Corresponding bit is unchanged.
		1	*Read:* DMA request is enabled. *Write:* DMA request is disabled.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3	CLEARINT3	0	Clear compare interrupt 3. *Read:* Interrupt is disabled. *Write:* Corresponding bit is unchanged.
		1	*Read:* Interrupt is enabled. *Write:* Interrupt is disabled.
2	CLEARINT2	0	Clear compare interrupt 2. *Read:* Interrupt is disabled. *Write:* Corresponding bit is unchanged.
		1	*Read:* Interrupt is enabled. *Write:* Interrupt is disabled.
1	CLEARINT1	0	Clear compare interrupt 1. *Read:* Interrupt is disabled. *Write:* Corresponding bit is unchanged.
		1	*Read:* Interrupt is enabled. *Write:* Interrupt is disabled.
0	CLEARINT0	0	Clear compare interrupt 0. *Read:* Interrupt is disabled. *Write:* Corresponding bit is unchanged.
		1	*Read:* Interrupt is enabled. *Write:* Interrupt is disabled.

13.3.27 RTI Interrupt Flag Register (RTIINTFLAG)

The corresponding flags are set at every compare match of the RTIFRCx and RTICOMPx values, whether the interrupt is enabled or not. This register is shown in Figure 13-38 and described in Table 13-28.

Figure 13-38. RTI Interrupt Flag Register (RTIINTFLAG) [offset = 88h]

31	19	18	17	16
Reserved		OVL1INT	OVL0INT	TBINT
R-0		R/W1CP-0	R/W1CP-0	R/W1C P-0

15	4	3	2	1	0
Reserved		INT3	INT2	INT1	INT0
R-0		R/W1C P-0	R/W1C P-0	R/W1C P-0	R/W1C P-0

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset

Table 13-28. RTI Interrupt Flag Register (RTIINTFLAG) Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reads return 0. Writes have no effect.
18	OVL1INT	0	Free running counter 1 overflow interrupt flag. This bit determines if an interrupt is pending. *Read:* No interrupt is pending. *Write:* Bit is unchanged.
		1	*Read:* Interrupt is pending. *Write:* Bit is cleared to 0.
17	OVL0INT	0	Free running counter 0 overflow interrupt flag. This bit determines if an interrupt is pending. *Read:* No interrupt is pending. *Write:* Bit is unchanged.
		1	*Read:* Interrupt is pending. *Write:* Bit is cleared to 0.
16	TBINT	0	Timebase interrupt flag. This flag is set when the TBEXT bit is cleared by detection of a missing external clock edge. It will not be set by clearing TBEXT by software. It determines if an interrupt is pending. *Read:* No interrupt is pending. *Write:* Bit is unchanged.
		1	*Read:* Interrupt is pending. *Write:* Bit is cleared to 0.
15-4	Reserved	0	Reads return 0. Writes have no effect.
3	INT3	0	Interrupt flag 3. These bits determine if an interrupt due to a Compare 3 match is pending. *Read:* No interrupt is pending. *Write:* Bit is unchanged.
		1	*Read:* Interrupt is pending. *Write:* Bit is cleared to 0.
2	INT2	0	Interrupt flag 2. These bits determine if an interrupt due to a Compare 2 match is pending. *Read:* No interrupt is pending. *Write:* Bit is unchanged.
		1	*Read:* Interrupt is pending. *Write:* Bit is cleared to 0.
1	INT1	0	Interrupt flag 1. These bits determine if an interrupt due to a Compare 1 match is pending. *Read:* No interrupt is pending. *Write:* Bit is unchanged.
		1	*Read:* Interrupt is pending. *Write:* Bit is cleared to 0.

Table 13-28. RTI Interrupt Flag Register (RTIINTFLAG) Field Descriptions (continued)

Bit	Field	Value	Description
0	INT0		Interrupt flag 0. These bits determine if an interrupt due to a Compare 0 match is pending.
		0	*Read:* No interrupt is pending. *Write:* Bit is unchanged.
		1	*Read:* Interrupt is pending. *Write:* Bit is cleared to 0.

13.3.28 Digital Watchdog Control Register (RTIDWDCTRL)

The software has to write to the DWDCTRL field in order to enable the DWD, as described below. Once enabled, the watchdog can only be disabled by a system reset. The application cannot disable the watchdog. However should the RTICLK source be changed to a source that is unimplemented it will have the same effect as disabling the watchdog. This register is shown in Figure 13-38 and described in Table 13-28.

Figure 13-39. Digital Watchdog Control Register (RTIDWDCTRL) [offset = 90h]

31	16
DWDCTRL	
R/WP-5312h	
15	**0**
DWDCTRL	
R/WP-ACEDh	

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

Table 13-29. Digital Watchdog Control Register (RTIDWDCTRL) Field Descriptions

Bit	Field	Value	Description
31-0	DWDCTRL		Digital Watchdog Control.
		5312 ACEDh	*Read:* DWD counter is disabled. *Write:* State of DWD counter is unchanged (stays enabled or disabled).
		A985 59DAh	*Read:* DWD counter is enabled. *Write:* DWD counter is enabled.
		All other values	*Read:* DWD counter state is unchanged (enabled or disabled). *Write:* State of DWD counter is unchanged (stays enabled or disabled). **Note:** Once the enable value is written, all other future writes are blocked. In other words, once DWD is enabled, it can only be disabled by system reset or power on reset. However should the RTICLK source be changed to a source that is unimplemented it will have the same effect as disabling the watchdog.

13.3.29 Digital Watchdog Preload Register (RTIDWDPRLD)

This register sets the expiration time of the DWD. This register is shown in Figure 13-38 and described in Table 13-28.

Figure 13-40. Digital Watchdog Preload Register (RTIDWDPRLD) [offset = 94h]

31 … 16
Reserved
R-0

15 … 12	11 … 0
Reserved	DWDPRLD
R-0	R/WP-FFFh

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

Table 13-30. Digital Watchdog Preload Register (RTIDWDPRLD) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reads return 0 and writes have no effect.
11-0	DWDPRLD	0-FFFh	Digital Watchdog Preload Value. *Read:* The current preload value *Write:* Set the preload value. The DWD preload register can be configured only when the DWD is disabled. Therefore, the application can only configure the DWD preload register before it enables the DWD down counter. The expiration time of the DWD Down Counter can be determined with following equation: texp = (DWDPRLD+1) x 2^{13} / RTICLK1 where: DWDPRLD = 0...4095

13.3.30 Watchdog Status Register (RTIWDSTATUS)

This register records the status of the DWD. The values of the following status bits will not be affected by a soft reset. These bits are cleared by a power-on reset, or by a write of 1. These bits can be used for debug purposes. This register is shown in Figure 13-38 and described in Table 13-28.

Figure 13-41. Watchdog Status Register (RTIWDSTATUS) [offset = 98h]

31-8
Reserved
R-0

7-6	5	4	3	2	1	0
Reserved	DWWD ST	END TIME VIOL	START TIME VIOL	KEY ST	DWD ST	Reserved
R-0	R/W1CP-x	R/W1CP-x	R/W1CP-x	R/W1CP-x	R/W1CP-x	R-0

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset

Table 13-31. Watchdog Status Register (RTIWDSTATUS) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reads return 0. Writes have no effect.
5	DWWD ST	0	Windowed Watchdog Status *Read:* No time-window violation has occurred. *Write:* Leaves the current value unchanged.
		1	*Read:* Time-window violation has occurred. The watchdog has generated either a system reset or a non-maskable interrupt to the CPU in this case. *Write:* Bit is cleared to 0. This will also clear all other status flags in the RTIWDSTATUS register. Clearing of the status flags will deassert the non-maskable interrupt generated due to violation of the DWWD.
4	END TIME VIOL	0	Windowed Watchdog End Time Violation Status. This bit indicates whether the Watchdog counter expired. *Read:* No end-time window violation has occurred. *Write:* Leaves the current value unchanged.
		1	*Read*: End-time defined by the windowed watchdog configuration has been violated. *Write:* Bit is cleared to 0.
3	START TIME VIOL	0	Windowed Watchdog Start Time Violation Status. This bit indicates whether the key is written before the watchdog window opened up. *Read:* No start-time window violation has occurred. *Write:* Leaves the current value unchanged.
		1	*Read:* Start-time defined by the windowed watchdog configuration has been violated. *Write:* Bit is cleared to 0.
2	KEY ST	0	Watchdog key status. This bit indicates a reset or NMI generated by a wrong key or key sequence written to the RTIWDKEY register. *Read:* No wrong key or key-sequence written. *Write:* Bit is unchanged.
		1	*Read:* Wrong key or key-sequence written to RTIWDKEY register. *Write:* Bit is cleared to 0.
1	DWD ST	0	DWD status. This bit is equivalent to bit END TIME VIOL. *Read:* No reset or NMI was generated. *Write:* Bit is unchanged.
		1	*Read:* Reset or NMI was generated. *Write:* Bit is cleared to 0.
0	Reserved	0	Reads return 0. Writes have no effect.

13.3.31 RTI Watchdog Key Register (RTIWDKEY)

This register must be written with the correct written key values to serve the watchdog. This register is shown in Figure 13-42 and described in Table 13-32.

NOTE: It has to be taken into account that the write to the RTIWDKEY register takes 3 VCLK cycles.

Figure 13-42. RTI Watchdog Key Register (RTIDWDKEY) [offset = 9Ch]

31	16
Reserved	
R-0	

15	0
WDKEY	
R/WP-A35Ch	

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

Table 13-32. RTI Watchdog Key Register (RTIDWDKEY) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0 and writes have no effect.
15-0	WDKEY	0-FFFFh	Watchdog key. These bits provide the key sequence location. Reads returns the current WDKEY value. A write of E51Ah followed by A35Ch in two separate write operations defines the key sequence and reloads the DWD. Writing any other value causes a reset or NMI, as shown in Table 13-33. Writing any other value will cause the WDKEY to reset to A35Ch.

Table 13-33. Example of a WDKEY Sequence

Step	Value Written to WDKEY	Result
1	A35Ch	No action
2	A35Ch	No action
3	E51Ah	WDKEY is enabled for reset or NMI by next A35Ch.
4	E51Ah	WDKEY is enabled for reset or NMI by next A35Ch.
5	E51Ah	WDKEY is enabled for reset or NMI by next A35Ch.
6	A35Ch	Watchdog is reset.
7	A35Ch	No action
8	E51Ah	WDKEY is enabled for reset or NMI by next A35Ch.
9	A35Ch	Watchdog is reset.
10	E51Ah	WDKEY is enabled for reset or NMI by next A35Ch.
11	2345h	System reset or NMI; incorrect value written to WDKEY.

13.3.32 RTI Digital Watchdog Down Counter (RTIDWDCNTR)

This register provides the current value of the DWD down counter. This register is shown in Figure 13-43 and described in Table 13-34.

Figure 13-43. RTI Watchdog Down Counter Register (RTIDWDCNTR) [offset = A0h]

31 – 25	24 – 16
Reserved	DWDCNTR
R-0	R-1FFh

15 – 0
DWDCNTR
R-FFFFh

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

Table 13-34. RTI Watchdog Down Counter Register (RTIDWDCNTR) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0 and writes have no effect.
24-0	DWDCNTR	0-1FF FFFFh	DWD down counter. Reads return the current counter value.

13.3.33 Digital Windowed Watchdog Reaction Control (RTIWWDRXNCTRL)

This register selects the DWWD reaction if the watchdog is serviced outside the time window. This register is shown in Figure 13-44 and described in Table 13-35.

Figure 13-44. Digital Windowed Watchdog Reaction Control (RTIWWDRXNCTRL) [offset = A4h]

31 – 16
Reserved
R-0

15 – 4	3 – 0
Reserved	WWDRXN
R-0	R/WP-5h

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

Table 13-35. Digital Windowed Watchdog Reaction Control (RTIWWDRXNCTRL) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0 and writes have no effect.
3-0	WWDRXN		The DWWD reaction
		5h	The windowed watchdog will cause a reset if the watchdog is serviced outside the time window defined by the configuration, or if the watchdog is not serviced at all.
		Ah	The windowed watchdog will generate a non-maskable interrupt to the CPU if the watchdog is serviced outside the time window defined by the configuration, or if the watchdog is not serviced at all.
		All other values	The windowed watchdog will cause a reset if the watchdog is serviced outside the time window defined by the configuration, or if the watchdog is not serviced at all.
			Note: The DWWD reaction can be selected by the application even when the DWWD counter is already enabled. If a change to the WWDRXN is made before the watchdog service window is opened, then the change in the configuration takes effect immediately. If a change to the WWDRXN is made when the watchdog service window is already open, then the change in configuration takes effect only after the watchdog is serviced.

13.3.34 Digital Windowed Watchdog Window Size Control (RTIWWDSIZECTRL)

This register selects the DWWD window size. This register is shown in Figure 13-45 and described in Table 13-36.

Figure 13-45. Digital Windowed Watchdog Window Size Control (RTIWWDSIZECTRL) [offset = A8h]

31	16
WWDSIZE	
R/WP-0000	

15	0
WWDSIZE	
R/WP-0005h	

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

Table 13-36. Digital Windowed Watchdog Window Size Control (RTIWWDSIZECTRL) Field Descriptions

Bit	Field	Value	Description
31-0	WWDSIZE	0	The DWWD window size
		0000 0005h	100% (The functionality is the same as the standard time-out digital watchdog.)
		0000 0050h	50%
		0000 0500h	25%
		0000 5000h	12.5%
		0005 0000h	6.25%
		All other values	3.125%
			Note: The DWWD window size can be selected by the application even when the DWWD counter is already enabled. If a change to the WWDSIZE is made before the watchdog service window is opened, then the change in the configuration takes effect immediately. If a change to the WWDSIZE is made when the watchdog service window is already open, then the change in configuration takes effect only after the watchdog is serviced.

13.3.35 RTI Compare Interrupt Clear Enable Register (RTIINTCLRENABLE)

When the RTI compare event is configured to generate a DMA request or triggers (all triggered by RTI compare interrupt request flag) to other peripherals, it is often desirable to clear the RTI compare flag automatically so that the requests can be generated repeatedly without any CPU intervention. This register works with the RTI compare clear registers to enable an "auto-clear" of the compare interrupt enable bit after a compare equal event. This register is shown in Figure 13-46 and described in Table 13-37.

Figure 13-46. RTI Compare Interrupt Clear Enable Register (RTIINTCLRENABLE) [offset = ACh]

31 28	27 24	23 20	19 16
Reserved	INTCLRENABLE3	Reserved	INTCLRENABLE2
R-0	R/WP-5h	R-0	R/WP-5h

15 12	11 8	7 4	3 0
Reserved	INTCLRENABLE1	Reserved	INTCLRENABLE0
R-0	R/WP-5h	R-0	R/WP-5h

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

Table 13-37. RTI Compare Interrupt Clear Enable Register (RTIINTCLRENABLE) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reads return 0. Writes have no effect.
27-24	INTCLRENABLE3		Enables the auto-clear functionality on the compare 3 interrupt.
		5h	*Read:* Auto-clear for compare 3 interrupt is disabled. *Privileged Write:* Auto-clear for compare 3 interrupt becomes disabled.
		All other values	*Read:* Auto-clear for compare 3 interrupt is enabled. *Privileged Write:* Auto-clear for compare 3 interrupt becomes enabled.
23-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	INTCLRENABLE2		Enables the auto-clear functionality on the compare 2 interrupt.
		5h	*Read:* Auto-clear for compare 2interrupt is disabled. *Privileged Write:* Auto-clear for compare 2 interrupt becomes disabled.
		All other values	*Read:* Auto-clear for compare 2 interrupt is enabled. *Privileged Write:* Auto-clear for compare 2 interrupt becomes enabled.
15-12	Reserved	0	Reads return 0. Writes have no effect.
11-8	INTCLRENABLE1		Enables the auto-clear functionality on the compare 1 interrupt.
		5h	*Read:* Auto-clear for compare 1 interrupt is disabled. *Privileged Write:* Auto-clear for compare 1 interrupt becomes disabled.
		All other values	*Read:* Auto-clear for compare 1 interrupt is enabled. *Privileged Write:* Auto-clear for compare 1 interrupt becomes enabled.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	INTCLRENABLE0		Enables the auto-clear functionality on the compare 0 interrupt.
		5h	*Read:* Auto-clear for compare 0 interrupt is disabled. *Privileged Write:* Auto-clear for compare 0 interrupt becomes disabled.
		All other values	*Read:* Auto-clear for compare 0 interrupt is enabled. *Privileged Write:* Auto-clear for compare 0 interrupt becomes enabled.

13.3.36 RTI Compare 0 Clear Register (RTICMP0CLR)

This registers holds an initial value which is larger than the value in the RTI Compare 0 register Section 13.3.4. The user needs to choose the value such that the compare clear 0 event occurs before next compare 0 event. If the Free Running Counter matches the compare value, the compare 0 interrupt request flag is cleared and the value in the RTIUDCP0 register Section 13.3.16 is added to this register. This register is shown in Figure 13-47 and described in Table 13-38.

Figure 13-47. RTI Compare 0 Clear Register (RTICMP0CLR) [offset = B0h]

31	16
CMP0CLR	
R/WP-0	

15	0
CMP0CLR	
R/WP-0	

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

Table 13-38. RTI Compare 0 Clear Register (RTICMP0CLR) Field Descriptions

Bit	Field	Value	Description
31-0	CMP0CLR	0-FFFF FFFFh	Compare 0 clear. This registers holds a compare value. If the Free Running Counter matches the compare value, the compare 0 interrupt request flag is cleared and the value in the RTIUDCP0 register Section 13.3.16 is added to this register. Reads return the current compare clear value. A privileged write to this register updates the compare clear value.

13.3.37 RTI Compare 1 Clear Register (RTICMP1CLR)

This registers holds an initial value which is larger than the value in the RTI Compare 1 register Section 13.3.4. The user needs to choose the value such that the compare clear 1 event occurs before next compare 1 event. If the Free Running Counter matches the compare value, the compare 1 interrupt request flag is cleared and the value in the RTIUDCP1 register Section 13.3.18 is added to this register. This register is shown in Figure 13-48 and described in Table 13-39.

Figure 13-48. RTI Compare 1 Clear Register (RTICMP1CLR) [offset = B4h]

31	16
CMP1CLR	
R/WP-0	

15	0
CMP1CLR	
R/WP-0	

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

Table 13-39. RTI Compare 1 Clear Register (RTICMP1CLR) Field Descriptions

Bit	Field	Value	Description
31-0	CMP0CLR	0-FFFF FFFFh	Compare 1 clear. This registers holds a compare value. If the Free Running Counter matches the compare value, the compare 1 interrupt request flag is cleared and the value in the RTIUDCP1 register Section 13.3.18 is added to this register. Reads return the current compare clear value. A privileged write to this register updates the compare clear value.

13.3.38 RTI Compare 2 Clear Register (RTICMP2CLR)

This registers holds an initial value which is larger than the value in the RTI Compare 2 register Section 13.3.4. The user needs to choose the value such that the compare clear 2 event occurs before next compare 2 event. If the Free Running Counter matches the compare value, the compare 2 interrupt request flag is cleared and the value in the RTIUDCP2 register Section 13.3.20 is added to this register. This register is shown in Figure 13-49 and described in Table 13-40.

Figure 13-49. RTI Compare 2 Clear Register (RTICMP2CLR) [offset = B8h]

31	16
CMP2CLR	
R/WP-0	

15	0
CMP2CLR	
R/WP-0	

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

Table 13-40. RTI Compare 2 Clear Register (RTICMP2CLR) Field Descriptions

Bit	Field	Value	Description
31-0	CMP2CLR	0-FFFF FFFFh	Compare 2 clear. This registers holds a compare value. If the Free Running Counter matches the compare value, the compare 2 interrupt request flag is cleared and the value in the RTIUDCP2 register Section 13.3.20 is added to this register. Reads return the current compare clear value. A privileged write to this register updates the compare clear value.

13.3.39 RTI Compare 3 Clear Register (RTICMP3CLR)

This registers holds an initial value which is larger than the value in the RTI Compare 3 register Section 13.3.4. The user needs to choose the value such that the compare clear 3 event occurs before next compare 3 event. If the Free Running Counter matches the compare value, the compare 3 interrupt request flag is cleared and the value in the RTIUDCP3 register Section 13.3.22 is added to this register. This register is shown in Figure 13-50 and described in Table 13-41.

Figure 13-50. RTI Compare 3 Clear Register (RTICMP3CLR) [offset = BCh]

31	16
CMP3CLR	
R/WP-0	

15	0
CMP3CLR	
R/WP-0	

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

Table 13-41. RTI Compare 3 Clear Register (RTICMP3CLR) Field Descriptions

Bit	Field	Value	Description
31-0	CMP3CLR	0-FFFF FFFFh	Compare 3 clear. This registers holds a compare value. If the Free Running Counter matches the compare value, the compare 3 interrupt request flag is cleared and the value in the RTIUDCP3 register Section 13.3.22 is added to this register. Reads return the current compare clear value. A privileged write to this register updates the compare clear value.

Cyclic Redundancy Check (CRC) Controller Module

This chapter describes the cyclic redundancy check (CRC) controller module.

NOTE: This chapter describes a superset implementation of the CRC module that includes features and functionality that require DMA. Since not all devices have DMA capability, consult your device-specific datasheet to determine applicability of these features and functions to your device being used.

Topic **Page**

14.1 Overview **500**
14.2 Module Operation **502**
14.3 Example **513**
14.4 CRC Control Registers **516**

14.1 Overview

The CRC controller is a module that is used to perform CRC (Cyclic Redundancy Check) to verify the integrity of memory system. A signature representing the contents of the memory is obtained when the contents of the memory are read into CRC controller. The responsibility of CRC controller is to calculate the signature for a set of data and then compare the calculated signature value against a pre-determined good signature value. CRC controller supports two channels to perform CRC calculation on multiple memories in parallel and can be used on any memory system.

14.1.1 Features

The CRC controller offers:

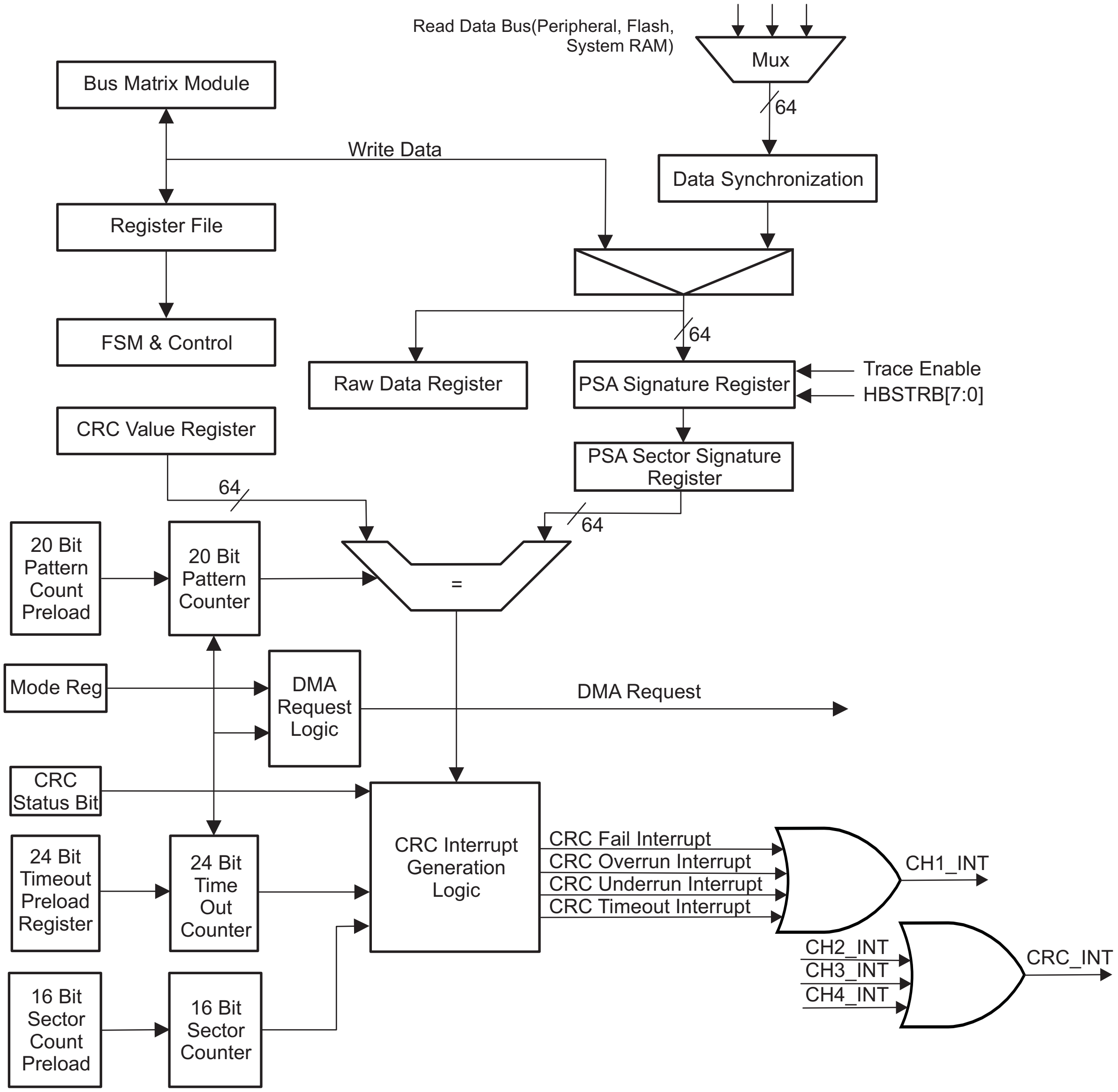
- Two channels to perform background signature verification on any memory sub-system.
- Data compression on 8, 16, 32, and 64 bit data size.
- Maximum-length PSA (Parallel Signature Analysis) register constructed based on 64 bit primitive polynomial.
- Each channel has a CRC Value Register that contains the pre-determined CRC value.
- Use timed base event trigger from timer to initiate DMA data transfer.
- Programmable 20-bit pattern counter per channel to count the number of data patterns for compression.
- Three modes of operation: Auto, Semi-CPU, and Full-CPU.
- For each channel, CRC can be performed either by CRC Controller or by CPU.
- Automatically perform signature verification without CPU intervention in AUTO mode.
- Generate interrupt to CPU in Semi-CPU mode to allow CPU to perform signature verification itself.
- Generate CRC fail interrupt in AUTO mode if signature verification fails.
- Generate Timeout interrupt if CRC is not performed within the time limit.
- Generate DMA request per channel to initiate CRC value transfer.
- Data trace capability on Peripheral Bus Master, Flash and System RAM data buses.

14.1.2 Block Diagram

Figure 14-1 shows a block diagram of the CRC controller.

NOTE: Only Channel 1 can support data trace. See Section 14.2.11.

Figure 14-1. CRC Controller Block Diagram For One Channel



14.2 Module Operation

14.2.1 General Operation

There are two channels in CRC controller and for each channel there is a memory-mapped PSA (Parallel Signature Analysis) Signature Register and a memory-mapped CRC (Cyclic Redundancy Check) Value register. A memory can be organized into multiple sectors with each sector consisting of multiple data patterns. A data pattern can be a 8, 16, 32, or 64 bit data. CRC module performs the signature calculation and compares the signature to a pre-determined value. The PSA Signature Register compresses an incoming data pattern into a signature when it is written. When one sector of data patterns are written into PSA Signature Register, a final signature corresponding to the sector is obtained. CRC Value Register stores the pre-determined signature corresponding to one sector of data patterns. The calculated signature and the pre-determined signature are then compared to each other for signature verification. To minimize CPU’s involvement, data patterns transfer can be carried out at the background of CPU using DMA controller. DMA is setup to transfer data from memory from which the contents to be verified to the memory-mapped PSA Signature Register. When DMA transfers data to the memory-mapped PSA Signature Register, a signature is generated. A programmable 20-bit data pattern counter is used for each channel to define the number of data patterns to calculate for each sector. Signature verification can be performed automatically by CRC controller in AUTO mode or by CPU itself in Semi-CPU or Full-CPU mode. In AUTO mode, a self sustained CRC signature calculation can be achieved without any CPU intervention. CRC Controller also provides data trace capability. Channel 1 can perform data trace on CPU data bus. During data trace, channel 1 monitors any data being read on CPU data bus and compresses it. When data trace is enabled for channel 1, all circuits related to DMA request and interrupt generation and counters are disabled.

14.2.2 CRC Modes of Operation

CRC Controller can operate in AUTO, Semi-CPU, and Full-CPU modes.

14.2.2.1 AUTO Mode

In AUTO mode, CRC Controller in conjunction with DMA controller can perform CRC totally without CPU intervention. A sustained transfer of data to both the PSA Signature Register and CRC Value Register are performed in the background of CPU. When a mismatch is detected, an interrupt is generated to CPU. A 16 bit current sector ID register is provided to identify which sector causes a CRC failure.

14.2.2.2 Semi-CPU Mode

In Semi-CPU mode, DMA controller is also utilized to perform data patterns transfer to PSA Signature Register. Instead of performing signature verification automatically, the CRC controller generates an compression complete interrupt to CPU after each sector is compressed. Upon responding to the interrupt the CPU performs the signature verification by reading the calculated signature stored at the PSA Sector Signature Register and compare it to a pre-determined CRC value.

14.2.2.3 Full CPU Mode

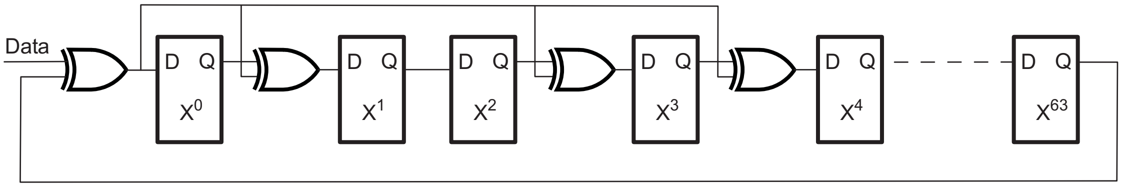
In Full-CPU mode, the CPU does the data patterns transfer and signature verification all by itself. When CPU has enough throughput, it can perform data patterns transfer by reading data from the memory system to the PSA Signature Register. After certain number of data patterns are compressed, the CPU can read from the PSA Signature Register and compare the calculated signature to the pre-determined CRC signature value. In Full-CPU mode, neither interrupt nor DMA request is generated. All counters are also disabled.

14.2.3 PSA Signature Register

The 64-bit PSA Signature Register is based on the primitive polynomial (as in the following equation) to produce the maximum length LFSR (Linear Feedback Shift Register), as shown in Figure 14-2.

$$f(x) = x^{64} + x^{4} + x^{3} + x + 1 \qquad (25)$$

Figure 14-2. LFSR



The serial implementation of LFSF has a limitation that, it requires ‘n’ clock cycles to calculate the CRC values for an ‘n’ bit data stream. The idea is to produce the same CRC value operating on a multi-bit data stream, as would occur if the CRC were computed one bit at a time over the whole data stream. The algorithm involves looping to simulate the shifting, and concatenating strings to build the equations after ‘n’ shift.

The parallel CRC calculation based on the polynomial can be illustrated in the following HDL code:

```
for i in 63 to 0 loop
   NEXT_CRC_VAL(0) := CRC_VAL(63) xor DATA(i);
     for j in 1 to 63 loop
       case j is
          when 1|3|4 =>
               NEXT_CRC_VAL(j) :=
                   CRC_VAL(j - 1) xor CRC_VAL(63) xor DATA(i);
          when others =>
               NEXT_CRC_VAL(j) := CRC_VAL(j - 1);
       end case;
     end loop;
   CRC_VAL := NEXT_CRC_VAL;
end loop;
```

NOTE: 1) The inner loop is to calculate the next value of each shift register bit after one cycle

2) The outer loop is to simulate 64 cycles of shifting. The equation for each shift register bit is thus built before it is compressed into the shift register.

3) MSB of the DATA is shifted in first

There is one PSA Signature Register per CRC channel. PSA Signature Register can be both read and written. When it is written, it can either compress the data or just capture the data depending on the state of CHx_MODE bits. If CHx_MODE=Data Capture, a seed value can be planted in the PSA Signature Register without compression. Other modes other than Data Capture will result with the data compressed by PSA Signature Register when it is written. Each channel can be planted with different seed value before compression starts. When PSA Signature Register is read, it gives the calculated signature.

CRC Controller should be used in conjunction with the on chip DMA controller to produce optimal system performance. The incoming data pattern to PSA Signature Register is typically initiated by the DMA master. When DMA is properly setup, it would read data from the pre-determined memory system and write them to the memory-mapped PSA Signature Register. Each time PSA Signature Register is written a signature is generated. CPU itself can also perform data transfer by reading from the memory system and perform write operation to PSA Signature Register if CPU has enough throughput to handle data patterns transfer.

After system reset and when AUTO mode is enabled, CRC Controller automatically generates a DMA request to request the pre-determined CRC value corresponding to the first sector of memory to be checked.

In AUTO mode, when one sector of data patterns is compressed, the signature stored at the PSA Signature Register is first copied to the PSA Sector Signature Register and PSA Signature Register is then cleared out to all zeros. An automatic signature verification is then performed by comparing the signature stored at the PSA Sector Signature Register to the CRC Value Register. After the comparison the CRC Controller can generate a DMA request. Upon receiving the DMA request the DMA controller will update the CRC Value Register by transferring the next pre-determined signature value associated with the next sector of memory system. If the signature verification fails then CRC Controller can generate a CRC fail interrupt.

In Full-CPU mode, no DMA request and interrupt are generated at all. The number of data patterns to be compressed is determined by CPU itself. Full-CPU mode is useful when DMA controller is not available to perform background data patterns transfer. The OS can periodically generate a software interrupt to CPU and use CPU to accomplish data transfer and signature verification.

CRC Controller supports doubleword, word, half word and byte access to the PSA Signature Register. During a non-doubleword write access, all unwritten byte lanes are padded with zero’s before compression. Note that comparison between PSA Sector Signature Register and CRC Value Register is always in 64 bit because a compressed value is always expressed in 64 bit.

There is a software reset per channel for PSA Signature Register. When set, the PSA Signature Register is reset to all zeros.

PSA Signature Register is reset to zero under the following conditions:

- System reset
- PSA Software reset
- One sector of data patterns are compressed

14.2.4 PSA Sector Signature Register

After one sector of data is compressed, the final resulting signature calculated by PSA Signature Register is transferred to the PSA Sector Signature Register. PSA Signature Register is a read only register. During Semi-CPU mode, the host CPU should read from the PSA Sector Signature Register instead of reading from PSA Signature Register for signature verification to avoid data coherency issue. The PSA Signature Register can be updated with new signature before the host CPU is able to retrieve it.

In Semi-CPU mode, no DMA request is generated. When one sector of data patterns is compressed, CRC controller first generates a compression complete interrupt. Responding to the interrupt, CPU will in the ISR read the PSA Sector Signature Register and compare it to the known good signature or write the signature value to another memory location to build a signature file. In Semi-CPU mode, CPU must perform the signature verification in a manner to prevent any overrun condition. The overrun condition occurs when the compression complete interrupt is generated after one sector of data patterns is compressed and CPU has not read from the PSA Sector Signature Register to perform necessary signature verification before PSA Sector Signature Register is overridden with a new value. An overrun interrupt can be enable to generate when overrun condition occurs. During Semi-CPU mode, the host CPU should read from the PSA Sector Signature Register instead of reading from PSA Signature Register for signature verification to avoid data coherency issue. The PSA Signature Register can be updated with new signature before the host CPU is able to retrieve it.

14.2.5 CRC Value Register

Associated with each channel there is a CRC Value Register. The CRC Value Register stores the pre-determined CRC value. After one sector of data patterns is compressed by PSA Signature Register, CRC Controller can automatically compare the resulting signature stored at the PSA Sector Signature Register with the pre-determined value stored at the CRC Value Register if AUTO mode is enabled. If the signature verification fails, CRC Controller can be enabled to generate an CRC fail interrupt. When the channel is set up for Semi-CPU mode, CRC controller first generates a compression complete interrupt to CPU. Upon servicing the interrupt, CPU will then read the PSA Sector Signature Register and then read the corresponding CRC value stored at another location and compare them. CPU should not read from the CRC Value Register during Semi-CPU or Full-CPU mode because the CRC Value Register is not updated during these two modes.

In AUTO mode, for first sector’s signature, DMA request is generated when mode is programmed to AUTO. For subsequent sectors, DMA request is generated after each sector is compressed. Responding to the DMA request, DMA controller reloads the CRC Value Register for the next sector of memory system to be checked.

When CRC Value Register is updated with a new CRC value, an internal flag is set to indicate that CRC Value Register contains the most current value. This flag is cleared when CRC comparison is performed. Each time at the end of the final data pattern compression of a sector, CRC Controller first checks to see if the corresponding CRC Value Register has the most current CRC value stored in it by polling the flag. If the flag is set then the CRC comparison can be performed. If the flag is not set then it means the CRC Value Register contains stale information. A CRC underrun interrupt is generated. When an underrun condition is detected, signature verification is not performed.

CRC Controller supports doubleword, word, half word and byte access to the CRC Value Register. As noted before comparison between PSA Sector Signature Register and CRC Value Register during AUTO mode is carried out in 64 bit.

14.2.6 Raw Data Register

The raw or un-compressed data written to the PSA Signature Register is also saved in the Raw Data Register. This register is read only.

14.2.7 Example DMA Controller Setup

DMA controller needs to be setup properly in either AUTO or Semi-CPU mode as DMA controller is used to transfer data patterns. Hardware or a combination of hardware and software DMA triggering are supported.

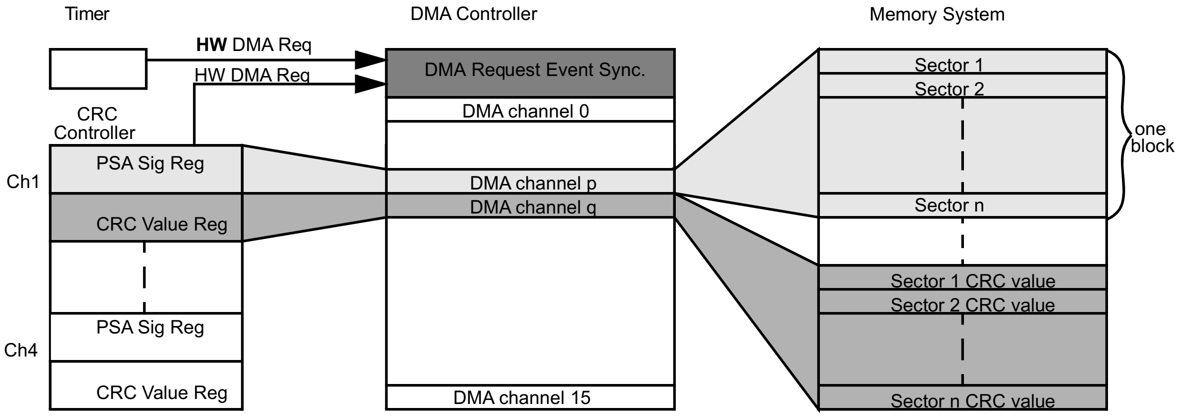
14.2.7.1 AUTO Mode Using Hardware Timer Trigger

There are two DMA channels associated with each CRC channel when in AUTO mode. One DMA channel is setup to transfer data patterns from the source memory to the PSA Signature Register. The second DMA channel is setup to transfer the pre-determined signature to the CRC Value Register. The trigger source for the first DMA channel can be either by hardware or by software. As illustrated in Figure 14-3 a timer can be used to trigger a DMA request to initiate transfer from the source memory system to PSA Signature Register. In AUTO mode, CRC Controller also generates DMA request after one sector of data patterns is compressed to initiate transfer of the next CRC value corresponding to the next sector of memory. Thus a new CRC value is always updated in the CRC Value Register by DMA synchronized to each sector of memory.

A block of memory system is usually divided into many sectors. All sectors are the same size. The sector size is programmed in the CRC_PCOUNT_REGx and the number of sectors in one block is programmed in the CRC_SCOUNT_REGx of the respective channel. CRC_PCOUNT_REGx multiplies CRC_SCOUNT_REGx and multiplies transfer size of each data pattern should give the total block size in number of bytes.

The total size of the memory system to be examined is also programmed in the respective transfer count register inside DMA module. The DMA transfer count register is divided into two parts. They are element count and frame count. Note that an HW DMA request can be programmed to trigger either one frame or one entire block transfer. In Figure 14-3, an HW DMA request from a timer is used as a trigger source to initiate DMA transfer. If all four CRC channels are active in AUTO mode then a total of four DMA requests would be generated by CRC Controller.

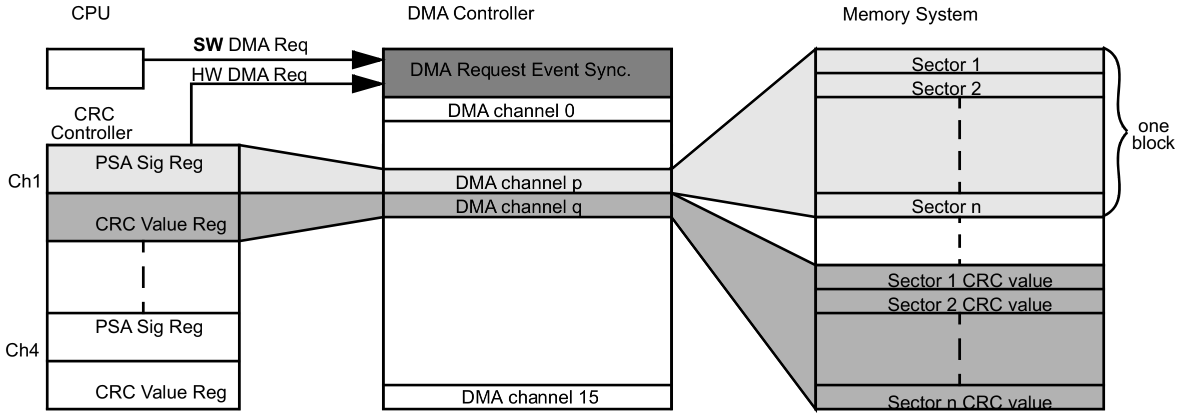
Figure 14-3. AUTO Mode Using Hardware Timer Trigger



14.2.7.2 AUTO Mode Using Software Trigger

The data patterns transfer can also be initiated by software. CPU can generate a software DMA request to activate the DMA channel to transfer data patterns from source memory system to the PSA Signature Register. To generate a software DMA request CPU needs to set the corresponding DMA channel in the DMA software trigger register. Note that just one software DMA request from CPU is enough to complete the entire data patterns transfer for all sectors. See Figure 14-4 for an illustration.

Figure 14-4. AUTO Mode With Software CPU Trigger



14.2.7.3 Semi-CPU Mode Using Hardware Timer Trigger

During semi-CPU mode, no DMA request is generated by CRC controller. Therefore, no DMA channel is allocated to update CRC Value Register. CPU should not read from CRC Value Register in semi-CPU mode as it contains stale value. Note that no signature verification is performed at all during this mode. Similar to AUTO mode, either by hardware or by software DMA request can be used as a trigger for data patterns transfer. Figure 14-5 illustrates the DMA setup using semi-CPU mode with hardware timer trigger.

Figure 14-5. Semi-CPU Mode With Hardware Timer Trigger

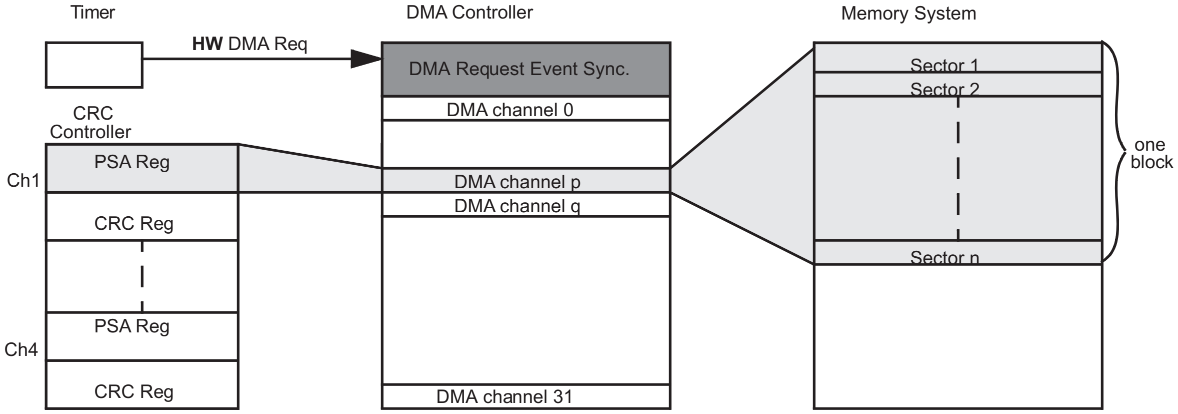


Table 14-1. CRC Modes in Which DMA Request and Counter Logic are Active or Inactive

Mode	DMA Request	Pattern Counter	Sector Counter	Timeout Counter
AUTO	Active	Active	Active	Active
Semi-CPU	Inactive	Active	Active	Active
Full-CPU	Inactive	Inactive	Inactive	Inactive

14.2.8 Pattern Count Register

There is a 20-bit data pattern counter for every CRC channel. The data pattern counter is a down counter and can be pre-loaded with a programmable value stored in the Pattern Count Register. When the data pattern counter reaches zero, a compression complete interrupt is generated in Semi-CPU mode and an automatic signature verification is performed in AUTO mode. In AUTO only, DMA request is generated to trigger the DMA controller to update the CRC Value Register.

NOTE: The data pattern count should be divisible by the total transfer count as programmed in DMA controller. The total transfer count is the product of element count and frame count.

14.2.9 Sector Count Register/Current Sector Register

Each channel contains a 16 bit sector counter. The sector count register stores the number of sectors. Sector counter is a free running counter and is incremented by one each time when one sector of data patterns is compressed. When the signature verification fails, the current value stored in the sector counter is saved into current sector register. If signature verification fails, CPU can read from the current sector register to identify the sector which causes the CRC mismatch. To aid and facilitate the CPU in determining the cause of a CRC failure, it is advisable to use the following equation during CRC and DMA setup:

CRC Pattern Count × CRC Sector Count = DMA Element Count × DMA Frame Count

The current sector register is frozen from being updated until both the current sector register is read and CRC fail status bit is cleared by CPU. If CPU does not respond to the CRC failure in a timely manner before another sector produces a signature verification failure, the current sector register is not updated with the new sector number. An overrun interrupt is generate instead. If current sector register is already frozen with an erroneous sector and emulation is entered with SUSPEND signal goes to high then the register still remains frozen even it is read.

In Semi-CPU mode, the current sector register is used to indicate the sector for which the compression complete has last happened.

The current sector register is reset when the PSA software reset is enabled.

NOTE: Both data pattern count and sector count registers must be greater than or equal to one for the counters to count. After reset, pattern count and sector count registers default to zero and the associated counters are inactive.

14.2.10 Interrupt

The CRC controller generates several types of interrupts per channel. Associated with each interrupt, there is an interrupt enable bit. No interrupt is generated in Full-CPU mode.

- Compression complete interrupt
- CRC fail interrupt
- Overrun interrupt
- Underrun interrupt
- Timeout interrupt

Table 14-2. Modes in Which Interrupt Condition Can Occur

	AUTO	Semi-CPU	Full-CPU
Compression Complete	no	yes	no
CRC Fail	yes	no	no
Overrun	yes	yes	no
Underrun	yes	no	no
Timeout	yes	yes	no

14.2.10.1 Compression Complete Interrupt

Compression complete interrupt is generated in Semi-CPU mode only. When the data pattern counter reaches zero, the compression complete flag is set and the interrupt is generated.

14.2.10.2 CRC Fail Interrupt

CRC fail interrupt is generated in AUTO mode only. When the signature verification fails, the CRC fail flag is set,. CPU should take action to address the fail condition and clear the CRC fail flag after it resolves the CRC mismatch.

14.2.10.3 Overrun Interrupt

Overrun interrupt is generated in either AUTO or Semi-CPU mode. During AUTO mode, if a CRC fail is detected, then the current sector number is recorded in the current sector register. If CRC fail status bit is not cleared and current sector register is not read by the host CPU before another CRC fail is detected for another sector, then an overrun interrupt is generated. During Semi-CPU mode, when the data pattern counter finishes counting, it generates a compression complete interrupt. At the same time, the signature is copied into the PSA Sector Signature Register. If the host CPU does not read the signature from PSA Sector Signature Register before it is updated again with a new signature value, then an overrun interrupt is generated.

14.2.10.4 Underrun Interrupt

Underrun interrupt only occurs in AUTO mode. The interrupt is generated when the CRC Value Register is not updated with the corresponding signature when the data pattern counter finishes counting. During AUTO mode, CRC Controller generates DMA request to update CRC Value Register in synchronization to the corresponding sector of the memory. Signature verification is also performed if underrun condition is detected. And CRC fail interrupt is generated at the same time as the underrun interrupt.

14.2.10.5 Timeout Interrupt

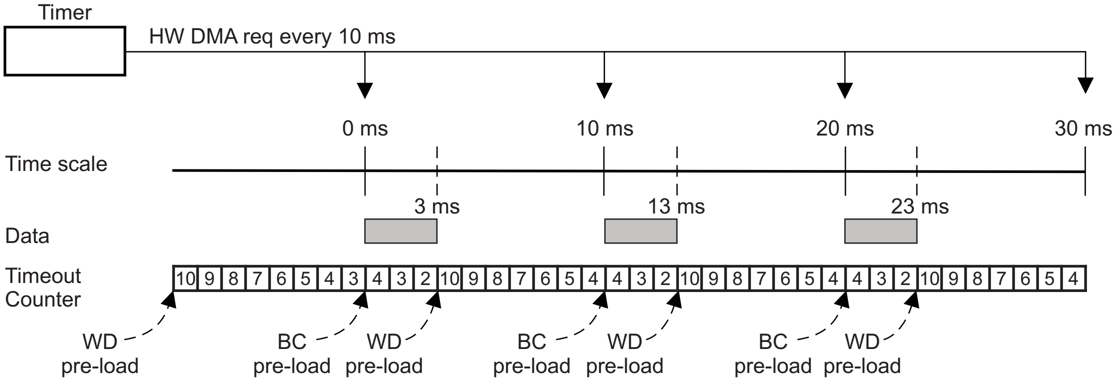
To ensure that the memory system is examined within a pre-defined time frame and no loss of incoming data there is a 24 bit timeout counter per CRC channel. The 24 bit timeout down counter can be pre-loaded with two different pre-load values, watchdog timeout pre-load value (CRC_WDTOPLDx) and block complete timeout pre-load value (CRC_BCTOPLDx). The timeout counter is clocked by a prescaler clock which is permanently running at division 64 of HCLK clock.

First pattern of data must be transferred by the DMA before the timeout counter expires, Watchdog timeout pre-load register (CRC_WDTOPLDx) is used as timeout counter. Block complete timeout pre-load register (CRC_BCTOPLDx) is used to check if one complete block of data patterns are compressed within a specific time frame. The timeout counter is first pre-loaded with CRC_WDTOPLDx after either AUTO or Semi-CPU mode is selected and starts to down count. If the timeout counter expires before DMA transfers any data pattern to PSA Signature Register then a timeout interrupt is generated. An incoming data pattern before the timeout counter expires will automatically pre-load the timeout counter with CRC_BCTOPLDx the block complete timeout pre-load value.

Block complete timeout pre-load value is used to check it one block of data patterns are compressed within a given time limit. If the timeout counter pre-loaded with CRC_BCTOPLDx value expires before one block of data patterns are compressed a timeout interrupt is generated. When one block (pattern count x sector count) of data patterns are compressed before the counter has expired, the counter is pre-loaded with CRC_WDTOPLDx value again. If the timeout counter is pre-loaded with zero then the counter is disable and no timeout interrupt is generated.

In Figure 14-6, a timer generates DMA request every 10ms to trigger one block (pattern count x sector count) transfer. Since we want to make sure that DMA does start to transfer a block every 10 ms we would set the first pre-load value to 10ms in CRC_WDTOPLDx. We also want to make sure that one block of data patterns are compressed within 4ms. With such a requirement, we would set the second pre-load value to 4ms in CRC_BCTOPLDx register.

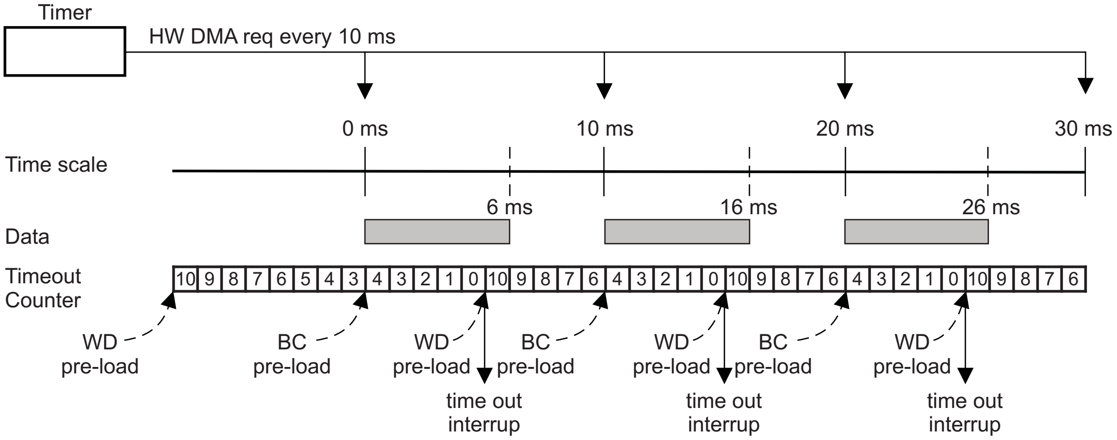
Figure 14-6. Timeout Example 1



WD pre-load = watchdog timeout pre-load (CRC_WDTOPLDx)
BC pre-load = block complete timeout pre-load (CRC_BCTOPLDx)

Note: No timeout interrupt is generated in this example since each block of data patterns are compressed in 3 ms and DMA does initiate a block transfer every 10 ms.

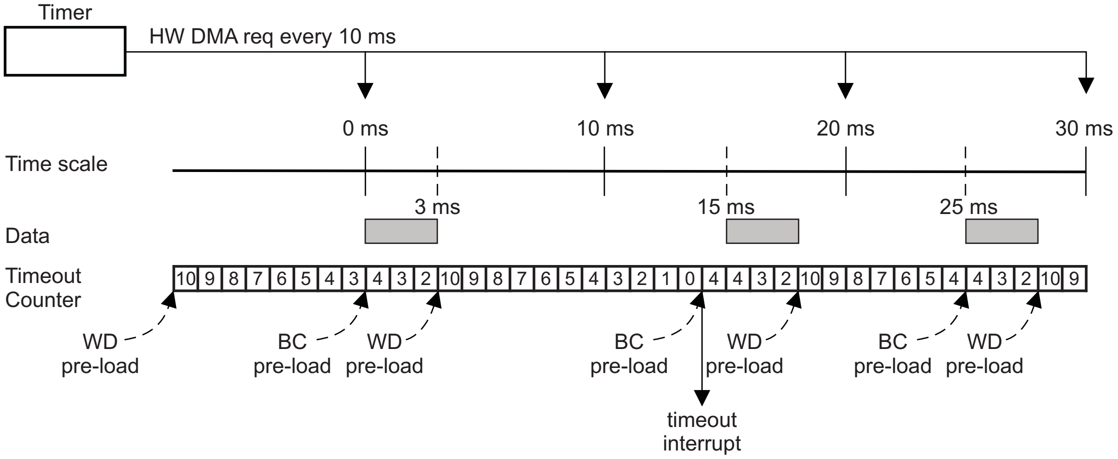
Figure 14-7. Timeout Example 2



WD pre-load = watchdog timeout pre-load (CRC_WDTOPLDx)
BC pre-load = block complete timeout pre-load (CRC_BCTOPLDx)

Note: Timeout interrupt is generated in this example since each block of data patterns are compressed in 6 ms and this is out of the 4ms time frame.

Figure 14-8. Timeout Example 3



WD pre-load = watchdog timeout pre-load (CRC_WDTOPLDx)
BC pre-load = block complete timeout pre-load (CRC_BCTOPLDx)

Note: Timeout interrupt is generated in this example since DMA can not transfer the second block of data within 10ms time limit and the reason may be that DMA is set up in fixed priority scheme and DMA is serving other higher priority channels at the time before it can service the timer request.

14.2.10.6 Interrupt Offset Register

CRC Controller only generates one interrupt request to interrupt manager. A interrupt offset register is provided to indicate the source of the pending interrupt with highest priority. Table 14-3 shows the offset interrupt vector address of each interrupt condition in an ascending order of priority.

Table 14-3. Interrupt Offset Mapping

Offset Value	Interrupt Condition
0	Phantom
1h	Ch1 CRC Fail
2h	Ch2 CRC Fail
3h-8h	Reserved
9h	Ch1 Compression Complete
Ah	Ch2 Compression Complete
Bh-10h	Reserved
11h	Ch1 Overrun
12h	Ch2 Overrun
13h-18h	Reserved
19h	Ch1 Underrun
1Ah	Ch2 Underrun
1Bh-20h	Reserved
21h	Ch1 Timeout
22h	Ch2 Timeout
23h-24h	Reserved

14.2.10.7 Error Handling

When an interrupt is generated, host CPU should take appropriate actions to identify the source of error and restart the respective channel in DMA and CRC module. To restart a CRC channel, the user should perform the following steps in the ISR:

1. Write to software reset bit in CRC_CTRL register to reset the respective PSA Signature Register.
2. Reset the CHx_MODE bits to 00 in CRC_CTRL register as Data capture mode.
3. Set the CHx_MODE bits in CRC_CTRL register to desired new mode again.
4. Release software reset.

The host CPU should use byte write to restart each individual channel.

14.2.11 CPU Data Trace

CRC channel 1 can be used to snoop Flash, System RAM and Peripheral Bus Master Data buses. However, at any one point only one bus is snooped. It is possible to disable the snooping of any of the buses by programming CRC_BUS_SEL register. While snooping the data, there is a priority scheme implemented between the buses. Peripheral Bus Master has the highest priority followed by Flash, Even System RAM and Odd System RAM. For each data read by CPU on its data bus the same data is compressed in the PSA Signature Register. A write to PSA Signature Register does not get compressed. Therefore, it is possible to write a seed value into PSA Signature Register before the bus snooping takes place. During data trace mode, all interrupts and DMA request logic are inactive. For non double word read on the data bus, all un-selected byte lanes are padded with zero during compression.

14.2.11.1 Data Capture Mode used in Conjunction with Data Trace

Data capture mode is especially useful when it is used in conjunction when data trace for channel 1 is enabled (CRC_CTRL2.CH1_TRACEEN = 1). The seed value can be planted in PSA Signature Register during data capture mode by writing a seed value into PSA Signature Register. The data trace enable bit is then set to snoop and compress any read transaction on DAHB bus. When trace enable bit is set, CRC_CTRL2.CH1_MODE is automatically reset to 0 (data capture mode). To change from one mode to another mode in the middle of an on-going mode, perform the following steps:

1. Assert software reset for the respective channel.
2. Change from the current active mode to Data Capture mode (CRC_CTRL2.CH1_MODE = 0).
3. Change from Data Capture mode to the new mode.
4. Release software reset.

14.2.12 Power Down Mode

CRC module can be put into power down mode when the power down control bit PWDN is set. The module wakes up when the PWDN bit is cleared. When CRC controller is in power down mode, no data tracing alone will happen. However, if CRC registers are accessed then data trace happens from channel 1.

14.2.13 Emulation

A read access from a register in functional mode can sometimes trigger a certain internal event to follow. For example, reading an interrupt offset register triggers an event to clear the corresponding interrupt status flag. During emulation when SUSPEND signal is high, a read access from any register should only return the register contents to the bus and should not trigger or mask any event as it would have in functional mode. This is to prevent debugger from reading the interrupt offset register during refreshing screen and cause the corresponding interrupt status flag to get cleared. Timeout counters are stopped to generate timeout interrupts in emulation mode. No Peripheral Master bus error should be generated if reading from the unimplemented locations. When channel 1 is placed under data trace, the PSA Signature Register does not compress any data read on CPU data bus when suspend is active.

14.2.14 Peripheral Bus Interface

CRC is a Peripheral slave module. The register interface is similar to other peripheral modules. CRC supports following features:

- Different sizes of burst operation.
- Aligned and unaligned accesses.
- Abort is generated for any illegal address accesses.

14.3 Example

This section illustrates several of the ways in which the CRC Controller can be utilized to perform CRC.

14.3.1 Example: Auto Mode Using Time Based Event Triggering

A large memory area with 2Mbyte (256k doubleword) is to be checked in the background of CPU. CRC is to be performed every 1K byte (128 doubleword). Therefore there should be 2048 pre-recorded CRC values. For illustration purpose, we map channel 1 CRC Value Register to DMA channel 1 and channel 1 PSA Signature Register to DMA channel 2. Assume all DMA transfers are carried out in 64-bit transfer size.

14.3.1.1 DMA Setup

- Set up DMA channel 1 with the starting address from which the pre-determined CRC values are stored. Set up the destination address to the memory-mapped channel 1 CRC Value Register. Put the source address at post increment addressing mode and put the destination address at constant addressing mode. Use ***hardware*** DMA request for channel 1 to trigger a ***frame*** transfer.
- Set up DMA channel 2 with the source address from which the contents of memory to be verified. Set up the destination address to the memory-mapped channel 1 PSA Signature Register. Program the element transfer count to 128 and the frame transfer count to 2048. Put the source address at post increment addressing mode and put the destination address at constant address mode. Use ***hardware*** DMA request for channel 2 to trigger an entire ***block*** transfer.

14.3.1.2 Timer Setup

The timer can be any general-purpose timer that is capable of generating a time-based DMA request.

- Set up timer to generate DMA request associated with DMA channel 2. For example, an OS can set up the timer to generate a DMA request every 10ms.

14.3.1.3 CRC Setup

- Program the pattern count to 128.
- Program the sector count to 2048.
- For example, we want the entire 2Mbytes to be compressed within 5ms. We can program the block complete timeout pre-load (CRC_BCTOPLDx) value to 15625 (5 ms / (1 HCLK period × 64)) if CRC is operating at 200 MHz.
- Enable AUTO mode and all interrupts.

After AUTO mode is selected, CRC Controller automatically generates a DMA request on channel 1. Around the same time the timer module also generates a DMA request on DMA channel 2. When the first incoming data pattern arrives at the PSA Signature Register, the CRC Controller will compress it. After some time, the DMA controller would update the CRC Value Register with a pre-determined value matching the calculated signature for the first sector of 128 64 bit data patterns. After one sector of data patterns are compressed, the CRC Controller generate a CRC fail interrupt if signature stored at the PSA Sector Signature Register does not match the CRC Value Register. CRC Controller generates a DMA request on DMA channel 1 when one sector of data patterns are compressed. This routine will continue until the entire 2Mbyte are consumed. If the timeout counter reached zero before the entire 2Mbytes are compressed a timeout interrupt is generated. After 2MBytes are transferred, the DMA can generate an interrupt to CPU. The entire operation will continue again when DMA responds to the DMA request from both the timer and CRC Controller. The CRC is performed totally without any CPU intervention.

14.3.2 Example: Auto Mode Without Using Time Based Triggering

A small but highly secured memory area with 1kbytes is to be checked in the background of CPU. CRC is to be performed every 1Kbytes. Therefore there is only one pre-recorded CRC value. For illustration purpose, we map channel 1 CRC Value Register to DMA channel 1 and channel 1 PSA Signature Register to DMA channel 2. Assume all transfers carried out by DMA are in 64 bit transfer size.

14.3.2.1 DMA Setup

- Set up DMA channel 1 with the source address from which the pre-determined CRC value is stored. Set up the destination address to the memory-mapped channel 1 CRC Value Register. Put the source address at constant addressing mode and put the destination address at constant addressing mode. Use ***hardware*** DMA request for channel 1.
- Set up DMA channel 2 with the source address from which the memory area to be verified. Set up the destination address to the memory-mapped channel 1 PSA Signature Register. Program the element transfer count to 128 and the frame transfer count to 1. Put the source address at post increment addressing mode and put the destination address at constant address mode. Generate a ***software*** DMA request on channel 2 after CRC has completed its setup. Enable autoinitiation for DMA channel 2.

14.3.2.2 CRC Setup

- Program the pattern count to 128.
- Program the sector count to 1.
- Leaving the timeout count register with the reset value of zero means no timeout interrupt is generated.
- Enable AUTO mode and all interrupts.

After AUTO mode is selected, the CRC Controller automatically generates a DMA request on channel 1. At the same time the CPU generates a ***software*** DMA request on DMA channel 2. When the first incoming data pattern arrives at the PSA Signature Register, the CRC Controller will compress it. After some time, the DMA controller would update the CRC Value Register with a pre-determined value matching the calculated signature for the first sector of 128 64 bit data patterns. After one sector of data patterns are compressed, the CRC Controller generates a CRC fail interrupt if signature stored at the PSA Sector Signature Register does not match the CRC Value Register. CRC Controller generates a DMA request on DMA channel 1 again after one sector is compressed. After 1kbytes are transferred, the DMA can generate an interrupt to CPU. Responding to the DMA interrupt CPU can restart the CRC routine by generating a software DMA request onto channel 2 again.

14.3.3 Example: Semi-CPU Mode

If DMA controller is available in a system, the CRC module can also operate in semi-CPU mode. This means that CPU can still make use of the DMA to perform data patterns transfer to CRC controller in the background. The difference between semi-CPU mode and AUTO mode is that CRC controller does not automatically perform the signature verification. CRC controllers generates a compression complete interrupt to CPU when the one sector of data patterns are compressed. CPU needs to perform the signature verification itself.

A memory area with 2Mbyte is to be verified with the help of the CPU. CRC operation is to be performed every 1K byte. Since there are 2Mbyte (256k doublewords) of memory to be check and we want to perform a CRC every 1Kbyte (128 doublewords) and therefore there should be 2048 pre-recorded CRC values. In Semi-CPU mode, the CRC Value Register is not updated and contains indeterminate data.

14.3.3.1 DMA Setup

Set up DMA channel 1 with the source address from which the memory area to be verified are mapped. Set up the destination address to the memory-mapped channel 1 PSA Signature Register. Put the starting address at post increment addressing mode and put the destination address at constant address mode. Use hardware DMA request to trigger an entire block transfer for channel 1. Disable autoinitiation for DMA channel 1.

14.3.3.2 Timer Setup

The timer can be any general-purpose timer that is capable of generating a time-based DMA request.

Set up timer to generate DMA request associated with DMA channel 1. For example, an OS can set up the timer to generate a DMA request every 10ms.

14.3.3.3 CRC Setup

- Program the pattern count to 128.
- Program the sector count to 2048.
- For example, we want the entire 2Mbytes to be compressed within 5ms. We can program the block complete timeout pre-load value to 15625 (5 ms / (1 HCLK period × 64)) if CRC is operating at 200 MHz.
- Enable Semi-CPU mode and enable all interrupts.

The timer module first generates a DMA request on DMA channel 1 when it is enabled. When the first incoming data pattern arrives at the PSA Signature Register, the CRC controller will compress it. After one sector of data patterns are compressed, the CRC controller generate a compression complete interrupt. Upon responding to the interrupt the CPU would read from the PSA Sector Signature Register. It is up to the CPU on how to deal with the PSA value just read. It can compare it to a known signature value or it can write it to another memory location to build a signature file or even transfer the signature out of the device via SCI or SPI. This routine will continue until the entire 2Mbyte are consumed. The latency of the interrupt response from CPU can cause overrun condition. If CPU does not read from PSA Sector Signature Register before the PSA value is overridden with the signature of the next sector of memory, an overrun interrupt will be generated by CRC controller.

14.3.4 Example: Full-CPU Mode

In a system without the availability of DMA controller, the CRC routine can be operated by CPU provided the CPU has enough throughput. CPU needs to read from the memory area from which CRC is to be performed.

A memory area with 2Mbyte is to be checked with the help of the CPU. CRC verification is to be performed every 1K byte. In CPU mode, the CRC Value Register is not updated and contains indeterminate data.

14.3.4.1 CRC Setup

- All control registers can be left in their reset state. Only enable Full-CPU mode.

CPU itself reads from the memory and write the data to the PSA Signature Register inside CRC Controller. When the first incoming data pattern arrives at the PSA Signature Register, the CRC Controller will compress it. After ***2MBytes*** data patterns are compressed, CPU can read from the PSA Signature Register. It is up to the CPU on how to deal with the PSA signature value just read. It can compare it to a known signature value stored at another memory location.

14.4 CRC Control Registers

All registers are in word boundary. 64-, 32-, 16-, and 8-bit write accesses are supported to all registers. The base address for the control registers is FE00 0000h.

Table 14-4. CRC Control Registers

Offset	Acronym	Register Description	Section
0h	CRC_CTRL0	CRC Global Control Register	Section 14.4.1
8h	CRC_CTRL1	CRC Global Control Register 1	Section 14.4.2
10h	CRC_CTRL2	CRC Global Control Register 2	Section 14.4.3
18h	CRC_INTS	CRC Interrupt Enable Set Register	Section 14.4.4
20h	CRC_INTR	CRC Interrupt Enable Reset Register	Section 14.4.5
28h	CRC_STATUS	CRC Interrupt Status Register	Section 14.4.6
30h	CRC_INT_OFFS_ET_REG	CRC Interrupt Offset Register	Section 14.4.7
38h	CRC_BUSY	CRC Busy Register	Section 14.4.8
40h	CRC_PCOUNT_REG1	CRC Channel 1 Pattern Counter Preload Register	Section 14.4.9
44h	CRC_SCOUNT_REG1	CRC Channel 1 Sector Counter Preload Register	Section 14.4.10
48h	CRC_CURSEC_REG1	CRC Channel 1 Current Sector Register	Section 14.4.11
4Ch	CRC_WDTOPLD1	CRC Channel 1 Watchdog Timeout Preload Register	Section 14.4.12
50h	CRC_BCTOPLD1	CRC Channel 1 Block Complete Timeout Preload Register	Section 14.4.13
60h	PSA_SIGREGL1	Channel 1 PSA Signature Low Register	Section 14.4.14
64h	PSA_SIGREGH1	Channel 1 PSA Signature High Register	Section 14.4.15
68h	CRC_REGL1	Channel 1 CRC Value Low Register	Section 14.4.16
6Ch	CRC_REGH1	Channel 1 CRC Value High Register	Section 14.4.17
70h	PSA_SECSIGREGL1	Channel 1 PSA Sector Signature Low Register	Section 14.4.18
74h	PSA_SECSIGREGH1	Channel 1 PSA Sector Signature High Register	Section 14.4.19
78h	RAW_DATAREGL1	Channel 1 Raw Data Low Register	Section 14.4.20
7Ch	RAW_DATAREGH1	Channel 1 Raw Data High Register	Section 14.4.21
80h	CRC_PCOUNT_REG2	CRC Channel 2 Pattern Counter Preload Register	Section 14.4.22
84h	CRC_SCOUNT_REG2	CRC Channel 2 Sector Counter Preload Register	Section 14.4.23
88h	CRC_CURSEC_REG2	CRC Current Sector Register 2	Section 14.4.24
8Ch	CRC_WDTOPLD2	CRC Channel 2 Watchdog Timeout Preload Register A	Section 14.4.25
90h	CRC_BCTOPLD2	CRC Channel 2 Block Complete Timeout Preload Register B	Section 14.4.26
A0h	PSA_SIGREGL2	Channel 2 PSA Signature Low Register	Section 14.4.27
A4h	PSA_SIGREGH2	Channel 2 PSA Signature High Register	Section 14.4.28
A8h	CRC_REGL2	Channel 2 CRC Value Low Register	Section 14.4.29
ACh	CRC_REGH2	Channel 2 CRC Value High Register	Section 14.4.30
B0h	PSA_SECSIGREGL2	Channel 2 PSA Sector Signature Low Register	Section 14.4.31
B4h	PSA_SECSIGREGH2	Channel 2 PSA Sector Signature High Register	Section 14.4.32
B8h	RAW_DATAREGL2	Channel 2 Raw Data Low Register	Section 14.4.33
BCh	RAW_DATAREGH2	Channel 2 Raw Data High Register	Section 14.4.34
140h	CRC_BUS_SEL	Data Bus Selection Register	Section 14.4.35

14.4.1 CRC Global Control Register 0 (CRC_CTRL0)

Figure 14-9. CRC Global Control Register 0 (CRC_CTRL0) [offset = 00h]

31	16
Reserved	
R-0	

15	9	8
Reserved		CH2_PSA_SWREST
R-0		R/W-0

7	1	0
Reserved		CH1_PSA_SWREST
R-0		R/W-0

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

Table 14-5. CRC Global Control Register 0 (CRC_CTRL0) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reads return 0. Writes have no effect.
8	CH2_PSA_SWREST		**Channel 2 PSA Software Reset.** When set, the PSA Signature Register is reset to all zero. Software reset does not reset software reset bit itself. Therefore, CPU is required to clear this bit by writing a 0.
		0	PSA Signature Register is not reset.
		1	PSA Signature Register is reset.
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	CH1_PSA_SWREST		**Channel 1 PSA Software Reset.** When set, the PSA Signature Register is reset to all zero. Software reset does not reset software reset bit itself. Therefore, CPU is required to clear this bit by writing a 0.
		0	PSA Signature Register is not reset.
		1	PSA Signature Register is reset.

14.4.2 CRC Global Control Register (CRC_CTRL1)

Figure 14-10. CRC Global Control Register 1 (CRC_CTRL1) [offset = 08h]

31	16
Reserved	
R-0	

15	1	0
Reserved		PWDN
R-0		RW-0

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

Table 14-6. CRC Global Control Register 1 (CRC_CTRL1) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	PWDN		**Power Down.** When set, CRC module is put in power-down mode
		0	CRC is not in power-down mode.
		1	CRC is in power-down mode.

14.4.3 CRC Global Control Register 2 (CRC_CTRL2)

Figure 14-11. CRC Global Control Register 2 (CRC_CTRL2) [offset = 10h]

31	16
Reserved	
R-0	

15	10	9	8
Reserved		CH2_MODE	
R-0		R/WP-0	

7	5	4	3	2	1	0
Reserved		CH1_TRACEEN	Reserved		CH1_MODE	
R-0		R/WP-0	R-0		R/WP-0	

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

Table 14-7. CRC Global Control Register 2 (CRC_CTRL2) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return 0. Writes have no effect.
9-8	CH2_MODE		**Channel 2 Mode Selection**
		0	Data Capture mode. In this mode, the PSA Signature Register does not compress data when it is written. Any data written to PSA Signature Register is simply captured by PSA Signature Register without any compression. This mode can be used to plant seed value into the PSA register.
		1h	AUTO Mode
		2h	Semi-CPU Mode
		3h	Full-CPU Mode
7-5	Reserved	0	Reads return 0. Writes have no effect.
4	CH1_TRACEEN		**Channel 1 Data Trace Enable.** When set, the channel is put into data trace mode. The channel snoops on the CPU Peripheral Bus Master, Flash, System RAM buses for any read transaction. Any read data on these buses is compressed by the PSA Signature Register. When suspend is on, the PSA Signature Register does not compress any read data on these buses. When trace enable bit is set, the CH1_MODE bit is automatically reset to 0 (Data Capture mode).
		0	Data Trace is disabled.
		1	Data Trace is enabled.
3-2	Reserved	0	Reads return 0. Writes have no effect.
1-0	CH1_MODE		**Channel 1 Mode Selection**
		0	Data Capture mode. In this mode, the PSA Signature Register does not compress data when it is written. Any data written to PSA Signature Register is simply captured by PSA Signature Register without any compression. This mode can be used to plant seed value into the PSA register.
		1h	AUTO mode
		2h	Semi-CPU mode
		3h	Full-CPU mode

14.4.4 CRC Interrupt Enable Set Register (CRC_INTS)

Figure 14-12. CRC Interrupt Enable Set Register (CRC_INTS) [offset = 18h]

31	16
Reserved	
R-0	

15	13	12	11	10	9	8
Reserved		CH2_ TIMEOUTENS	CH2_ UNDERENS	CH2_ OVERENS	CH2_ CRCFAILENS	CH2_ CCITENS
R-0		R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

7	5	4	3	2	1	0
Reserved		CH1_ TIMEOUTENS	CH1_ UNDERENS	CH1_ OVERENS	CH1_ CRCFAILENS	CH1_ CCITENS
R-0		R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

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

Table 14-8. CRC Interrupt Enable Set Register (CRC_INTS) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reads return 0. Writes have no effect.
12	CH2_TIMEOUTENS		**Channel 2 Timeout Interrupt Enable Bit**.
			User and Privileged mode (read):
		0	Timeout Interrupt is disabled.
		1	Timeout Interrupt is enabled.
			Privileged mode (write):
		0	Has no effect.
		1	Timeout Interrupt is enabled.
11	CH2_UNDERENS		**Channel 2 Underrun Interrupt Enable Bit**.
			User and Privileged mode (read):
		0	Underrun Interrupt is disabled.
		1	Underrun Interrupt is enabled.
			Privileged mode (write):
		0	Has no effect.
		1	Underrun Interrupt is enabled.
10	CH2_OVERENS		**Channel 2 Overrun Interrupt Enable Bit.**
			User and Privileged mode (read):
		0	Overrun Interrupt is disabled.
		1	Overrun Interrupt is enabled.
			Privileged mode (write):
		0	Has no effect.
		1	Overrun Interrupt is enabled.
9	CH2_CRCFAILENS		**Channel 2 CRC Fail Interrupt Enable Bit.**
			User and Privileged mode (read):
		0	CRC Fail Interrupt is disabled.
		1	CRC Fail Interrupt is enabled.
			Privileged mode (write):
		0	Has no effect.
		1	CRC Fail Interrupt is enabled.

Table 14-8. CRC Interrupt Enable Set Register (CRC_INTS) Field Descriptions (continued)

Bit	Field	Value	Description
8	CH2_CCITENS		**Channel 2 Compression Complete Interrupt Enable Bit.**
			User and Privileged mode (read):
		0	Compression Complete Interrupt is disabled.
		1	Compression Complete Interrupt is enabled.
			Privileged mode (write):
		0	Has no effect.
		1	Compression Complete Interrupt is enabled.
7-5	Reserved	0	Reads return 0. Writes have no effect.
4	CH1_TIMEOUTENS		**Channel 1 Timeout Interrupt Enable Bit**.
			User and Privileged mode (read):
		0	Timeout Interrupt is disabled.
		1	Timeout Interrupt is enabled.
			Privileged mode (write):
		0	Has no effect.
		1	Timeout Interrupt is enabled.
3	CH1_UNDERENS		**Channel 1 Underrun Interrupt Enable Bit**.
			User and Privileged mode (read):
		0	Underrun Interrupt is disabled.
		1	Underrun Interrupt is enabled.
			Privileged mode (write):
		0	Has no effect.
		1	Underrun Interrupt is enabled.
2	CH1_OVERENS		**Channel 1 Overrun Interrupt Enable Bit.**
			User and Privileged mode (read):
		0	Overrun Interrupt is disabled.
		1	Overrun Interrupt is enabled.
			Privileged mode (write):
		0	Has no effect.
		1	Overrun Interrupt is enabled.
1	CH1_CRCFAILENS		**Channel 1 CRC Fail Interrupt Enable Bit.**
			User and Privileged mode (read):
		0	CRC Fail Interrupt is disabled.
		1	CRC Fail Interrupt is enabled.
			Privileged mode (write):
		0	Has no effect.
		1	CRC Fail Interrupt is enabled.
0	CH1_CCITENS		**Channel 1 Compression Complete Interrupt Enable Bit.**
			User and Privileged mode (read):
		0	Compression Complete Interrupt is disabled.
		1	Compression Complete Interrupt is enabled.
			Privileged mode (write):
		0	Has no effect.
		1	Compression Complete Interrupt is enabled.

14.4.5 CRC Interrupt Enable Reset Register (CRC_INTR)

Figure 14-13. CRC Interrupt Enable Reset Register (CRC_INTR) [offset = 20h]

31	16
Reserved	
R-0	

15	13	12	11	10	9	8
Reserved		CH2_ TIMEOUTENR	CH2_ UNDERENR	CH2_ OVERENR	CH2_ CRCFAILENR	CH2_ CCITENR
R-0		R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

7	5	4	3	2	1	0
Reserved		CH1_ TIMEOUTENR	CH1_ UNDERENR	CH1_ OVERENR	CH1_ CRCFAILENR	CH1_ CCITENR
R-0		R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

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

Table 14-9. CRC Interrupt Enable Reset Register (CRC_INTR) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reads return 0. Writes have no effect.
12	CH2_TIMEOUTENR		**Channel 2 Timeout Interrupt Enable Reset Bit**.
			User and Privileged mode (read):
		0	Timeout Interrupt is disabled.
		1	Timeout Interrupt is enabled.
			Privileged mode (write):
		0	Has no effect.
		1	Timeout Interrupt is disabled.
11	CH2_UNDERENR		**Channel 2 Underrun Interrupt Enable Reset Bit**.
			User and Privileged mode (read):
		0	Underrun Interrupt is disabled.
		1	Underrun Interrupt is enabled.
			Privileged mode (write):
		0	Has no effect.
		1	Underrun Interrupt is disabled.
10	CH2_OVERENR		**Channel 2 Overrun Interrupt Enable Reset Bit.**
			User and Privileged mode (read):
		0	Overrun Interrupt is disabled
		1	Overrun Interrupt is enabled
			Privileged mode (write):
		0	Has no effect
		1	Overrun Interrupt disable
9	CH2_CRCFAILENR		**Channel 2 CRC Fail Interrupt Enable Reset Bit.**
			User and Privileged mode (read):
		0	CRC Fail Interrupt is disabled.
		1	CRC Fail Interrupt is enabled.
			Privileged mode (write):
		0	Has no effect.
		1	CRC Fail Interrupt is disabled.

Table 14-9. CRC Interrupt Enable Reset Register (CRC_INTR) Field Descriptions (continued)

Bit	Field	Value	Description
8	CH2_CCITENR		**Channel 2 Compression Complete Interrupt Enable Reset Bit.**
			User and Privileged mode (read):
		0	Compression Complete Interrupt is disabled.
		1	Compression Complete Interrupt is enabled.
			Privileged mode (write):
		0	Has no effect.
		1	Compression Complete Interrupt is disabled.
7-5	Reserved	0	Reads return 0. Writes have no effect.
4	CH1_TIMEOUTENR		**Channel 1 Timeout Interrupt Enable Reset Bit**.
			User and Privileged mode (read):
		0	Timeout Interrupt is disabled.
		1	Timeout Interrupt is enabled.
			Privileged mode (write):
		0	Has no effect.
		1	Timeout Interrupt is disabled.
3	CH1_UNDERENR		**Channel 1 Underrun Interrupt Enable Reset Bit**.
			User and Privileged mode (read):
		0	Underrun Interrupt is disabled.
		1	Underrun Interrupt is enabled.
			Privileged mode (write):
		0	Has no effect.
		1	Underrun Interrupt is disabled.
2	CH1_OVERENR		**Channel 1 Overrun Interrupt Enable Reset Bit.**
			User and Privileged mode (read):
		0	Overrun Interrupt is disabled.
		1	Overrun Interrupt is enabled.
			Privileged mode (write):
		0	Has no effect.
		1	Overrun Interrupt is disabled.
1	CH1_CRCFAILENR		**Channel 1 CRC Fail Interrupt Enable Reset Bit.**
			User and Privileged mode (read):
		0	CRC Fail Interrupt is disabled.
		1	CRC Fail Interrupt is enabled.
			Privileged mode (write):
		0	Has no effect.
		1	CRC Fail Interrupt is disabled.
0	CH1_CCITENR		**Channel 1 Compression Complete Interrupt Enable Reset Bit.**
			User and Privileged mode (read):
		0	Compression Complete Interrupt is disabled.
		1	Compression Complete Interrupt is enabled.
			Privileged mode (write):
		0	Has no effect.
		1	Compression Complete Interrupt is disabled.

14.4.6 CRC Interrupt Status Register (CRC_STATUS)

Figure 14-14. CRC Interrupt Status Register (CRC_STATUS) [offset = 28h]

31-16
Reserved
R-0

15-13	12	11	10	9	8
Reserved	CH2_TIMEOUT	CH2_UNDER	CH2_OVER	CH2_CRCFAIL	CH2_CCIT
R-0	R/W1CP-0	R/W1CP-0	R/W1CP-0	R/W1CP-0	R/W1CP-0

7-5	4	3	2	1	0
Reserved	CH1_TIMEOUT	CH1_UNDER	CH1_OVER	CH1_CRCFAIL	CH1_CCIT
R-0	R/W1CP-0	R/W1CP-0	R/W1CP-0	R/W1CP-0	R/W1CP-0

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset

Table 14-10. CRC Interrupt Status Register (CRC_STATUS) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reads return 0. Writes have no effect.
12	CH2_TIMEOUT		**Channel 2 CRC Timeout Interrupt Status Flag**. This bit is set in both AUTO and Semi-CPU mode.
			User and Privileged mode (read):
		0	No timeout interrupt is active.
		1	Timeout interrupt is active.
			Privileged mode (write):
		0	Has no effect.
		1	Bit is cleared.
11	CH2_UNDER		**Channel 2 CRC Underrun Interrupt Status Flag**. This bit is set in AUTO mode only.
			User and Privileged mode (read):
		0	No Underrun Interrupt is active.
		1	Underrun Interrupt is active.
			Privileged mode (write):
		0	Has no effect.
		1	Bit is cleared.
10	CH2_OVER		**Channel 2 CRC Overrun Interrupt Status Flag**. This bit is set in either AUTO or Semi-CPU mode.
			User and Privileged mode (read):
		0	No Overrun Interrupt is active.
		1	Overrun Interrupt is active.
			Privileged mode (write):
		0	Has no effect.
		1	Bit is cleared.
9	CH2_CRCFAIL		**Channel 2 CRC Compare Fail Interrupt Status Flag**. This bit is set in AUTO mode only.
			User and Privileged mode (read):
		0	No CRC Fail Interrupt is active.
		1	CRC Fail Interrupt is active.
			Privileged mode (write):
		0	Has no effect.
		1	Bit is cleared.

Table 14-10. CRC Interrupt Status Register (CRC_STATUS) Field Descriptions (continued)

Bit	Field	Value	Description
8	CH2_CCIT		**Channel 2 CRC Pattern Compression Complete Interrupt Status Flag**. This bit is only set in Semi-CPU mode.
			User and Privileged mode (read):
		0	No Compression Complete Interrupt is active.
		1	Compression Complete Interrupt is active.
			Privileged mode (write):
		0	Has no effect.
		1	Bit is cleared.
7-5	Reserved	0	Reads return 0. Writes have no effect.
4	CH1_TIMEOUT		**Channel 1 CRC Timeout Interrupt Status Flag**.
			User and Privileged mode (read):
		0	No timeout interrupt is active.
		1	Timeout interrupt is active.
			Privileged mode (write):
		0	Has no effect.
		1	Bit is cleared.
3	CH1_UNDER		**Channel 1 Underrun Interrupt Status Flag.**
			User and Privileged mode (read):
		0	No Underrun Interrupt is active.
		1	Underrun Interrupt is active.
			Privileged mode (write):
		0	Has no effect.
		1	Bit is cleared.
2	CH1_OVER		**Channel 1 Overrun Interrupt Status Flag.**
			User and Privileged mode (read):
		0	No Overrun Interrupt is active.
		1	Overrun Interrupt is active.
			Privileged mode (write):
		0	Has no effect.
		1	Bit is cleared.
1	CH1_CRCFAIL		**Channel 1 CRC Compare Fail Interrupt Status Flag.**
			User and Privileged mode (read):
		0	No CRC Fail Interrupt is active.
		1	CRC Fail Interrupt is active.
			Privileged mode (write):
		0	Has no effect.
		1	Bit is cleared.
0	CH1_CCIT		**Channel 1 CRC Pattern Compression Complete Interrupt Status Flag**.
			User and Privileged mode (read):
		0	No Compression Complete Interrupt is active.
		1	Compression Complete Interrupt is active.
			Privileged mode (write):
		0	Has no effect.
		1	Bit is cleared.

14.4.7 CRC Interrupt Offset (CRC_INT_OFFSET_REG)

Figure 14-15. CRC Interrupt Offset (CRC_INT_OFFSET_REG) [offset = 30h]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		OFSTREG	
R-0		R-0	

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

Table 14-11. CRC Interrupt Offset (CRC_INT_OFFSET_REG) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	OFSTREG		**CRC Interrupt Offset.** This register indicates the highest priority pending interrupt vector address. Reading the offset register automatically clears the respective interrupt flag.
		0	Phantom
		1h	Ch1 CRC Fail
		2h	Ch2 CRC Fail
		3h-8h	Reserved
		9h	Ch1 Compression Complete
		Ah	Ch2 Compression Complete
		Bh-10h	Reserved
		11h	Ch1 Overrun
		12h	Ch2 Overrun
		13h-18h	Reserved
		19h	Ch1 Underrun
		1Ah	Ch2 Underrun
		1Bh-20h	Reserved
		21h	Ch1 Timeout
		22h	Ch2 Timeout
		23h-FFh	Reserved

14.4.8 CRC Busy Register (CRC_BUSY)

Figure 14-16. CRC Busy Register (CRC_BUSY) [offset = 38h]

31–16
Reserved
R-0

15–9	8	7–1	0
Reserved	CH2_BUSY	Reserved	CH1_BUSY
R-0	R-0	R-0	R-0

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

Table 14-12. CRC Busy Register (CRC_BUSY) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reads return 0. Writes have no effect.
8	CH2_BUSY		**CH2_BUSY**. During AUTO or Semi-CPU mode, the busy flag is set when the first data pattern of the block is compressed and remains set until the last data pattern of the block is compressed. The flag is cleared when the last data pattern of the block is compressed.
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	CH1_BUSY		**CH1_BUSY**. During AUTO or Semi-CPU mode, the busy flag is set when the first data pattern of the block is compressed and remains set until the last data pattern of the block is compressed. The flag is cleared when the last data pattern of the block is compressed.

14.4.9 CRC Pattern Counter Preload Register 1 (CRC_PCOUNT_REG1)

Figure 14-17. CRC Pattern Counter Preload Register 1 (CRC_PCOUNT_REG1) [offset = 40h]

31–20	19–16
Reserved	CRC_PAT_COUNT1
R-0	R/W-0

15–0
CRC_PAT_COUNT1
R/W-0

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

Table 14-13. CRC Pattern Counter Preload Register 1 (CRC_PCOUNT_REG1) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reads return 0. Writes have no effect.
19-0	CRC_PAT_COUNT1		**Channel 1 Pattern Counter Preload Register**. This register contains the number of data patterns in one sector to be compressed before a CRC is performed.

14.4.10 CRC Sector Counter Preload Register 1 (CRC_SCOUNT_REG1)

Figure 14-18. CRC Sector Counter Preload Register 1 (CRC_SCOUNT_REG1) [offset = 44h]

31	16
Reserved	
R-0	

15	0
CRC_SEC_COUNT1	
R/W-0	

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

Table 14-14. CRC Sector Counter Preload Register 1 (CRC_SCOUNT_REG1) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	CRC_SEC_COUNT1		**Channel 1 Sector Counter Preload Register.** This register contains the number of sectors in one block of memory.

14.4.11 CRC Current Sector Register 1 (CRC_CURSEC_REG1)

Figure 14-19. CRC Current Sector Preload Register 1 (CRC_CURSEC_REG1) [offset = 48h]

31	16
Reserved	
R-0	

15	0
CRC_CURSEC1	
R-0	

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

Table 14-15. CRC Current Sector Register 1 (CRC_CURSEC_REG1) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	CRC_CURSEC1		**Channel 1 Current Sector ID Register.** In AUTO mode, this register contains the current sector number of which the signature verification fails. The sector counter is a free running up counter. When a sector fails, the erroneous sector number is logged into current sector ID register and the CRC fail interrupt is generated The sector ID register is frozen until it is read and the CRC fail status bit is cleared by CPU. While it is frozen, it does not capture another erroneous sector number. When this condition happens, an overrun interrupt is generated instead. Once the register is read and the CRC fail interrupt flag is cleared it can capture new erroneous sector number. In Semi-CPU mode, this register is used to indicate the sector number for which the compression complete has last happened.

14.4.12 CRC Channel 1 Watchdog Timeout Preload Register A (CRC_WDTOPLD1)

Figure 14-20. CRC Channel 1 Watchdog Timeout Preload Register A (CRC_WDTOPLD1) [offset = 4Ch]

31 24	23 16
Reserved	CRC_WDTOPLD1
R-0	R/W-0

15   0
CRC_WDTOPLD1
R/W-0

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

Table 14-16. CRC Channel 1 Watchdog Timeout Preload Register A (CRC_WDTOPLD1) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23-0	CRC_WDTOPLD1		**Channel 1 Watchdog Timeout Counter Preload Register.** This register contains the number of clock cycles within which the DMA must transfer the next block of data patterns. In Semi-CPU mode, this register is used to indicate the sector number for which the compression complete has last happened.

14.4.13 CRC Channel 1 Block Complete Timeout Preload Register B (CRC_BCTOPLD1)

Figure 14-21. CRC Channel 1 Block Complete Timeout Preload Register B (CRC_BCTOPLD1) [offset = 50h]

31 24	23 16
Reserved	CRC_BCTOPLD1
R-0	R/W-0

15   0
CRC_BCTOPLD1
R/W-0

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

Table 14-17. CRC Channel 1 Block Complete Timeout Preload Register B (CRC_BCTOPLD1) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23-0	CRC_BCTOPLD1		**Channel 1 Block Complete Timeout Counter Preload Register.** This register contains the number of clock cycles within which the CRC for an entire block needs to complete before a timeout interrupt is generated.

14.4.14 Channel 1 PSA Signature Low Register (PSA_SIGREGL1)

Figure 14-22. Channel 1 PSA Signature Low Register (PSA_SIGREGL1) [offset = 60h]

31 ... 0
PSASIG1
R/W-0

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

Table 14-18. Channel 1 PSA Signature Low Register (PSA_SIGREGL1) Field Descriptions

Bit	Field	Description
31-0	PSASIG1	**Channel 1 PSA Signature Low Register.** This register contains the value stored at PSASIG1[31:0] register.

14.4.15 Channel 1 PSA Signature High Register (PSA_SIGREGH1)

Figure 14-23. Channel 1 PSA Signature High Register (PSA_SIGREGH1) [offset = 64h]

31 ... 0
PSASIG1
R/W-0

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

Table 14-19. Channel 1 PSA Signature High Register (PSA_SIGREGH1) Field Descriptions

Bit	Field	Description
31-0	PSASIG1	**Channel 1 PSA Signature High Register.** This register contains the value stored at PSASIG1[63:32] register.

14.4.16 Channel 1 CRC Value Low Register (CRC_REGL1)

Figure 14-24. Channel 1 CRC Value Low Register (CRC_REGL1) [offset = 68h]

31 ... 0
CRC1
R/W-0

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

Table 14-20. Channel 1 CRC Value Low Register (CRC_REGL1) Field Descriptions

Bit	Field	Description
31-0	CRC1	**Channel 1 CRC Value Low Register.** This register contains the current known good signature value stored at CRC1[31:0] register.

14.4.17 Channel 1 CRC Value High Register (CRC_REGH1)

Figure 14-25. Channel 1 CRC Value High Register (CRC_REGH1) [offset = 6Ch]

31 — 0
CRC1
R/W-0

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

Table 14-21. Channel 1 CRC Value High Register (CRC_REGH1) Field Descriptions

Bit	Field	Description
31-0	CRC1	**Channel 1 CRC Value High Register.** This register contains the current known good signature value stored at CRC1[63:32] register.

14.4.18 Channel 1 PSA Sector Signature Low Register (PSA_SECSIGREGL1)

Figure 14-26. Channel 1 PSA Sector Signature Low Register (PSA_SECSIGREGL1) [offset = 70h]

31 — 0
PSASECSIG1
R-0

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

Table 14-22. Channel 1 PSA Sector Signature Low Register (PSA_SECSIGREGL1) Field Descriptions

Bit	Field	Description
31-0	PSASECSIG1	**Channel 1 PSA Sector Signature Low Register.** This register contains the value stored at PSASECSIG1[31:0] register.

14.4.19 Channel 1 PSA Sector Signature High Register (PSA_SECSIGREGH1)

Figure 14-27. Channel 1 PSA Sector Signature High Register (PSA_SECSIGREGH1) [offset = 74h]

31 — 0
PSASECSIG1
R-0

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

Table 14-23. Channel 1 PSA Sector Signature High Register (PSA_SECSIGREGH1) Field Descriptions

Bit	Field	Description
31-0	PSASECSIG1	**Channel 1 PSA Sector Signature High Register.** This register contains the value stored at PSASECSIG1[63:32] register.

14.4.20 Channel 1 Raw Data Low Register (RAW_DATAREGL1)

Figure 14-28. Channel 1 Raw Data Low Register (RAW_DATAREGL1) [offset = 78h]

31	0
RAW_DATA1	
R-0	

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

Table 14-24. Channel 1 Raw Data Low Register (RAW_DATAREGL1) Field Descriptions

Bit	Field	Description
31-0	RAW_DATA1	**Channel 1 Raw Data Low Register.** This register contains bits 31:0 of the uncompressed raw data.

14.4.21 Channel 1 Raw Data High Register (RAW_DATAREGH1)

Figure 14-29. Channel 1 Raw Data High Register (RAW_DATAREGH1) [offset = 7Ch]

31	0
RAW_DATA1	
R-0	

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

Table 14-25. Channel 1 Raw Data High Register (RAW_DATAREGH1) Field Descriptions

Bit	Field	Description
31-0	RAW_DATA1	**Channel 1 Raw Data High Register.** This register contains bits 63:32 of the uncompressed raw data.

14.4.22 CRC Pattern Counter Preload Register 2 (CRC_PCOUNT_REG2)

Figure 14-30. CRC Pattern Counter Preload Register 2 (CRC_PCOUNT_REG2) [offset = 80h]

31 18	19 16
Reserved	CRC_PAT_COUNT2
R-0	R/W-0

15	0
CRC_PAT_COUNT2	
R/W-0	

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

Table 14-26. CRC Pattern Counter Preload Register 2 (CRC_PCOUNT_REG2) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reads return 0. Writes have no effect.
19-0	CRC_PAT_COUNT2		**Channel 2 Pattern Counter Preload Register**. This register contains the number of data patterns in one sector to be compressed before a CRC is performed.

14.4.23 CRC Sector Counter Preload Register 2 (CRC_SCOUNT_REG2)

Figure 14-31. CRC Sector Counter Preload Register 2 (CRC_SCOUNT_REG2) [offset = 84h]

31	16
Reserved	
R-0	

15	0
CRC_SEC_COUNT2	
R/W-0	

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

Table 14-27. CRC Sector Counter Preload Register 2 (CRC_SCOUNT_REG2) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	CRC_SEC_COUNT2		**Channel 2 Sector Counter Preload Register.** This register contains the number of sectors in one block of memory.

14.4.24 CRC Current Sector Register 2 (CRC_CURSEC_REG2)

Figure 14-32. CRC Current Sector Register 2 (CRC_CURSEC_REG2) [offset = 88h]

31	16
Reserved	
R-0	

15	0
CRC_CURSEC2	
R-0	

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

Table 14-28. CRC Current Sector Register 2 (CRC_CURSEC_REG2) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	CRC_CURSEC2		**Channel 2 Current Sector ID Register.** In AUTO mode, this register contains the current sector number of which the signature verification fails. The sector counter is a free running up counter. When a sector fails, the erroneous sector number is logged into current sector ID register and the CRC fail interrupt is generated The sector ID register is frozen until it is read and the CRC fail status bit is cleared by CPU. While it is frozen, it does not capture another erroneous sector number. When this condition happens, an overrun interrupt is generated instead. Once the register is read and the CRC fail interrupt flag is cleared it can capture new erroneous sector number. In Semi-CPU mode, this register is used to indicate the sector number for which the compression complete has last happened.

14.4.25 CRC Channel 2 Watchdog Timeout Preload Register A (CRC_WDTOPLD2)

Figure 14-33. CRC Channel 2 Watchdog Timeout Preload Register A (CRC_WDTOPLD2) [offset = 8Ch]

31 24	23 16
Reserved	CRC_WDTOPLD2
R-0	R/W-0

15 0
CRC_WDTOPLD2
R/W-0

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

Table 14-29. CRC Channel 2 Watchdog Timeout Preload Register A (CRC_WDTOPLD2) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23-0	CRC_WDTOPLD2		**Channel 2 Watchdog Timeout Counter Preload Register.** This register contains the number of clock cycles within which the DMA must transfer the next block of data patterns. In Semi-CPU mode, this register is used to indicate the sector number for which the compression complete has last happened.

14.4.26 CRC Channel 2 Block Complete Timeout Preload Register B (CRC_BCTOPLD2)

Figure 14-34. CRC Channel 2 Block Complete Timeout Preload Register B (CRC_BCTOPLD2) [offset = 90h]

31 24	23 16
Reserved	CRC_BCTOPLD2
R-0	R/W-0

15 0
CRC_BCTOPLD2
R/W-0

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

Table 14-30. CRC Channel 2 Block Complete Timeout Preload Register B (CRC_BCTOPLD2) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23-0	CRC_BCTOPLD2		**Channel 2 Block Complete Timeout Counter Preload Register.** This register contains the number of clock cycles within which the CRC for an entire block needs to complete before a timeout interrupt is generated.

14.4.27 Channel 2 PSA Signature Low Register (PSA_SIGREGL2)

Figure 14-35. Channel 2 PSA Signature Low Register (PSA_SIGREGL2) [offset = A0h]

31	0
PSASIG2	
R/W-0	

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

Table 14-31. Channel 2 PSA Signature Low Register (PSA_SIGREGL2) Field Descriptions

Bit	Field	Description
31-0	PSASIG2	**Channel 2 PSA Signature Low Register.** This register contains the value stored at PSASIG2[31:0] register.

14.4.28 Channel 2 PSA Signature High Register (PSA_SIGREGH2)

Figure 14-36. Channel 2 PSA Signature High Register (PSA_SIGREGH2) [offset = A4h]

31	0
PSASIG2	
R/W-0	

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

Table 14-32. Channel 2 PSA Signature High Register (PSA_SIGREGH2) Field Descriptions

Bit	Field	Description
31-0	PSASIG2	**Channel 2 PSA Signature High Register.** This register contains the value stored at PSASIG2[63:32] register.

14.4.29 Channel 2 CRC Value Low Register (CRC_REGL2)

Figure 14-37. Channel 2 CRC Value Low Register (CRC_REGL2) [offset = A8h]

31	0
CRC2	
R/W-0	

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

Table 14-33. Channel 2 CRC Value Low Register (CRC_REGL2) Field Descriptions

Bit	Field	Description
31-0	CRC2	**Channel 2 CRC Value Low Register.** This register contains the current known good signature value stored at CRC2[31:0] register.

14.4.30 Channel 2 CRC Value High Register (CRC_REGH2)

Figure 14-38. Channel 2 CRC Value High Register (CRC_REGH2) [offset = ACh]

<table>
<tr><th>31</th><th>0</th></tr>
<tr><td colspan="2">CRC2</td></tr>
<tr><td colspan="2">R/W-0</td></tr>
</table>

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

Table 14-34. Channel 2 CRC Value High Register (CRC_REGH2) Field Descriptions

Bit	Field	Description
31-0	CRC2	**Channel 2 CRC Value High Register.** This register contains the current known good signature value stored at CRC2[63:32] register.

14.4.31 Channel 2 PSA Sector Signature Low Register (PSA_SECSIGREGL2)

Figure 14-39. Channel 2 PSA Sector Signature Low Register (PSA_SECSIGREGL2) [offset = B0h]

<table>
<tr><th>31</th><th>0</th></tr>
<tr><td colspan="2">PSASECSIG2</td></tr>
<tr><td colspan="2">R-0</td></tr>
</table>

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

Table 14-35. Channel 2 PSA Sector Signature Low Register (PSA_SECSIGREGL2) Field Descriptions

Bit	Field	Description
31-0	PSASECSIG2	**Channel 2 PSA Sector Signature Low Register.** This register contains the value stored at PSASECSIG2[31:0] register.

14.4.32 Channel 2 PSA Sector Signature High Register (PSA_SECSIGREGH2)

Figure 14-40. Channel 2 PSA Sector Signature High Register (PSA_SECSIGREGH2) [offset = B4h]

<table>
<tr><th>31</th><th>0</th></tr>
<tr><td colspan="2">PSASECSIG2</td></tr>
<tr><td colspan="2">R-0</td></tr>
</table>

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

Table 14-36. Channel 2 PSA Sector Signature High Register (PSA_SECSIGREGH2) Field Descriptions

Bit	Field	Description
31-0	PSASECSIG2	**Channel 2 PSA Sector Signature High Register.** This register contains the value stored at PSASECSIG2[63:32] register.

14.4.33 Channel 2 Raw Data Low Register (RAW_DATAREGL2)

Figure 14-41. Channel 2 Raw Data Low Register (RAW_DATAREGL2) [offset = B8h]

31	0
RAW_DATA2	
R-0	

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

Table 14-37. Channel 2 Raw Data Low Register (RAW_DATAREGL2) Field Descriptions

Bit	Field	Description
31-0	RAW_DATA2	**Channel 2 Raw Data Low Register.** This register contains bits 31:0 of the uncompressed raw data..

14.4.34 Channel 2 Raw Data High Register (RAW_DATAREGH2)

Figure 14-42. Channel 2 Raw Data High Register (RAW_DATAREGH2) [offset = BCh]

31	0
RAW_DATA2	
R-0	

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

Table 14-38. Channel 2 Raw Data High Register (RAW_DATAREGH2) Field Descriptions

Bit	Field	Description
31-0	RAW_DATA2	**Channel 2 Raw Data High Register.** This register contains bits 63:32 of the uncompressed raw data..

14.4.35 Data Bus Selection Register (CRC_TRACE_BUS_SEL)

Figure 14-43. Data Bus Selection Register (CRC_TRACE_BUS_SEL) [offset = 140h]

31-16
Reserved
R-0

15-3	2	1	0
Reserved	MEn	DTCMEn	ITCMEn
R-0	R/W-1	R/W-1	R/W-1

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

Table 14-39. Data Bus Selection Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	MEn		Enable/disables the tracing of Peripheral Bus Master
		0	Tracing of Peripheral Bus Master has been disabled.
		1	Tracing of Peripheral Bus Master has been enabled.
1	DTCMEn		Enable/disables the tracing of data TCM
		0	Tracing of System Odd and Even RAM buses have been disabled.
		1	Tracing of System Odd and Even RAM buses have been enabled.
0	ITCMEn		Enable/disables the tracing of instruction TCM
		0	Tracing of Flash data bus has been disabled.
		1	Tracing of Flash data bus has been enabled.

Chapter 15

SPNU514C–March 2018

Vectored Interrupt Manager (VIM) Module

This chapter describes the behavior of the vectored interrupt manager (VIM) module of the device family.

Topic **Page**

15.1 Overview 539
15.2 Device Level Interrupt Management 539
15.3 Interrupt Handling Inside VIM 542
15.4 Interrupt Vector Table (VIM RAM) 546
15.5 VIM Wakeup Interrupt 549
15.6 Capture Event Sources 550
15.7 Examples 550
15.8 VIM Registers 553

15.1 Overview

The vectored interrupt manager (VIM) provides hardware assistance for prioritizing and controlling the many interrupt sources present on a device. Interrupts are caused by events outside of the normal flow of program execution. Normally, these events require a timely response from the central processing unit (CPU); therefore, when an interrupt occurs, the CPU switches execution from the normal program flow to an interrupt service routine (ISR).

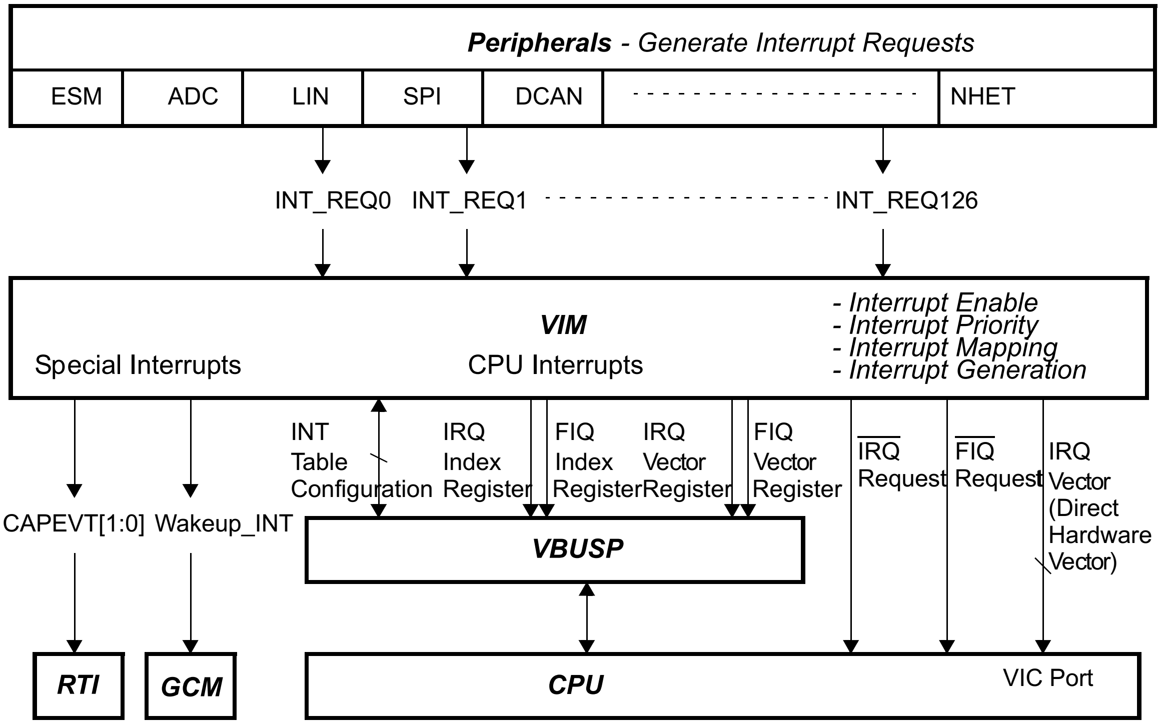
The VIM module has the following features:

- Supports 128 interrupt channels, in both register vectored interrupt and hardware vectored interrupt mode.
  - Provides IRQ vector directly to the CPU VIC port
  - Provides FIQ/IRQ vector through registers
  - Provides programmable priority and enable for interrupt request lines
- Provides a direct hardware dispatch mechanism for fastest IRQ dispatch.
- Provides two software dispatch mechanisms for backward compatibility with earlier generation of TI processors.
  - Index interrupt
  - Register vectored interrupt
- Parity protected vector interrupt table against soft errors.

15.2 Device Level Interrupt Management

A block diagram of device level interrupt handling is shown in Figure 15-1. When an event occurs within a peripheral, the peripheral makes an interrupt request to the VIM. Then, VIM prioritizes the requests from peripherals and provides the address of the highest interrupt service routine (ISR) to the CPU. Finally, CPU starts executing the ISR instructions from that address in the ISR. Section 15.2.1 through Section 15.2.3 provide additional details about these three steps.

Figure 15-1. Device Level Interrupt Block Diagram



15.2.1 Interrupt Generation at the Peripheral

Interrupt generation begins when an event occurs within a peripheral module. Some examples of interrupt-capable events are expiration of a counter within a timer module, receipt of a character in a communications module, and completion of a conversion in an analog-to-digital converter (ADC) module. Some device peripherals are capable of requesting interrupts on more than one interrupt request line.

Interrupts are not always generated when an event occurs; the peripheral must make an interrupt request to the VIM based on the event occurrence. Typically, the peripheral contains:

- An interrupt flag bit for each event to signify the event occurrence.
- An interrupt enable bit to control whether the event occurrence causes an interrupt request to the VIM.

15.2.2 Interrupt Handling at the CPU

The ARM CPU provides two vectors for interrupt requests—fast interrupt requests (FIQs) and normal interrupt requests (IRQs). FIQs are higher priority than IRQs, and FIQ interrupts may interrupt IRQ interrupts.

NOTE: The FIQ implemented in Cortex-R4F is Non-Maskable Fast Interrupts (NMFI). Once FIQ is enabled (by clearing F bit in CPSR), it can NOT be disabled by setting F bit in CPSR. Only a reset or an FIQ will be able to set the F bit in CPSR. By hardware, Non Maskable FIQ are not reentrant.

After reset (power reset or warm reset), both FIQ and IRQ are disabled. The CPU may enable these interrupt request channels individually within the CPSR (Current Program Status Register); CPSR bits 6 and 7 must be cleared to enable the FIQ (bit 6) and IRQ (bit 7) interrupt requests at the CPU. CPSR is writable in privilege mode only. Example 15-2 shows how to enable the IRQ and FIQ through CPSR.

When the CPU receives an interrupt request, the CPSR mode field changes to either FIQ or IRQ mode. When an IRQ interrupt is received, the CPU disables other IRQ interrupts by setting CPSR bit 7. When an FIQ interrupt is received, the CPU disables both IRQ and FIQ interrupts by setting CPSR bits 6 and 7.

A write of 1 to CPSR bit 7 disables the IRQ from CPU. However, a write of 1 to CPSR bit 6 leaves it unchanged. Example 15-2 also shows how to disable the IRQ through CPSR.

15.2.3 Software Interrupt Handling Options

The device supports three different possibilities for software to handle interrupts

1. Index interrupts mode (compatible with TMS470R1x legacy code)

   After the interrupt is received by the CPU, the CPU branches to 0x18 (IRQ) or 0x1C (FIQ) to execute the main ISR. The main ISR routine reads the offset register (IRQINDEX, FIQINDEX) to determine the source of the interrupt.

   This mode is compatible with the TMS470R1x (CIM) module and provides the same interrupt registers.

   This mode could be used if legacy code needs to be reused, porting it from the TMS470R1x family. However, imported software will not benefit from the VIM improvements.

   To port legacy software, the interrupt vector at 0x18 (IRQ) or 0x1C (FIQ) only needs to be a branch statement to a software interrupt table. The software interrupt table reads the pending interrupt from a vector offset register (FIQINDEX[7:0] for FIQ interrupts and IRQINDEX[7:0] for IRQ interrupts). All pending interrupts can be viewed in the INTREQ register. Example 15-4 shows how to respond to FIQ with short latency in this mode.

2. Register vectored interrupts (automatically provide vector address to application)

   Before enabling interrupts, the application software also has to initiate the interrupt vector table (VIM RAM).

   Once the VIM receives an interrupt, it loads the address of ISR from interrupt vector table, and store it into the interrupt vector register (IRQVECREG for IRQ interrupt, FIQVECREG for FIQ interrupt).

   After the interrupt is received by the CPU, the CPU executes the instruction placed at 0x18 or 0x1C (IRQ or FIQ vector) to load the address of ISR (interrupt vector) from the interrupt vector register. Example 15-3 illustrates the configuration for the exception vectors using this mode.

3. Hardware vectored interrupts (automatically dispatch to ISR, IRQ only)

   Before enabling interrupts, the application software must initiate the interrupt vector table (VIM RAM) pointing to the ISR for each interrupt channel.

   After the interrupt (IRQ) is received by the CPU, CPU reads the address of ISR directly from the interface with VIM (VIC port) instead of branching to 0x18. The CPU will branch directly to the ISR.

   The hardware vectored interrupt behavior must be explicitly enabled by setting the vector enable (VE) bit in the CP15 R1 register. This bit resets to 0, so that the default state after reset is backward compatible to earlier ARM CPU. Example 15-1 shows how to enable the hardware vectored interrupt.

   NOTE: This mode is NOT available for FIQ.

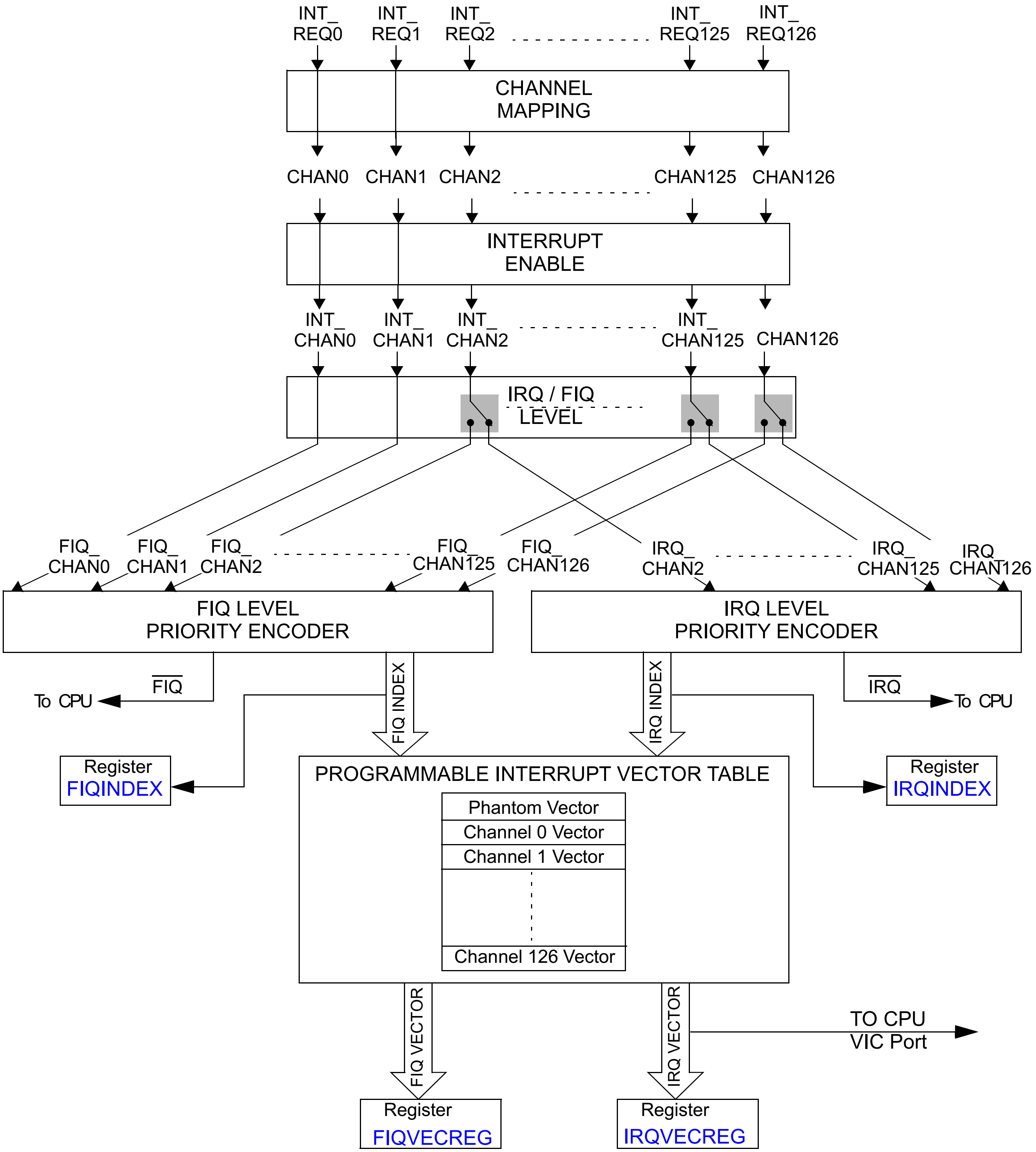
4. Software-Based Priority Decoding Scheme

   If the application uses a software-based interrupt priority decoding scheme instead of the hardware vector capabilities, then there is an additional step which was not required on earlier devices. This version of the VIM will hold an interrupt request generated by a peripheral. When the software clears the interrupt condition in the source module (for example, RTI, GIO, and so on), then it must also perform an additional clear of the interrupt request in the VIM. This can be done by reading the IRQVECREG register (Section 15.8.12) or FIQVECREG register (Section 15.8.13), or by writing a 1 to the INTREQ(i) bit (Section 15.8.7) in the VIM. This is not necessary if any of the three previous methods are used as the interrupt request bit in the VIM will be automatically cleared when the vector is read.

15.3 Interrupt Handling Inside VIM

A block diagram of the interrupt handling inside VIM is shown in Figure 15-2

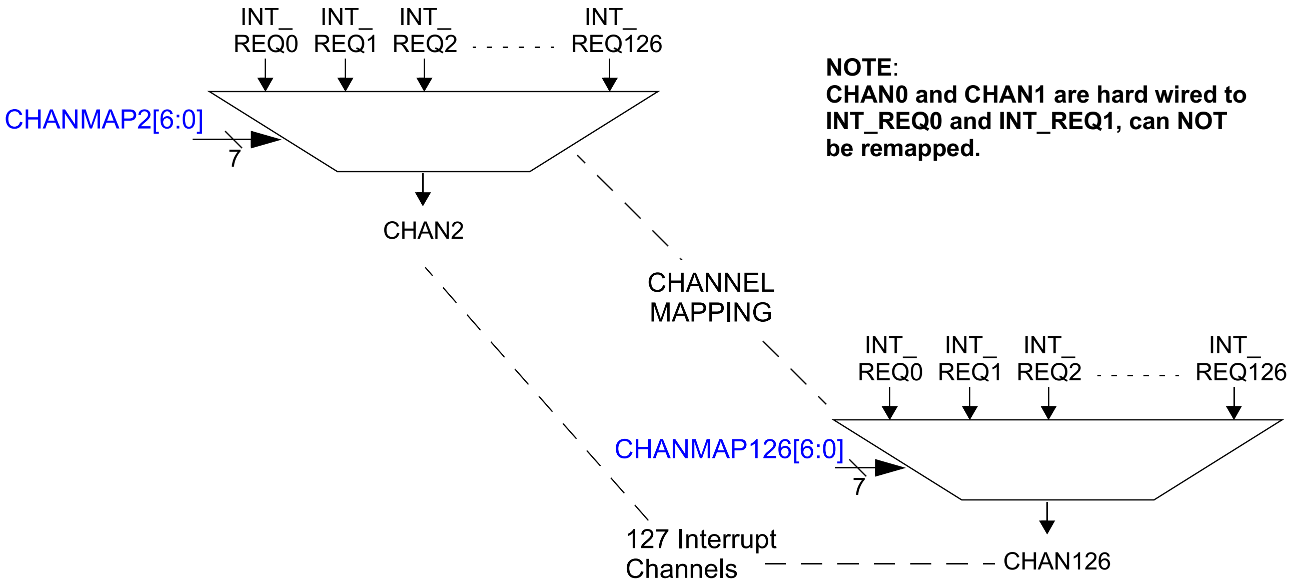
Figure 15-2. VIM Interrupt Handling Block Diagram



15.3.1 VIM Interrupt Channel Mapping

The VIM supports 128 interrupt channels (including phantom interrupt). A block diagram of the VIM interrupt requests arrangement from peripheral modules to the interrupt channels is provided in Figure 15-3. Each interrupt channel (CHANx) has a corresponding mapping register bit field (CHANMAPx[6:0]). This mapping register determines which interrupt channel it maps each VIM interrupt request. With this scheme, the same request can be mapped to multiple channels. A lower numbered channel in each FIQ and IRQ has higher priority. The programmability of the VIM allows software to control the interrupt priority.

Figure 15-3. VIM Channel Mapping



NOTE: **CHAN0 and CHAN1 are hard wired to INT_REQ0 and INT_REQ1, can NOT be remapped.**

NOTE: CHAN127

CHAN127 has no dedicated interrupt vector table entry. Therefore, CHAN127 shall NOT be remapped to other INT_REQ (INT_REQ127 is reserved at device level).

In the reset state, the VIM maps all of the interrupt requests in the system to their respective interrupt channels. Figure 15-4 shows the default state following the reset.

Figure 15-5 shows the VIM INT2 is remapped to both Channel 2 and 4, and INT3 is mapped to channel 3.

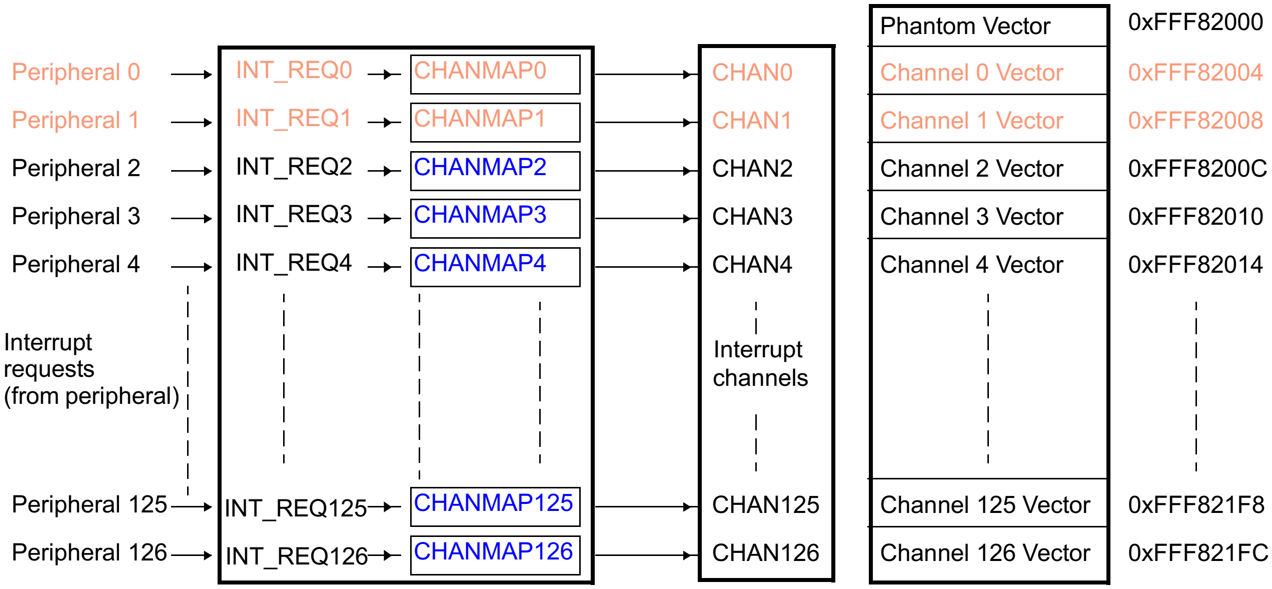
NOTE: By mapping INT2 to channel 2 and channel 4, and mapping INT3 to channel 3, it is possible for the software to change the priority dynamically by changing the ENABLE register (REQENASET and REQENACLR). When channel 2 is enabled, the priority is:

1. INT0
2. INT1
3. INT2
4. INT3

Disabling channel 2, the priority becomes:

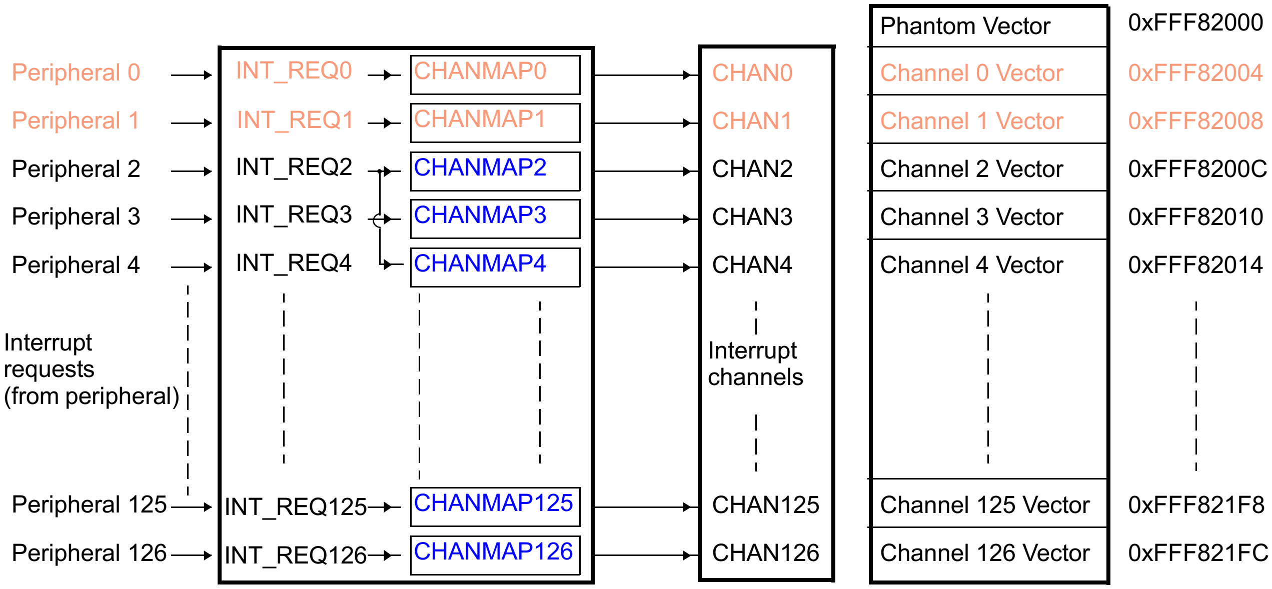
1. INT0
2. INT1
3. INT3
4. INT2

Figure 15-4. VIM in Default State



NOTE: CHAN0 and CHAN1 are hardwired to INT_REQ0 and INT_REQ1, so they cannot be remapped.

Figure 15-5. VIM in a Programmed State



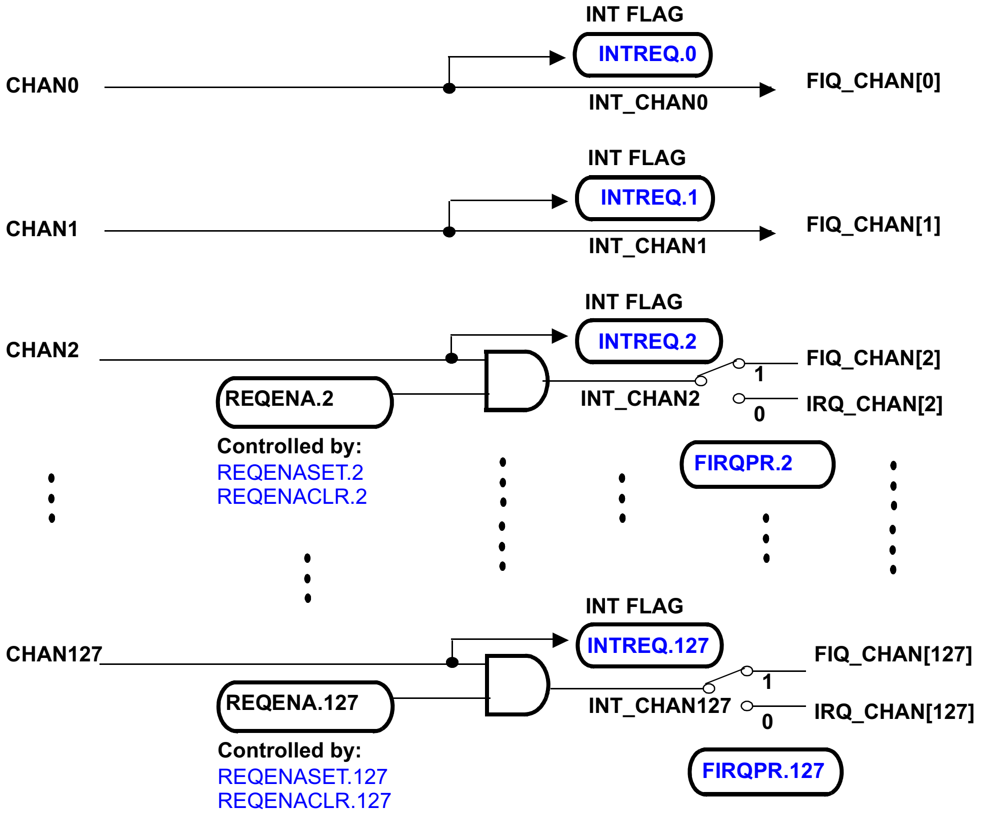
NOTE: CHAN0 and CHAN1 are hard wired to INT_REQ0 and INT_REQ1, so they cannot be remapped.

15.3.2 VIM Input Channel Management

As shown in Figure 15-6, the VIM enables channels on a channel-by-channel basis (in the REQENASET and REQENACLR registers); unused channels may be masked to prevent spurious interrupts.

NOTE: The interrupt ENABLE register does not affect the value of INTREQ.

Figure 15-6. Interrupt Channel Management



By default, interrupt CHAN0 is mapped to ESM (Error Signal Module) high level interrupt and CHAN1 is reserved for other NMI. For safety reasons, these two channels are mapped to FIQ only and can **NOT** be disabled through ENABLE registers.

NOTE: **NMI Channel**

Channel 0 and channel 1 are not maskable by the REQENASET and REQENACLR bit and both channel are routed exclusively to FIQ/NMI request line (FIRQPR0 and FIRQPR1 have no effect).

The VIM prioritizes the received interrupts based upon a programmed prioritization scheme. The VIM can send two interrupt requests to the CPU simultaneously—one IRQ and one FIQ. If both interrupt types are enabled at the CPU level, then the FIQ has greater priority and is handled first. Each interrupt channel, except channel 0 and 1, can be assigned to send either an FIQ or IRQ request to the CPU (in the FIRQPR register).

The VIM provides a default prioritization scheme, which sends the lowest numbered active channel (in each FIQ and IRQ classes) to the CPU. Within the FIQ and IRQ classes of interrupts, the lowest channel has the highest priority interrupt. The channel number is programmable through register CHANMAPx.

After the VIM has generated the vector corresponding to the highest active IRQ, it updates the FIQINDEX or the IRQINDEX register, depending on the class of interrupt. Then, it accesses the interrupt vector table using the vector value to fetch the address of the corresponding ISR. If the request is an FIQ class interrupt, the address read from the interrupt vector table, is written to the FIQVECREG register. If the request is an IRQ class interrupt, the address is written to the IRQVECREG register and put on the VIC port of the CPU (in case of hardware vectored interrupt is enabled).

All of the interrupt registers are updated when a new high priority interrupt line becomes active.

15.4 Interrupt Vector Table (VIM RAM)

Interrupt vector table stores the address of ISRs. During register vectored interrupt and hardware vectored interrupt, VIM accesses the interrupt vector table using the vector value to fetch the address of the corresponding ISR.

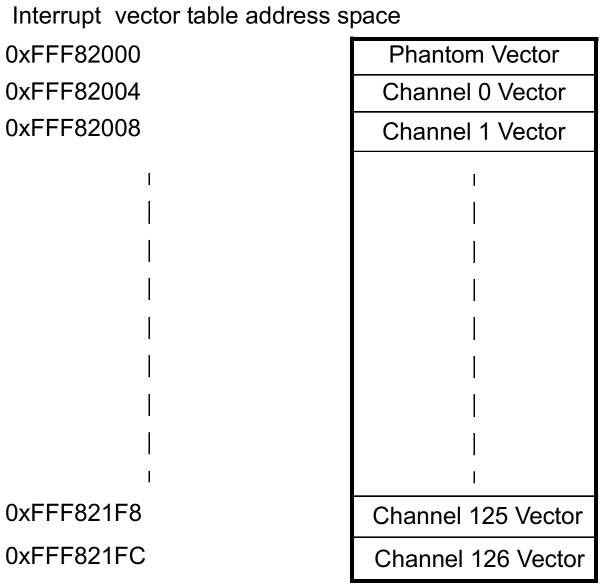
For safety reasons, the interrupt vector table has protection by parity to indicate corruption due to soft errors. The parity scheme is implemented as a continuous background check based on memory access. Section 15.4.1 through Section 15.4.4 describe how parity works in the interrupt vector table.

NOTE: Writes to the interrupt vector table parity flag register (PARFLG) and the interrupt vector table parity control register (PARCTL) are in privilege mode only.

15.4.1 Interrupt Vector Table Operation

The interrupt vector table is organized in 128 words of 32 bits. 32-bit, 16-bit, and 8-bit accesses are supported (when parity is disabled). Figure 15-7 shows the interrupt memory mapping. The table base address is FFF8 2000h.

Figure 15-7. VIM Interrupt Address Memory Map



NOTE: The interrupt vector table only has 128 entries, one phantom vector and 127 interrupt channels. Channel 127 does not have a dedicated vector and shall not be used.

There is one bit of parity per 32-bit ISR address. When a write is performed into the interrupt vector table, the parity is calculated for the 32-bit word and a parity bit is written into the corresponding parity region of interrupt vector table.

NOTE: Only 32-bit write/read access are allowed on interrupt vector table if parity is required. Non 32-bit access might result in parity errors.

When a read occurs from the CPU or VIM, the VIM calculates the parity from the data coming from the interrupt vector table and compares it to the parity stored in the table. The access of the data and the parity is performed in the same clock cycle.

If the parity bit does not match the calculated parity, a parity error is generated and the VIM stores the address of the error in the ADDERR register. The parity flag error (PARFLG) is set.

NOTE: The PARFLG register is only for bypassing the interrupt vector table in case of a parity fault. It should be used only to maintain the interrupt vector table bypassed. The checking of the parity fault should be done in the error signaling module (ESM) module where all parity errors are flagged.

Since the interrupt vector table may have an error, the FBPARERR register will provide to the VIC port, IRQVECREG and FIQVECREG, a fall-back address to an ISR that can restore the interrupt vector table content. The FBPARERR register should be set before initializing the interrupt in the interrupt vector table, to avoid branching to an unpredictable location.

The normal operation is restored when the PARFLG is cleared by the CPU. It is recommended to restore the content of the VIM before clearing the PARFLG.

The parity error signal is forwarded to the ESM.

15.4.2 Enabling and Controlling the VIM Parity

The polarity of VIM parity is controlled by the DEVCR1 register in the system module (address FFFF FFDCh). The parity enable is controlled by the PARCTL register. After reset, the parity is disabled.

Parity checking can be enabled by writing 0xA (1010b) in the PARENA[3:0] bit field of the PARCTL register. The default polarity is odd. The polarity can be changed to even by writing 5 (0101b) in the DEVPARSEL[3:0] bit field of the DEVCR1 register.

15.4.3 Interrupt Vector Table Initialization

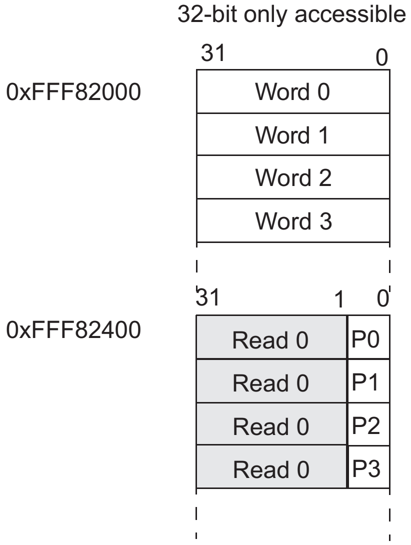
After reset, the interrupt vector table content including the parity bits is not initialized. Therefore, the CPU needs to initialize all of the interrupt addresses into the table, before enabling the corresponding interrupt channel. If parity is required, this initialization should be done after the parity functionality is enabled. In this way, the corresponding parity bit will be automatically updated. This initialization is only required when vectored interrupts are used, index interrupt management does not need the table to be initialized.

15.4.4 Interrupt Vector Table Parity Testing

To test the parity checking mechanism, the parity RAM allows manual insertion of faults. This option is implemented by the test bit in the PARCTL register. Once the bit is set, the parity bits are mapped to 0xFFF82400. After that, user can force faults into the parity bits. Finally, the parity error can be triggered by reading interrupt vector table (not parity bit) from VIM or CPU.

The interrupt vector table parity can also be verified by inserting faults into interrupt vector table. Once the VIM parity is disabled in system module, user can modify interrupt vector table without impacting the parity bit. After user re-enable interrupt vector table parity, the parity error can be triggered by reading interrupt vector table from VIM or CPU.

Figure 15-8. Parity Bit Mapping

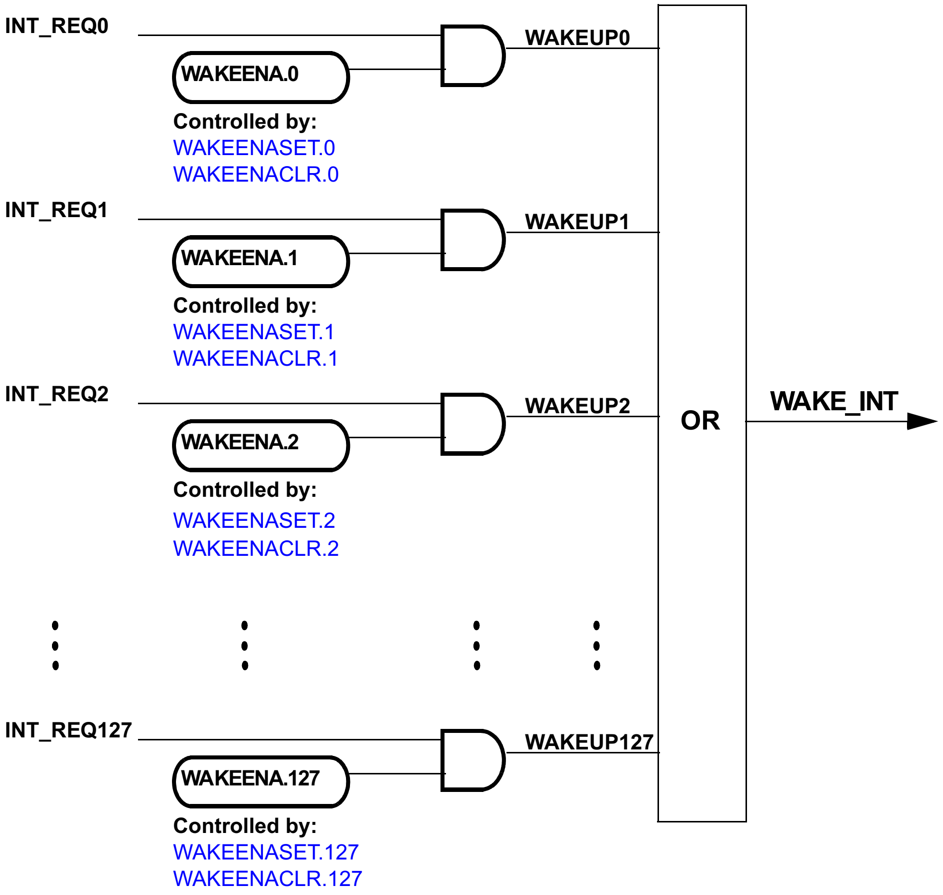


15.5 VIM Wakeup Interrupt

The wakeup interrupts are used to come out of low power mode (LPM). Any interrupt requests can be used to wake up the device. After reset, all interrupt requests are set to wake up from LPM. However, the VIM can mask unwanted interrupt lines for wake-up by using the WAKEENASET and WAKEENACLR register. The value in REQENASET / REQENACLR does NOT impact the wakeup interrupt.

As shown in Figure 15-9, the WAKEENASET and WAKEENACLR registers will enable/disable an interrupt for wake-up from low-power mode. All wake-up interrupts are “ORed” into a single signal WAKE_INT connected to the Global Clock Module.

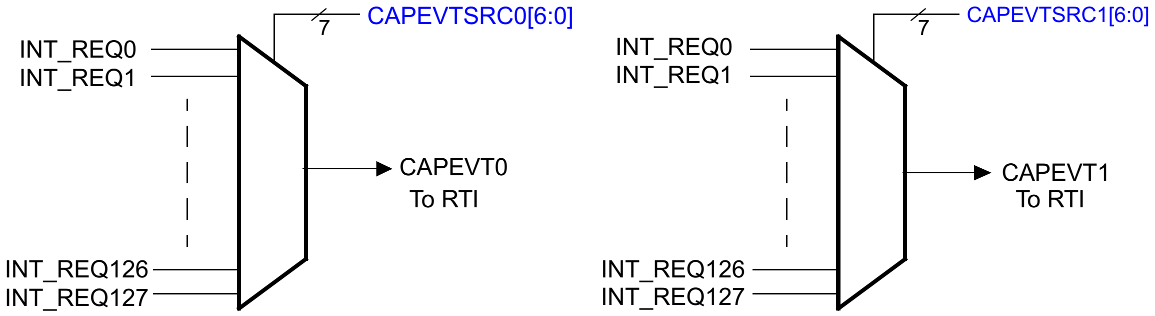
Figure 15-9. Detail of the IRQ Input



15.6 Capture Event Sources

The VIM can select any of the 128 interrupt requests to generate up to two capture events for the real-time interrupt (RTI) module (see Figure 15-10). The value in REQENASET / REQENACLR does NOT impact the capture event. Two registers (Section 15.8.14) are available, one for each capture event source.

Figure 15-10. Capture Event Sources



15.7 Examples

The following sections provide examples about the operation of the VIM.

15.7.1 Examples - Configure CPU to Receive Interrupts

Example 15-1 shows how to set the vector enable (VE) bit in the CP15 R1 register to enable the hardware vector interrupt. Example 15-2 shows how to enable/disable the IRQ and FIQ through CPSR. As a convention, the program who calls these subroutines shall preserve register R1 if needed. Example 15-2 can ONLY run in privileged mode. However, in USER mode, the application software can force the program into software interrupt by instruction ‘SWI’. Then, in the software interrupt service routine, user can write register SPSR, which is the copy of CPSR in this exception mode.

Example 15-1. Enable Hardware Vector Interrupt (IRQ Only)

```
_HW_Vec_Init
    MRC p15 ,#0 ,R1 ,c1 ,c0 ,#0
    ORR R1 ,R1 ,#0x01000000      ; Mask 0-31 bits except bit 24 in Sys
                                 ; Ctrl Reg of CORTEX-R4
    MCR p15 ,#0 ,R1 ,c1 ,c0 ,#0  ; Enable bit 24
    MOV PC, LR
```

Example 15-2. Enable/Disable IRQ/FIQ through CPSR

```
FIQENABLE .equ 0x40
IRQENABLE .equ 0x80
......
_Enable_Fiq
    MRS R1, CPSR
    BIC R1, R1, #FIQENABLE
    MSR CPSR, R1
    MOV PC, LR
......
_Disable_Irq
    MRS R1, CPSR
    ORR R1, R1, #IRQENABLE
    MSR CPSR, R1
    MOV PC, LR
......
_Enable_Irq
    MRS R1, CPSR
    BIC R1, R1, #IRQENABLE
    MSR CPSR, R1
    MOV PC, LR
```

15.7.2 Examples - Register Vector Interrupt and Index Interrupt Handling

Example 15-3 illustrates the configuration for the exception vectors in Register Vector Interrupt handling. After the interrupt is received by the CPU, the CPU branches to 0x18 (IRQ) or 0x1C (FIQ). The instruction placed here should be *LDR PC, [PC,#-0x1B0]*. The pending ISR address is written into the corresponding vector register (IRQVECREG for IRQ, FIQVECREG for FIQ). The CPU reads the content of the register and branches to the ISR.

Example 15-3. Exception Vector Configuration for VIM Vector

```
          .sect ".intvecs"
00000000h b _RESET              ; RESET interrupt
00000004h b _UNDEF_INST_INT     ; UNDEFINED INSTRUCTION interrupt
00000008h b _SW_INT             ; SOFTWARE interrupt
0000000Ch b _ABORT_PREF_INT     ; ABORT (PREFETCH) interrupt
00000010h b _ABORT_DATA_INT     ; ABORT (DATA) interrupt
00000014h b #-8                 ; Reserved
00000018h ldr pc,[pc,#-0x1B0]  ; IRQ interrupt
0000001Ch ldr pc,[pc,#-0x1B0]  ; FIQ interrupt
```

NOTE: Program Counter (PC) always pointers two instructions beyond the current executed instruction. In this case, PC equals to ‘*0x18 or 0x1C* + *0x08*’. The *LDR* instruction load the memory at ‘*PC* - *0x1B0*’, which is ‘*0x18 or 0x1C* + *0x08* - *0x1B0* = 0xFFFFFE70 or 0xFFFFFE74’. These are the address of IRQVECREG and FIQVECREG, which store the pending ISR address.

Example 15-4 shows a fast response to the FIQ interrupt in Index Interrupt and can be applied to a system that has more than one channel assigned as a FIQ. It is built in Index Interrupt compatible with TMS470R1x legacy code.

Example 15-4. How to Respond to FIQ With Short Latency

```
          .sect ".intvecs"         ; Interrupt and exception vector sector
00000000h b _RESET                 ; RESET interrupt
00000004h b _UNDEF_INST_INT        ; UNDEFINED INSTRUCTION interrupt
00000008h b _SW_INT                ; SOFTWARE interrupt
0000000Ch b _ABORT_PREF_INT        ; ABORT (PREFETCH) interrupt
00000010h b _ABORT_DATA_INT        ; ABORT (DATA) interrupt
00000014h b #-8                    ; Reserved
00000018h b _IRQ_ENTRY_0           ; IRQ interrupt
                                   ;*********************************
                                   ; INTERRUPT PROCESSING AREA
                                   ;*********************************
0000001Ch ldrb R8, [PC,#-0x21d]    ; FIQ INTERRUPT ENTRY
                                   ; R8 used to get the FIQ index
                                   ; with address pointer to the
                                   ; first FIQ banked register
00000020h ldr PC, [PC, R8, LSL#2]  ; Branch to the indexed interrupt
                                   ; routine. The prefetch
                                   ; operation causes the PC to be 2
                                   ; words (8 bytes) ahead of the
                                   ; current instruction, so
                                   ; pointing to _INT_TABLE.
00000024h nop                      ; Required due to pipeline.
                                   ;=================================
00000028h _INT_TABLE               ; FIQ INTERRUPT DISPATCH
                                   ;=================================
0000002Ch .word _FIQ_TABLE         ; beginning of FIQ Dispatch
00000030h .word _ISR1              ; dispatch to interrupt routine 1
00000034h .word _ISR2              ; dispatch to interrupt routine 2
          .
          .
```

Another way to improve the FIQ latency is to assign only one channel to the FIQ interrupt and to map the ISR code corresponding to this channel directly starting at 0x1C.

NOTE: When the CPU is in vector-enabled mode, Example 15-3 and Example 15-4 are still valid. The difference is that the CPU will not read from the 0x18 location during IRQ interrupt, but will jump directly to the corresponding ISR routine.

15.8 VIM Registers

Table 15-1 lists the VIM module registers. Each register begins on a word boundary. All registers are 32-bit, 16-bit, and 8-bit accessible for read and write. Write is only possible in privilege mode. The base address of the control registers is FFFF FE00h. The base address of the parity-related VIM registers is FFFF FD00h. The address locations not listed are reserved.

Table 15-1. VIM Control Registers

Offset	Acronym	Register Description	Section
		Parity-related Registers	
ECh	PARFLG	Interrupt Vector Table Parity Flag Register	Section 15.8.1
F0h	PARCTL	Interrupt Vector Table Parity Control Register	Section 15.8.2
F4h	ADDERR	Address Parity Error Register	Section 15.8.3
F8h	FBPARERR	Fall-Back Address Parity Error Register	Section 15.8.4
		Control Registers	
00h	IRQINDEX	IRQ Index Offset Vector Register	Section 15.8.5.1
04h	FIQINDEX	FIQ Index Offset Vector Register	Section 15.8.5.2
10h	FIRQPR0	FIQ/IRQ Program Control Register 0	Section 15.8.6
14h	FIRQPR1	FIQ/IRQ Program Control Register 1	Section 15.8.6
18h	FIRQPR2	FIQ/IRQ Program Control Register 2	Section 15.8.6
1Ch	FIRQPR3	FIQ/IRQ Program Control Register 3	Section 15.8.6
20h	INTREQ0	Pending Interrupt Read Location Register 0	Section 15.8.7
24h	INTREQ1	Pending Interrupt Read Location Register 1	Section 15.8.7
28h	INTREQ2	Pending Interrupt Read Location Register 2	Section 15.8.7
2Ch	INTREQ3	Pending Interrupt Read Location Register 3	Section 15.8.7
30h	REQENASET0	Interrupt Enable Set Register 0	Section 15.8.8
34h	REQENASET1	Interrupt Enable Set Register 1	Section 15.8.8
38h	REQENASET2	Interrupt Enable Set Register 2	Section 15.8.8
3Ch	REQENASET3	Interrupt Enable Set Register 3	Section 15.8.8
40h	REQENACLR0	Interrupt Enable Clear Register 0	Section 15.8.9
44h	REQENACLR1	Interrupt Enable Clear Register 1	Section 15.8.9
48h	REQENACLR2	Interrupt Enable Clear Register 2	Section 15.8.9
4Ch	REQENACLR3	Interrupt Enable Clear Register 3	Section 15.8.9
50h	WAKEENASET0	Wake-up Enable Set Register 0	Section 15.8.10
54h	WAKEENASET1	Wake-up Enable Set Register 1	Section 15.8.10
58h	WAKEENASET2	Wake-up Enable Set Register 2	Section 15.8.10
5Ch	WAKEENASET3	Wake-up Enable Set Register 3	Section 15.8.10
60h	WAKEENACLR0	Wake-up Enable Clear Register 0	Section 15.8.11
64h	WAKEENACLR1	Wake-up Enable Clear Register 1	Section 15.8.11
68h	WAKEENACLR2	Wake-up Enable Clear Register 2	Section 15.8.11
6Ch	WAKEENACLR3	Wake-up Enable Clear Register 3	Section 15.8.11
70h	IRQVECREG	IRQ Interrupt Vector Register	Section 15.8.12
74h	FIQVECREG	FIQ Interrupt Vector Register	Section 15.8.13
78h	CAPEVT	Capture Event Register	Section 15.8.14
80h-FCh	CHANCTRL0 to CHANCTRL31	VIM Interrupt Control Register	Section 15.8.15

15.8.1 Interrupt Vector Table Parity Flag Register (PARFLG)

Figure 15-11 and Table 15-2 describe this register.

Figure 15-11. Interrupt Vector Table Parity Flag Register (PARFLG) [offset = ECh]

31–16
Reserved
R-0

15–1	0
Reserved	PARFLG
R-0	R/WP-0

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

Table 15-2. Interrupt Vector Table Parity Flag Register (PARFLG) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	PARFLG		The PARFLG indicates that a parity error has been found and that the Interrupt Vector Table is bypassed. The resulting vector of any IRQ/FRQ interrupt is then the value contained in the FBPARERR register until this bit has been cleared.
		0	Read: No parity error has occurred. Write: No effect.
		1	Read: A parity error has occurred and the Interrupt Vector Table is bypassed. Write: The PARFLG is cleared and the interrupt vector can be read from the Interrupt Vector Table.

15.8.2 Interrupt Vector Table Parity Control Register (PARCTL)

Figure 15-12. Interrupt Vector Table Parity Control Register (PARCTL) [offset = F0h]

31–16
Reserved
R-0

15–9	8	7–4	3–0
Reserved	TEST	Reserved	PARENA
R-0	R/WP-0	R-0	R/WP-5h

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

Table 15-3. Interrupt Vector Table Parity Control Register (PARCTL) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reads return 0. Writes have no effect.
8	TEST		This bit maps the parity bits into the Interrupt Vector Table frame to make them accessible by the CPU.
		0	Parity bits are not memory mapped.
		1	Parity bits are memory mapped.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	PARENA		VIM parity enable.
		5h	VIM parity is disabled.
		All other values	VIM parity is enabled. **Note: To avoid soft error to disable VIM parity checking, it is recommended to write Ah to enable parity checking.**

15.8.3 Address Parity Error Register (ADDERR)

The address parity error register gives the address of the first parity error location.

NOTE: No computation is needed when reading the complete register to retrieve the address in the Interrupt Vector Table.

This register will never be reset by a power-on reset nor any other reset source.

Figure 15-13. Address Parity Error Register (ADDERR) [offset = F4h]

31–16
Interrupt Vector Table offset
R-FFF8h

15–9	8–2	1–0
Interrupt Vector Table offset	ADDERR	Word offset
R-0010 000b	R-x	R-0

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

Table 15-4. Address Parity Error Register (PARERR) Field Descriptions

Bit	Field	Description
31-9	Interrupt Vector Table offset	Interrupt Vector Table offset. Reads are always FFF8 2xxxh; writes have no effect.
8-2	ADDERR	Address parity error register. This register gives the address of the first encountered parity error since the flag has been clear. Subsequent parity errors will not update this register until the PARFLG register has been cleared. **Note: This register is valid only when PARFLG is set (see Section 15.8.1).**
1-0	Word offset	Word offset. Reads are always 0; writes have no effect.

15.8.4 Fall-Back Address Parity Error Register (FBPARERR)

This register provides a fall-back address to the VIM if a parity error has occurred in the Interrupt Vector Table. Figure 15-14 and Table 15-5 describe this register.

NOTE: This register will never be reset by a power-on reset nor any other reset source.

Figure 15-14. Fall-Back Address Parity Error Register (FBPARERR) [offset = F8h]

31–16
FBPARERR
R/WP-x

15–0
FBPARERR
R/WP-x

LEGEND: R/W = Read/Write; WP = Write in privilege mode only; x = value is indeterminate; -*n* = value after reset

Table 15-5. Fall Back Address Parity Error Register (FBPARERR) Field Descriptions

Bit	Field	Value	Description
31-0	FBPARERR	0-FFFF FFFFh	Fall back address parity error. This register is used by the VIM if the Interrupt Vector Table has been corrupted. The contents of the IRQVECREG and FIQVECREG registers will reflect the value programmed in FBPARERR. The value provided to the VIC port will also reflect FBPARERR until the PARFLG register has been cleared. This register provides the address of the ISR that will restore the integrity of the Interrupt Vector Table.

15.8.5 VIM Offset Vector Registers

The VIM offset register provides the user with the numerical index value that represents the pending interrupt with the highest precedence. The register IRQINDEX holds the index to the highest priority IRQ interrupt; the register FIQINDEX holds the index to the highest priority FIQ interrupt. The index can be used to locate the interrupt routine in a dispatch table, as shown in Table 15-6.

Table 15-6. Interrupt Dispatch

IRQINDEX / FIQINDEX Register Bit Field	Highest Priority Pending Interrupt Enabled
0x00	No interrupt
0x01	Channel 0
:	:
0x7F	Channel 126
0x80	Channel 127

NOTE: Channel 127 has no dedicated interrupt vector table entry. Therefore, Channel 127 shall NOT be used in application.

The VIM offset registers are read only. They are updated continuously by the VIM. When an interrupt is serviced, the offset vectors show the index for the next highest pending interrupt or 0x0 if no interrupt is pending.

15.8.5.1 IRQ Index Offset Vector Register (IRQINDEX)

The IRQ offset register provides the user with the numerical index value that represents the pending IRQ interrupt with the highest priority. Figure 15-15 and Table 15-7 describe this register.

Figure 15-15. IRQ Index Offset Vector Register (IRQINDEX) [offset = 00h]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		IRQINDEX	
R-0		R-0	

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

Table 15-7. IRQ Index Offset Vector Register (IRQINDEX) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	IRQINDEX	0-FFh	IRQ index vector. The least-significant bits represent the index of the IRQ pending interrupt with the highest precedence, as shown in Table 15-6. When no interrupts are pending, the least significant byte of IRQINDEX is 0. **Note:** A read of register IRQINDEX or IRQVECREG will cause IRQINDEX / IRQVECREG to reflect the index/ISR address for the next highest-priority pending IRQ interrupt. In case there is no other interrupt pending, the IRQINDEX will read 0x00 and the IRQVECREG register will read the phantom interrupt address.

15.8.5.2 FIQ Index Offset Vector Registers (FIQINDEX)

The FIQINDEX register provides the user with a numerical index value that represents the pending FIQ interrupt with the highest priority. Figure 15-16 and Table 15-8 describe this register.

Figure 15-16. FIQ Index Offset Vector Register (FIQINDEX) [offset = 04h]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		FIQINDEX	
R-0		R-0	

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

Table 15-8. FIQ Index Offset Vector Register (FIQINDEX) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	FIQINDEX	0-FFh	FIQ index offset vector. The least-significant bits represent the index of the FIQ pending interrupt with the highest precedence, as shown in Table 15-6. When no interrupts are pending, the least significant byte of FIQINDEX is 0x00. **Note:** A read of register FIQINDEX or FIQVECREG will cause FIQINDEX / FIQVECREG to reflect the index/ISR address for the next highest-priority pending FIQ interrupt. In case there is no other interrupt pending, the FIQINDEX will read 0x00 and the FIQVECREG register will read the phantom interrupt address.

15.8.6 FIQ/IRQ Program Control Registers (FIRQPR[0:3])

The FIQ/IRQ program control registers (FIRQPR[0]) determine whether a given interrupt request will be either FIQ or IRQ. Figure 15-17, Figure 15-18, Figure 15-19, Figure 15-20 and Table 15-9 describe these registers.

NOTE: Channel 0 and 1 are FIQ only, not impacted by this register.

Figure 15-17. FIQ/IRQ Program Control Register 0 (FIRQPR0) [offset = 10h]

31–16
FIRQPR0[31:16]
R/WP-0

15–2	1–0
FIRQPR0[15:2]	Reserved
R/WP-0	R-3h

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

Figure 15-18. FIQ/IRQ Program Control Register 1 (FIRQPR1) [offset = 14h]

31–16
FIRQPR1[63:48]
R/WP-0

15–0
FIRQPR1[47:32]
R/WP-0

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

Figure 15-19. FIQ/IRQ Program Control Register 2 (FIRQPR2) [offset = 18h]

31–16
FIRQPR2[95:80]
R/WP-0

15–0
FIRQPR2[79:64]
R/WP-0

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

Figure 15-20. FIQ/IRQ Program Control Register 3 (FIRQPR3) [offset = 1Ch]

31–16
FIRQPR3[127:112]
R/WP-0

15–0
FIRQPR3[111:96]
R/WP-0

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

Table 15-9. FIQ/IRQ Program Control Registers (FIRQPR) Field Descriptions

Bit	Field	Value	Description
127-2	FIRQPRx[127:2]		FIQ/IRQ program control bits. These bits determine whether an interrupt request from a peripheral is of type FIQ or IRQ. Bit FIRQPRx[127:2] corresponds to request channel[127:2].
		0	Interrupt request is of IRQ type.
		1	Interrupt request is of FIQ type.
1-0	Reserved	3h	Read only. Writes have no effect.

15.8.7 Pending Interrupt Read Location Registers (INTREQ[0:3])

The pending interrupt register gives the pending interrupt requests. The register is updated every vbus clock cycle. Figure 15-21, Figure 15-22, Figure 15-23, Figure 15-24 and Table 15-10 describe this register.

Figure 15-21. Pending Interrupt Read Location Register 0 (INTREQ0) [offset = 20h]

31	16
INTREQ0[31:16]	
R/W1CP-0	

15	0
INTREQ0[15:0]	
R/W1CP-0	

LEGEND: R/W = Read/Write; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset

Figure 15-22. Pending Interrupt Read Location Register 1 (INTREQ1) [offset = 24h]

31	16
INTREQ1[63:48]	
R/W1CP-0	

15	0
INTREQ1[47:32]	
R/W1CP-0	

LEGEND: R/W = Read/Write; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset

Figure 15-23. Pending Interrupt Read Location Register 2 (INTREQ2) [offset = 28h]

31	16
INTREQ2[95:80]	
R/W1CP-0	

15	0
INTREQ2[79:64]	
R/W1CP-0	

LEGEND: R/W = Read/Write; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset

Figure 15-24. Pending Interrupt Read Location Register 3 (INTREQ3) [offset = 2Ch]

31	16
INTREQ3[127:112]	
R/W1CP-0	

15	0
INTREQ3[111:96]	
R/W1CP-0	

LEGEND: R/W = Read/Write; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset

Table 15-10. Pending Interrupt Read Location Registers (INTREQ) Field Descriptions

Bit	Field	Value	Description
127-0	INTREQx[127:0]		Pending interrupt bits. These bits determine whether an interrupt request is pending for the request channel between 0 and 127. The interrupt ENABLE register does not affect the value of the interrupt pending bit. Bit INTREQx[127:0] corresponds to request channel[127:0].
			User and Privilege Mode read:
		0	No interrupt event has occurred.
		1	An interrupt is pending.
			Privilege Mode write only:
		0	Writing 0 has no effect.
		1	Clears the "interrupt pending" status flag. This write-clear functionality is intended to allow clearing those interrupts which have been signaled to VIM before enabling the interrupt channel, if they are undesired.

15.8.8 Interrupt Enable Set Registers (REQENASET[0:3])

The interrupt register enable selectively enables individual request channels. Figure 15-25, Figure 15-26, Figure 15-27, Figure 15-28 and Table 15-11 describe these registers.

NOTE: Channel 0 and 1 are always enabled, not impacted by this register.

Figure 15-25. Interrupt Enable Set Register 0 (REQENASET0) [offset = 30h]

31	16
REQENASET0[31:16]	
R/WP-0	

15	2	1	0
REQENASET0[15:2]		Reserved	
R/WP-0		R-3h	

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

Figure 15-26. Interrupt Enable Set Register 1 (REQENASET1) [offset = 34h]

31	16
REQENASET1[63:48]	
R/WP-0	

15	0
REQENASET1[47:32]	
R/WP-0	

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

Figure 15-27. Interrupt Enable Set Register 2 (REQENASET2) [offset = 38h]

31	16
REQENASET295:80]	
R/WP-0	

15	0
REQENASET2[79:64]	
R/WP-0	

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

Figure 15-28. Interrupt Enable Set Register 3 (REQENASET3) [offset = 3Ch]

31	16
REQENASET3127:112]	
R/WP-0	

15	0
REQENASET3[111:96]	
R/WP-0	

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

Table 15-11. Interrupt Enable Set Registers (REQENASET) Field Descriptions

Bit	Field	Value	Description
127-2	REQENASETx[127:2]		Request enable set bits. This vector determines whether the interrupt request channel is enabled. Bit REQENASETx[127:2] corresponds to request channel[127:2].
		0	Read: Interrupt request channel is disabled. Write: No effect.
		1	Read or Write: The interrupt request channel is enabled.
1-0	Reserved	3h	Read only. Writes have no effect.

15.8.9 Interrupt Enable Clear Registers (REQENACLR[0:3])

The interrupt register enable selectively disables individual request channels. Figure 15-29, Figure 15-30, Figure 15-31, Figure 15-32 and Table 15-12 describe these registers.

NOTE: Channel 0 and 1 are always enabled, not impacted by this register.

Figure 15-29. Interrupt Enable Clear Register 0 (REQENACLR0) [offset = 40h]

31	16
REQENACLR0[31:16]	
R/WP-0	

15	2	1	0
REQENACLR0[15:2]		Reserved	
R/WP-0		R-3h	

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

Figure 15-30. Interrupt Enable Clear Register 1 (REQENACLR1) [offset = 44h]

31	16
REQENACLR1[63:48]	
R/WP-0	

15	0
REQENACLR1[47:32]	
R/WP-0	

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

Figure 15-31. Interrupt Enable Clear Register 2 (REQENACLR2) [offset = 48h]

31	16
REQENACLR2[95:80]	
R/WP-0	

15	0
REQENACLR2[79:64]	
R/WP-0	

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

Figure 15-32. Interrupt Enable Clear Register 3 (REQENACLR3) [offset = 4Ch]

31	16
REQENACLR3[127:112]	
R/WP-0	

15	0
REQENACLR3[111:96]	
R/WP-0	

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

Table 15-12. Interrupt Enable Clear Registers (REQENACLR) Field Descriptions

Bit	Field	Value	Description
127-2	REQENACLRx[127:2]		Request enable clear bits. This vector determines whether the interrupt request channel is enabled. Bit REQENACLRx[127:2] corresponds to request channel[127:2].
		0	Read: Interrupt request channel is disabled. Write: No effect.
		1	Read: The interrupt request channel is enabled. Write: The interrupt request channel is disabled.
1-0	Reserved	3h	Read only. Writes have no effect.

15.8.10 Wake-Up Enable Set Registers (WAKEENASET[0:3])

The wake-up enable registers selectively enables individual wake-up interrupt request lines. Figure 15-33, Figure 15-34, Figure 15-35, Figure 15-36 and Table 15-13 describe these registers.

Figure 15-33. Wake-Up Enable Set Register 0 (WAKEENASET0) [offset = 50h]

31	16
WAKEENASET0[31:16]	
R/WP-FFFFh	

15	0
WAKEENASET0[15:0]	
R/WP-FFFFh	

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

Figure 15-34. Wake-Up Enable Set Register 1 (WAKEENASET1) [offset = 54h]

31	16
WAKEENASET1[63:48]	
R/WP-FFFFh	

15	0
WAKEENASET1[47:32]	
R/WP-FFFFh	

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

Figure 15-35. Wake-Up Enable Set Register 2 (WAKEENASET2) [offset = 58h]

31	16
WAKEENASET2[95:80]	
R/WP-FFFFh	

15	0
WAKEENASET2[79:64]	
R/WP-FFFFh	

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

Figure 15-36. Wake-Up Enable Set Register 3 (WAKEENASET3) [offset = 5Ch]

31	16
WAKEENASET3[127:112]	
R/WP-FFFFh	

15	0
WAKEENASET3[111:96]	
R/WP-FFFFh	

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

Table 15-13. Wake-Up Enable Set Registers (WAKEENASET) Field Descriptions

Bit	Field	Value	Description
127-0	WAKEENASETx[127:0]		Wake-up enable set bits. This vector determines whether the wake-up interrupt line is enabled. Bit WAKEENASETx[127:0] corresponds to interrupt request channel[127:0].
		0	Read: Interrupt request channel is disabled. Write: No effect.
		1	Read or Write: The interrupt request channel is enabled.

15.8.11 Wake-Up Enable Clear Registers (WAKEENACLR[0:3])

The wake-up enable register selectively disables individual wake-up interrupt request lines. Figure 15-37, Figure 15-38, Figure 15-39, Figure 15-40 and Table 15-14 describe these registers.

Figure 15-37. Wake-Up Enable Clear Register 0 (WAKEENACLR0) [offset = 60h]

31	16
WAKEENACLR0[31:16]	
R/WP-FFFFh	
15	0
WAKEENACLR0[15:0]	
R/WP-FFFFh	

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

Figure 15-38. Wake-Up Enable Clear Register 1 (WAKEENACLR1) [offset = 64h]

31	16
WAKEENACLR1[63:48]	
R/WP-FFFFh	
15	0
WAKEENACLR1[47:32]	
R/WP-FFFFh	

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

Figure 15-39. Wake-Up Enable Clear Register 2 (WAKEENACLR2) [offset = 68h]

31	16
WAKEENACLR2[95:80]	
R/WP-FFFFh	
15	0
WAKEENACLR2[79:64]	
R/WP-FFFFh	

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

Figure 15-40. Wake-Up Enable Clear Register 3 (WAKEENACLR3) [offset = 6Ch]

31	16
WAKEENACLR3[127:112]	
R/WP-FFFFh	
15	0
WAKEENACLR3[111:96]	
R/WP-FFFFh	

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

Table 15-14. Wake-Up Enable Clear Registers (WAKEENACLR) Field Descriptions

Bit	Field	Value	Description
127-0	WAKEENACLRx[127:0]		Wake-up enable clear bits. This vector determines whether the wake-up interrupt line is enabled. Bit WAKEENACLRx[127:0] corresponds to interrupt request channel[127:0].
		0	Read: Wake-up interrupt channel is disabled. Write: No effect.
		1	Read: The wake-up interrupt channel is enabled. Write: The wake-up interrupt channel is disabled.

15.8.12 IRQ Interrupt Vector Register (IRQVECREG)

The interrupt vector register gives the address of the enabled and active IRQ interrupt. Figure 15-41 and Table 15-15 describe these registers.

Figure 15-41. IRQ Interrupt Vector Register (IRQVECREG) [offset = 70h]

31 … 0
IRQVECREG
R-0

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

Table 15-15. IRQ Interrupt Vector Register (IRQVECREG) Field Descriptions

Bit	Field	Value	Description
31-0	IRQVECREG	From Section 15.4	IRQ interrupt vector register. This vector gives the address of the ISR with the highest pending IRQ request. The CPU reads the address and branches to this location.

15.8.13 FIQ Interrupt Vector Register (FIQVECREG)

The interrupt vector register gives the address of the enabled and active FIQ interrupt. Figure 15-42 and Table 15-16 describe these registers.

Figure 15-42. IRQ Interrupt Vector Register (FIQVECREG) [offset = 74h]

31 … 0
FIQVECREG
R-0

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

Table 15-16. FIQ Interrupt Vector Register (FIQVECREG) Field Descriptions

Bit	Field	Value	Description
31-0	FIQVECREG	From Section 15.4	FIQ interrupt vector register. This vector gives the address of the ISR with the highest pending FIQ request. The CPU reads the address and branches to this location.

15.8.14 Capture Event Register (CAPEVT)

Figure 15-43 and Table 15-17 describe this register.

Figure 15-43. Capture Event Register (CAPEVT) [offset = 78h]

31 23	22 16
Reserved	CAPEVTSRC1
R-U	R/W-0

15 7	6 0
Reserved	CAPEVTSRC0
R-U	R/W-0

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

Table 15-17. Capture Event Register (CAPEVT) Field Descriptions

Bit	Field	Value	Description
31-23	Reserved	0	Reads are indeterminate and writes have no effect.
22-16	CAPEVTSRC1		Capture event source 1 mapping control. These bits determine which interrupt request maps to the capture event source 1 of the RTI:
		0	Interrupt request 0.
		1h	Interrupt request 1.
		:	:
		7Fh	Interrupt request 127.
15-7	Reserved	0	Reads are indeterminate and writes have no effect.
6-0	CAPEVTSRC0		Capture event source 0 mapping control. These bits determine which interrupt request maps to the capture event source 0 of the RTI:
		0	Interrupt request 0.
		1h	Interrupt request 1.
		:	:
		7Fh	Interrupt request 127.

15.8.15 VIM Interrupt Control Registers (CHANCTRL[0:31])

Thirty-two interrupt control registers control the 128 interrupt channels of the VIM. Each register controls four interrupt channels: each of them is indexed from 0 to 127. Table 15-18 shows the organization of all the registers and the reset value of each. Each four fields of the register has been named with a generic index that refers to the detailed register organization. Figure 15-44 and Table 15-19 describe these registers.

Table 15-18. Interrupt Control Registers Organization

Address	Register Acronym	Register Field 31:24 CHANMAPx$_0$	Register Field 23:16 CHANMAPx$_1$	Register Field 15:8 CHANMAPx$_2$	Register Field 7:0 CHANMAPx$_3$	Reset Value
FFFF FE80h	CHANCTRL0	CHANMAP0	CHANMAP1	CHANMAP2	CHANMAP3	0001 0203h
FFFF FE84h	CHANCTRL1	CHANMAP4	CHANMAP5	CHANMAP6	CHANMAP7	0405 0607h
:	:	:	:	:	:	:
FFFF FEF8h	CHANCTRL30	CHANMAP120	CHANMAP121	CHANMAP122	CHANMAP123	7879 7A7Bh
FFFF FEFCh	CHANCTRL31	CHANMAP124	CHANMAP125	CHANMAP126	CHANMAP127	7C7D 7E7Fh

NOTE: CHANMAP0 and CHANMAP1 are not programable. CHAN0 and CHAN1 are hard wired to INT_REQ0 and INT_REQ1.

Do NOT write any value other than 0x7F to CHANMAP127. Channel 127 is reserved because no interrupt vector table entry supports this channel.

Figure 15-44. Interrupt Control Registers (CHANCTRL[0:31]) [offset = 80h-FCh]

31	30 24	23	22 16
Rsvd	CHANMAPx$_0$	Rsvd	CHANMAPx$_1$
R-U	R/W-n	R-U	R/W-n

15	14 8	7	6 0
Rsvd	CHANMAPx$_2$	Rsvd	CHANMAPx$_3$
R-U	R/W-n	R-U	R/W-n

LEGEND: R/W = Read/Write; R = Read only; U = value is undefined; *-n* = value after reset (see Table 15-18)

Table 15-19. Interrupt Control Registers (CHANCTL) Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reads are indeterminate and writes have no effect.
30-24	CHANMAPx$_0$		CHANMAPx$_0$(6-0). Interrupt CHANx$_0$ mapping control. These bits determine which interrupt request the priority channel CHANx$_0$ maps to:
		0	Read: Interrupt request 0 maps to channel priority CHANx$_0$. Write: The default value of this bit after reset is given in Table 15-18 . The channel priority CHANx$_0$ is set with the interrupt request.
		1h :	Read: Interrupt request 1 maps to channel priority CHANx$_0$. Write: The default value of this bit after reset is given in Table 15-18. The channel priority CHANx$_0$ is set with the interrupt request. :
		7Fh	Read: Interrupt request 127 maps to channel priority CHANx$_0$. Write: The default value of this bit after reset is given in Table 15-18. The channel priority CHANx$_0$ is set with the interrupt request.
23	Reserved	0	Reads are indeterminate and writes have no effect.

Table 15-19. Interrupt Control Registers (CHANCTL) Field Descriptions (continued)

Bit	Field	Value	Description
22-16	CHANMAPx$_1$		CHANMAPx$_1$(6-0). Interrupt CHANx$_1$ mapping control. These bits determine which interrupt request the priority channel CHANx$_1$ maps to:
		0	Read: Interrupt request 0 maps to channel priority CHANx$_1$.
			Write: The default value of this bit after reset is given in Table 15-18. The channel priority CHANx$_1$ is set with the interrupt request.
		1h	Read: Interrupt request 1 maps to channel priority CHANx$_1$.
			Write: The default value of this bit after reset is given in Table 15-18. The channel priority CHANx$_1$ is set with the interrupt request.
		:	:
		7Fh	Read: Interrupt request 127 maps to channel priority CHANx$_1$.
			Write: The default value of this bit after reset is given in Table 15-18. The channel priority CHANx$_1$ is set with the interrupt request.
15	Reserved	0	Reads are indeterminate and writes have no effect.
14-8	CHANMAPx$_2$		CHANMAPx$_2$(6-0). Interrupt CHANx$_2$ mapping control. These bits determine which interrupt request the priority channel CHANx$_2$ maps to:
		0	Read: Interrupt request 0 maps to channel priority CHANx$_2$.
			Write: The default value of this bit after reset is given in Table 15-18. The channel priority CHANx$_2$ is set with the interrupt request.
		1h	Read: Interrupt request 1 maps to channel priority CHANx$_2$.
			Write: The default value of this bit after reset is given in Table 15-18. The channel priority CHANx$_2$ is set with the interrupt request.
		:	:
		7Fh	Read: Interrupt request 127 maps to channel priority CHANx$_2$.
			Write: The default value of this bit after reset is given in Table 15-18. The channel priority CHANx$_2$ is set with the interrupt request.
7	Reserved	0	Reads are indeterminate and writes have no effect.
6-0	CHANMAPx$_3$		CHANMAPx$_3$(6-0). Interrupt CHANx$_3$ mapping control. These bits determine which interrupt request the priority channel CHANx$_3$ maps to:
		0	Read: Interrupt request 0 maps to channel priority CHANx$_3$.
			Write: The default value of this bit after reset is given in Table 15-18. The channel priority CHANx$_3$ is set with the interrupt request.
		1h	Read: Interrupt request 1 maps to channel priority CHANx$_3$.
			Write: The default value of this bit after reset is given in Table 15-18. The channel priority CHANx$_3$ is set with the interrupt request.
		:	:
		7Fh	Read: Interrupt request 127 maps to channel priority CHANx$_3$.
			Write: The default value of this bit after reset is given in Table 15-18. The channel priority CHANx$_3$ is set with the interrupt request.

Direct Memory Access Controller (DMA) Module

This chapter describes the direct memory access (DMA) controller.

Topic **Page**

16.1 Overview 575
16.2 Module Operation 576
16.3 Control Registers and Control Packets 595

16.1 Overview

The DMA controller is used to transfer data between two locations in the memory map in the background of CPU operations. Typically, the DMA is used to:

- Transfer blocks of data between external and internal data memories
- Restructure portions of internal data memory
- Continually service a peripheral
- Page program sections to internal program memory

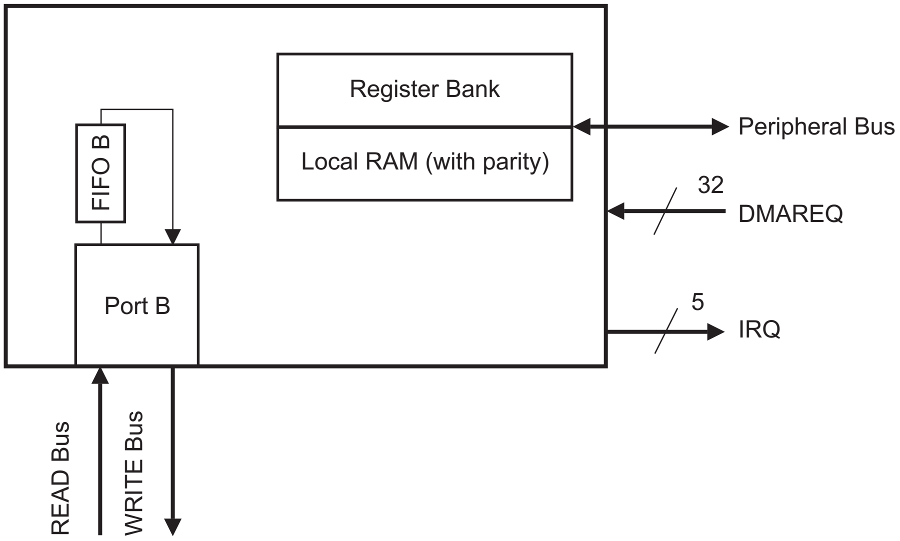
16.1.1 Main Features

- CPU independent data transfer
- One master port - PortB (64-bits wide) that interfaces microcontrollers Memory System.
- FIFO buffer (4 entries deep and each 64-bits wide)
- Channel control information is stored in RAM protected by parity
- 16 channels with individual enable
- Channel chaining capability
- 32 peripheral DMA requests
- Hardware and Software DMA requests
- 8-, 16-, 32-, or 64-bit transactions supported
- Multiple addressing modes for source/destination (fixed, increment, offset)
- Auto-initiation
- Power-management mode
- Memory Protection for the address range DMA can access with four configurable memory regions

16.1.1.1 Block Diagram

Figure 16-1 gives a detailed view of the DMA internal architecture. DMA data read and write access happens through Port B. FIFO B is 4 levels deep and 64-bits wide. 32 DMA requests go into the DMA that can trigger DMA transfers. Five interrupt request lines go out of the DMA to signal that a certain transfer status is reached. Register banks hold the memory-mapped DMA configuration registers. Local RAM consists of DMA control packets and is secured by parity. All the programming / configuration of the DMA controller is done via the Peripheral bus.

Figure 16-1. DMA Block Diagram



16.2 Module Operation

The DMA acts as an independent master in the platform architecture. All DMA memory and register accesses are performed in user mode. If the DMA writes to registers that are only accessible in privileged mode, the write will not be performed.

The DMA registers and its local RAM can only be accessed in privilege mode. Therefore, it is not possible for the DMA to reprogram itself.

16.2.1 Memory Space

The DMA controller makes no distinction between program memory and data memory. The DMA controller can transfer to and from any space within the 4 gigabyte physical address map, by programming the absolute address for the source and destination in the control packet. Control packets store the transfer information such as source address, destination address, transfer count and control attributes for each channel.

16.2.2 DMA Data Access

The DMA controller refers to data in three levels of granularity:

- **Element:** Depending on the programmed data type, an 8-bit, 16-bit, 32-bit, or a 64-bit value. The type can be individually selected for the source (read) and destination (write). See Figure 16-2 and Figure 16-3 for an example of the use of elements. An element transfer cannot be interrupted.
- **Frame:** One or more elements to be transferred as a unit. A frame transfer can be interrupted between element transfers. See Figure 16-2 for an example. Use a frame size of one and frame transfer trigger source for transfers of one element per request.
- **Block:** One or more frames to be transferred as a unit. Each channel can transfer one block of data (once or multiple times). See Figure 16-3 for an example.

Figure 16-2. Example of a DMA Transfer Using Frame Trigger Source

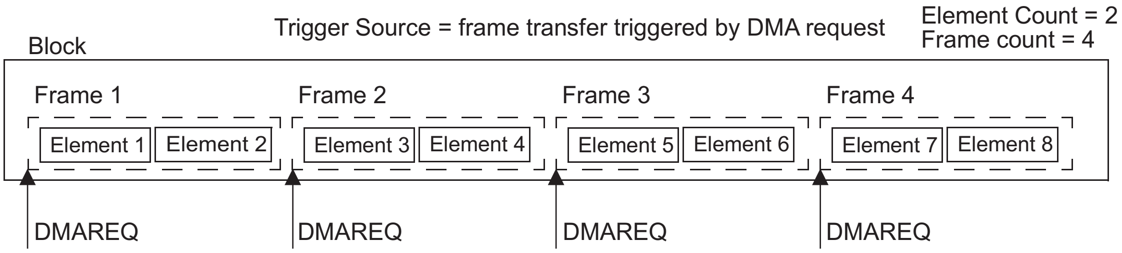
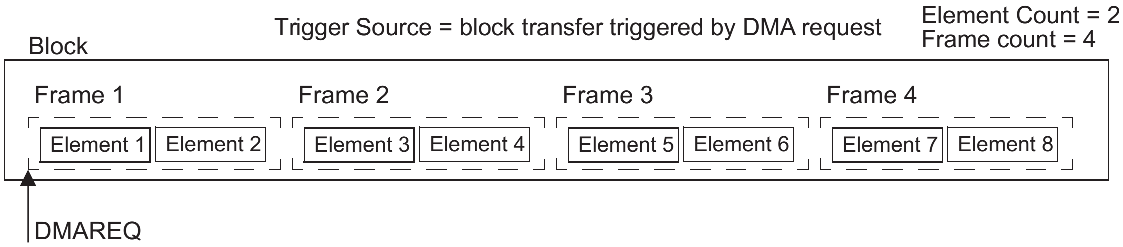


Figure 16-3. Example of a DMA Transfer Using Block Trigger Source



16.2.3 Addressing Modes

There are three addressing modes supported by the DMA controller that can be setup independent for the source and the destination address:

- Constant -- source and/or destination addresses do not change.
- Post incremented -- source and/or destination address are post-incremented by the element size.
- Indexed -- source and/or destination address is post-incremented as defined in the Element Index Offset Register (Section 16.3.2.5) and the Frame Index Offset Register (Section 16.3.2.6).

An unaligned address with respect to the element size is not supported.

16.2.4 DMA Channel Control Packets

There are a total of 16 control packets. Each control packet is associated with a channel in a fixed order. For example, control packet 0 stores channel information for channel 0. The DMA requests can be mapped to the individual channels as described in Section 16.2.7. The mapping scheme between DMA requests and channels is shown in Figure 16-4. Each control packet contains nine fields. The first six fields compose the primary control packet and are programmable during DMA setup. The last three fields compose working control packet and are only readable by the CPU. The working control packets are used to support auto-initiation. The organization of control packets is shown in Figure 16-5.

The primary control packet contains channel information such as source address, destination address, transfer count, element/frame offset value and channel configuration. Source address, destination address and transfer count also have their respective working images. The three fields of working images compose a working control packet and are not accessible to the CPU in write access.

The first time a DMA channel is selected for a transaction, the following process occurs:

1. The primary control packet is first read by the DMA state machine.
2. Once the channel is arbitrated, the current source address, destination address and transfer count are then copied to their respective working images.
3. When the channel is serviced again by the DMA, the state machine will read both the primary control packet and the working control packet to continue the DMA transaction until the end of an entire block transfer.

When the same channel is requested again, the state machine will start again by reading only the primary control packet and then continue the same process described above. The user software need not set up control packets again because the contents of the primary control packet were never lost. The working images of the control packets are reducing the software overhead and interaction with the DMA module to a minimum.

NOTE: Changing the contents of a channel control packet will clear the corresponding pending bit (Section 16.3.1.2) if the channel has a pending status. If the control packet of an active channel (as indicated in Section 16.3.1.3) is changed, then the channel will stop immediately at an arbitration boundary. When the same channel is triggered again, it will begin with the new control packet information.

16.2.4.1 Initial Source Address

This field stores the absolute 32-bit source address of the DMA transfer.

16.2.4.2 Initial Destination Address

This field stores the absolute 32-bit destination address of the DMA transfer.

Figure 16-4. DMA Request Mapping and Control Packet Organization

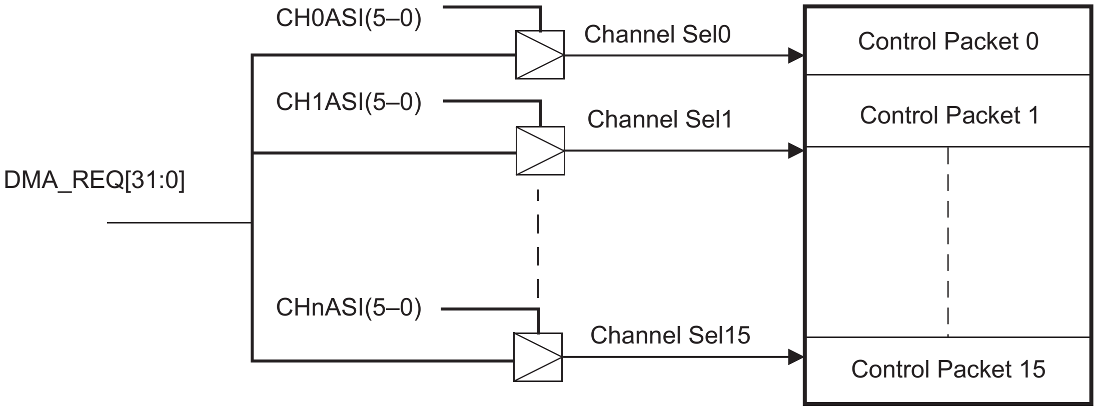
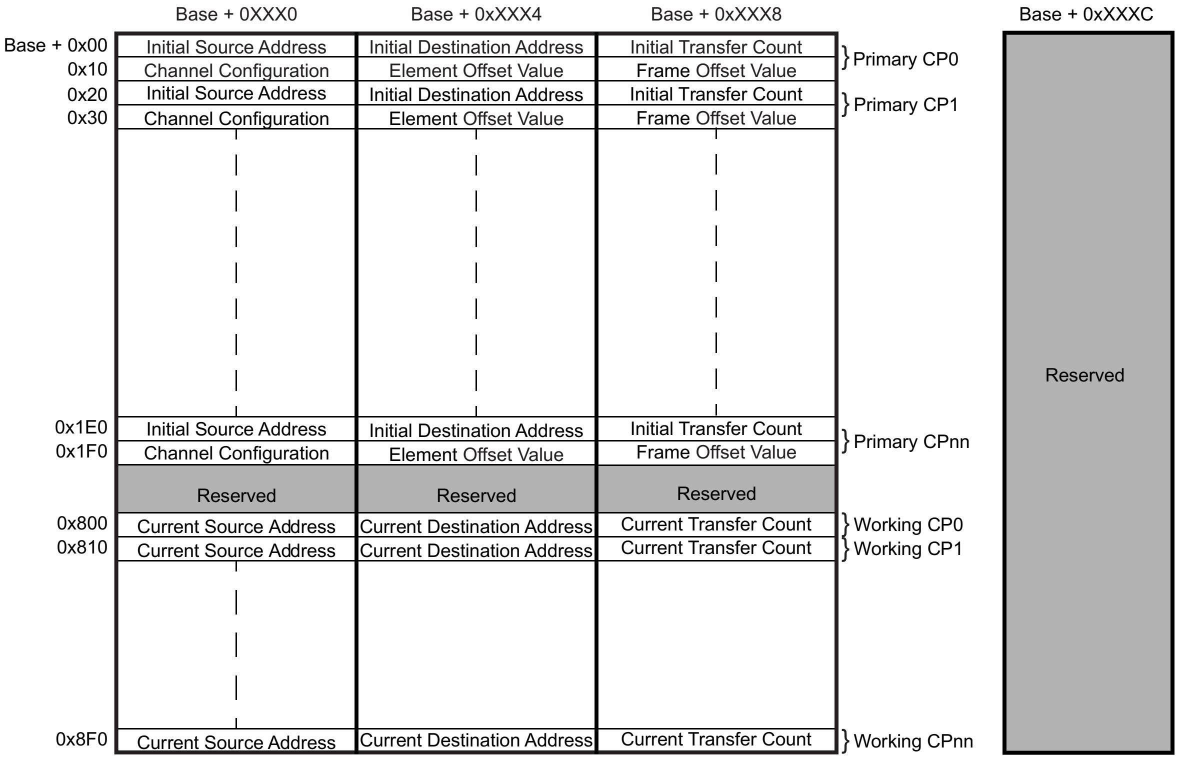


Figure 16-5. Control Packet Organization and Memory Map



	Base + 0XXX0	Base + 0xXXX4	Base + 0xXXX8		Base + 0xXXXC
Base + 0x00	Initial Source Address	Initial Destination Address	Initial Transfer Count	} Primary CP0	Reserved
0x10	Channel Configuration	Element Offset Value	Frame Offset Value		
0x20	Initial Source Address	Initial Destination Address	Initial Transfer Count	} Primary CP1	
0x30	Channel Configuration	Element Offset Value	Frame Offset Value		
0x1E0	Initial Source Address	Initial Destination Address	Initial Transfer Count	} Primary CPnn	
0x1F0	Channel Configuration	Element Offset Value	Frame Offset Value		
	Reserved	Reserved	Reserved		
0x800	Current Source Address	Current Destination Address	Current Transfer Count	} Working CP0	
0x810	Current Source Address	Current Destination Address	Current Transfer Count	} Working CP1	
0x8F0	Current Source Address	Current Destination Address	Current Transfer Count	} Working CPnn	

16.2.4.3 Initial Transfer Count

The transfer count field is composed of two parts. The frame transfer count value and the element transfer count value. Each count value is 13 bits wide. As a Single Block transfer maximum of 512 Mbytes of data can be transferred. Element count and frame count are programmed according to the source data structure.

The total transfer size is calculated as:

$$T_{sz} = E_{rsz} \bullet E_{tc} \bullet F_{tc} \qquad (26)$$

where

T_{sz} = Total Transfer Size

E_{rsz} = Read Element Size

E_{tc} = Element Transfer Count

F_{tc} = Frame Transfer Count

NOTE: A zero element count with a non-zero frame count or a non-zero element count with a zero frame count are all considered as zero total transfer count. No DMA transaction is initiated with any of the counters set to 0.

16.2.4.4 Channel Configuration Word

The channel configuration defines the following individual parameters

- Read element size
- Write element size
- Trigger type (frame or block)
- Addressing mode for source
- Addressing mode for destination
- Auto-initiation mode
- Next control packet to be triggered at control packet finish (Channel Chaining)

16.2.4.5 Element/Frame Offset Value

There are 4 offset values that allow the creation of different types of buffers in RAM and address registers in a structured manner: an element offset value for source and destination and a frame offset value for source and destination.

The element offset value for source and/or destination defines the offset to be added after each element transfer to the source and/or destination address. The frame offset value for source and/or destination defines the offset to be added to the source and/or destination address after the element count reaches zero. The element and frame offset values must be defined in terms of the number of bytes of offset. The DMA controller does not adjust the element/frame index number according to the element size. An index of 2 means *increment the address by 2* and not by 16 when the element size is 64 bits.

16.2.4.6 Current Source Address

The current source address field contains the current working source address during a DMA transaction. The current source address is incremented during post increment addressing mode or indexing mode.

16.2.4.7 Current Destination Address

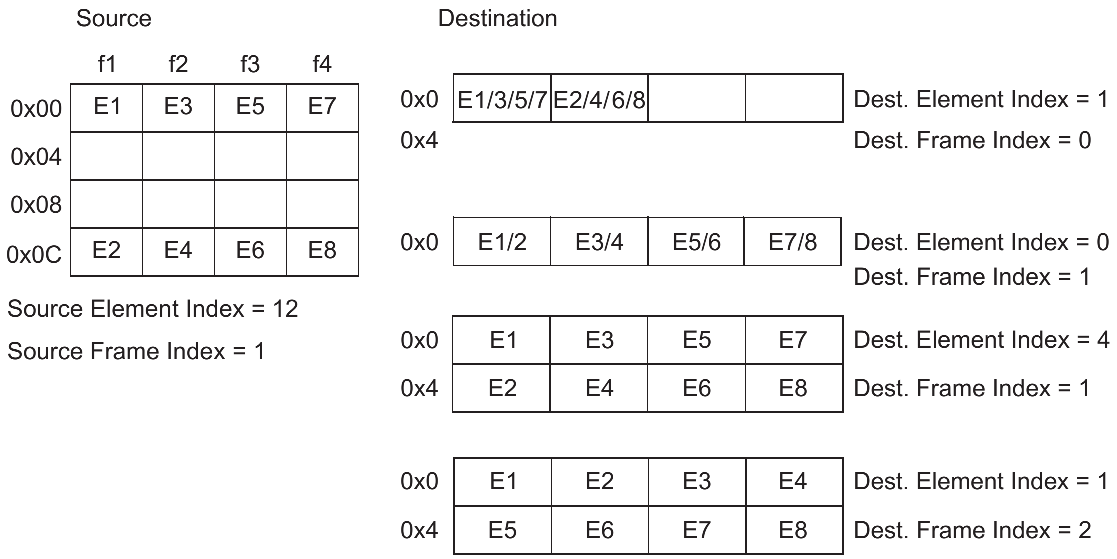
The current destination address field contains the current working destination address during a DMA transaction. The current destination address is incremented during post-increment addressing mode or indexing mode.

16.2.4.8 Current Transfer Count

The current transfer count stores the remaining number of elements to be transferred in a block. It is decremented by one for each element read from the source location.

Figure 16-6, Figure 16-7, and Figure 16-8 show some examples of DMA transfers.

Figure 16-6. DMA Transfer Example 1



Source

	f1	f2	f3	f4
0x00	E1	E3	E5	E7
0x04				
0x08				
0x0C	E2	E4	E6	E8

Source Element Index = 12

Source Frame Index = 1

Destination

0x0	E1/3/5/7	E2/4/6/8		
0x4				

Dest. Element Index = 1

Dest. Frame Index = 0

0x0	E1/2	E3/4	E5/6	E7/8

Dest. Element Index = 0

Dest. Frame Index = 1

0x0	E1	E3	E5	E7
0x4	E2	E4	E6	E8

Dest. Element Index = 4

Dest. Frame Index = 1

0x0	E1	E2	E3	E4
0x4	E5	E6	E7	E8

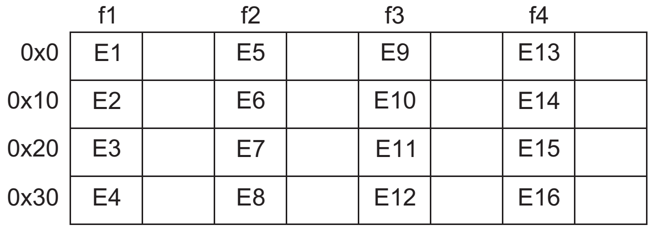
Dest. Element Index = 1

Dest. Frame Index = 2

The example assumes the following setup.

Read Element Size = 8 bit
Write Element Size = 8 bit
Element Count = 2
Frame Count = 4

Figure 16-7. DMA Indexing Example 1



	f1		f2		f3		f4	
0x0	E1		E5		E9		E13	
0x10	E2		E6		E10		E14	
0x20	E3		E7		E11		E15	
0x30	E4		E8		E12		E16	

Element Index = 16

Frame Index = 4

This example can be applied to either source or destination indexing and assumes the following setup.

Element Size = 16 bit
Element Count = 4
Frame Count = 4

Figure 16-8. DMA Indexing Example 2

0x0	E1	E4	E7	E10	E13	E16	E19	E22
0x20								
0x40	E2	E5	E8	E11	E14	E17	E20	E23
0x60								
0x80	E3	E6	E9	E12	E15	E18	E21	E23

Element Index = 64
Frame Index = 4

This example can be applied to either source or destination indexing and assumes the following setup.

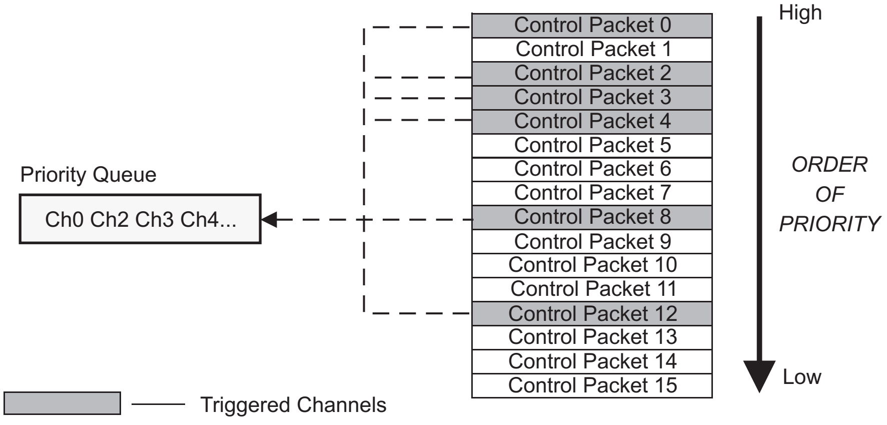
Element Size = 32 bit
Element Count = 3
Frame Count = 8

16.2.5 Priority Queue

User can assign channels in to priority queues to facilitate request handling during arbitration. The port has two priority queues: a high and a low priority queue. The queue can be configured to follow a fixed or rotating priority scheme. Fixed priority is such that the lower the channel number (Figure 16-9), the higher its priority. Rotating priority is based on a round-robin scheme. Initially, the priority list is sorted according to the fixed priority scheme. Channels assigned to the high priority queue are always serviced first according to the selected priority scheme before channels in the low priority queue are serviced. Table 16-1 describes how arbitration is performed according to different priority schemes.

NOTE: Since the DMA controller provides the capability to map any one of the 32 hardware DMA request lines to any channel, the numerical order of the hardware DMA request does not imply any priority. The priority of each hardware DMA request is programmed and determined by software.

Figure 16-9. Fixed Priority Scheme



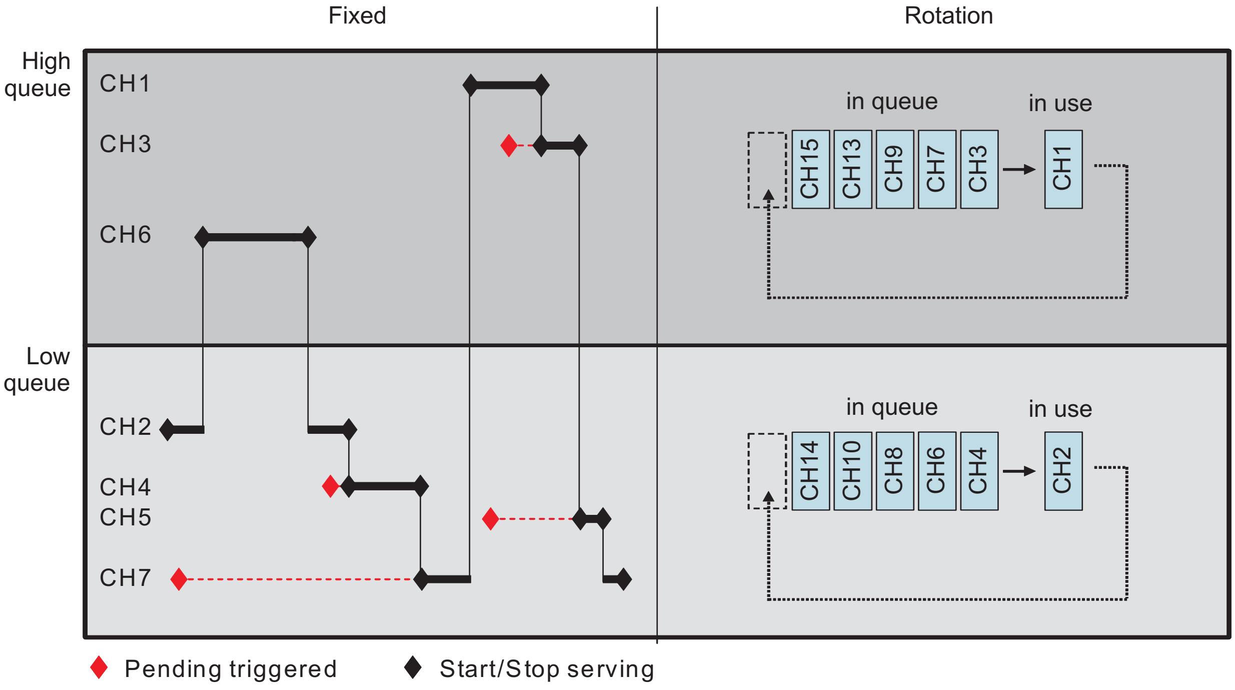
The above figure illustrates that by default Lower the channel number, higher the Priority.

Table 16-1. Arbitration According to Priority Queues and Priority Schemes

Queue	Priority Scheme	Remark
High priority	Fixed	Channels are serviced in an ascending order according to the channel number. The lower the channel number, the higher the priority. A channel will be arbitrated out whenever there is a higher pending channel. Otherwise a channel is completely serviced until its transfer count reaches zero before the next highest pending channel is serviced. When there is no pending channels left in high queue then the DMA switches to service low queue channels.
	Rotating	Channels are arbitrated by using the round-robin scheme. Arbitration is performed when the FIFO is empty. When there are no pending channels left in high queue then the DMA switches to service low queue channels.
Low priority	Fixed	Channels are serviced in an ascending order according to the channel number. The lower the channel number the higher the priority. A channel will be arbitrated out whenever there is a higher-priority pending channel. Otherwise a channel is completely serviced until its transfer count reaches zero, before the next highest pending channel is serviced. If there is a pending channel in the high-priority queue while DMA is servicing a low queue channel then DMA will switch back to service high queue channel after an arbitration boundary.
	Rotating	Channels are arbitrated by using round-robin scheme. Arbitration is performed when the FIFO is empty.

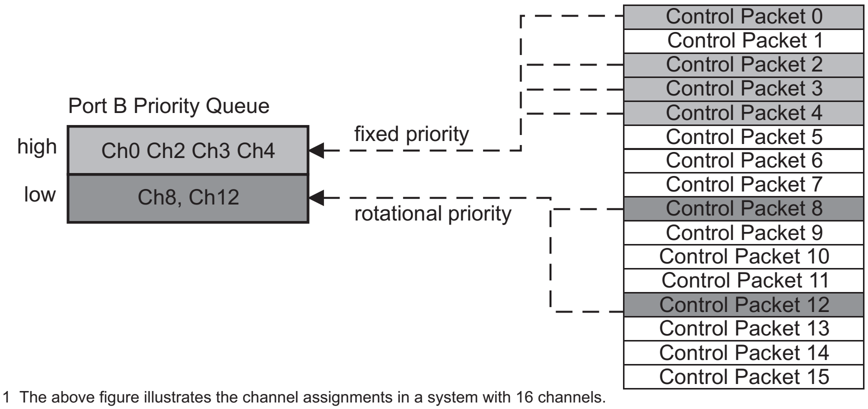
A Simple Priority Queues example in both Fixed and Rotation Scheme is shown in Figure 16-10.

Figure 16-10. Example of Priority Queues



For optimal system performance, the high priority channels should be put in fixed arbitration scheme and low priority channels in the rotating priority scheme as illustrated in Figure 16-11.

Figure 16-11. Example Channel Assignments



1 The above figure illustrates the channel assignments in a system with 16 channels. This approach can be scaled dependent on the total channels available.

16.2.6 Data Packing and Unpacking

The DMA controller automatically performs the necessary data packing and unpacking when the read element size differs from the write element size. Data packing is required when the read element size is smaller than the write element size; data unpacking is required when the read element size is larger than the write element size. When the read element size is equal to the write element size, no packing is performed during read, nor is any unpacking performed during write.

Figure 16-12 shows an example of data unpacking in which the DMA is used to transfer 128 transmit data elements to the MibSPI FIFO buffer. In this example, data unpacking is required because the read element size is 64 while the write element size is 16. The DMA first performs an 64-bit read from the source into its FIFO buffer. After the 64-bit data is read into the DMA FIFO buffer, it must unpack the data into four 16-bit data elements before writing out to the destination. Therefore the DMA would need to perform four 16-bit write operations to the destination.

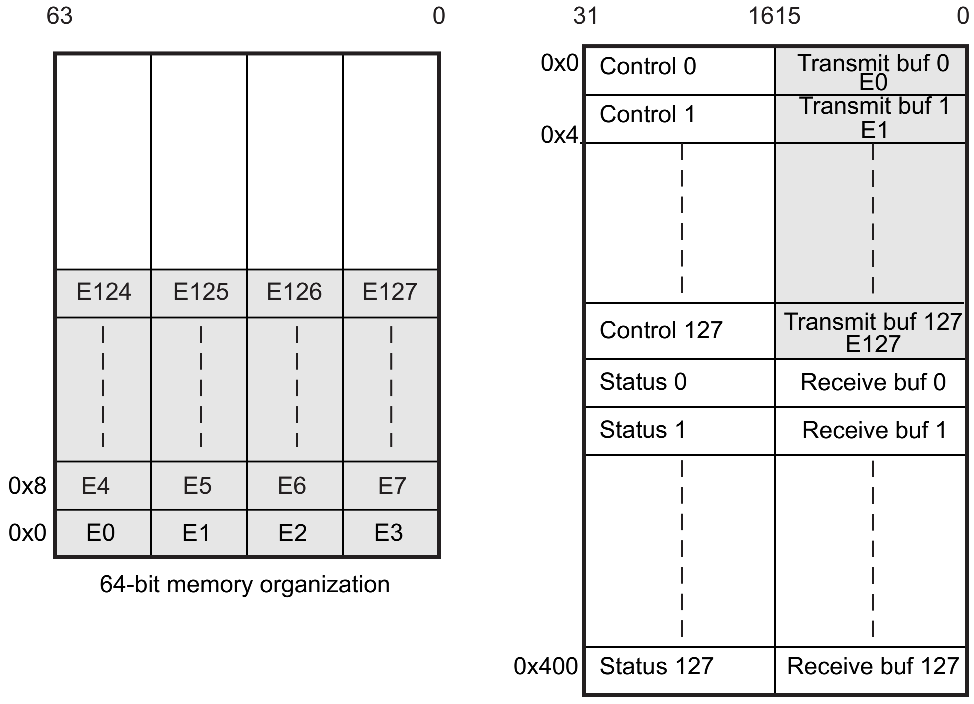
NOTE: In the example in Figure 16-12, to transmit data at the lower bits of the MibSPI, bits 15:0, the destination address should be incremented by a factor of 2.

NOTE: 1) The element Count (Section 16.3.2.3) refers only to the read element.

2) Data unpacking does not require the DMA request. Once the DMA request is received, data from Source is moved in to FIFO and unpacking happens until the FIFO is empty.

3) DMA assumes the destination is always ready and will perform write immediately. In case of data unpacking and Constant Addressing Mode write (Section 16.3.2.4 (1 - 0) = 0) the destination data will be overwritten by next data or next data might be skipped in case the destination has overflow protection (eg., SCITD register). User should configure DMA to avoid data unpacking if the Destination is configured as Constant Addressing Mode write to avoid data loss.

Figure 16-12. Example of DMA Data Unpacking



64-bit memory organization

MIBSPI FIFO organization

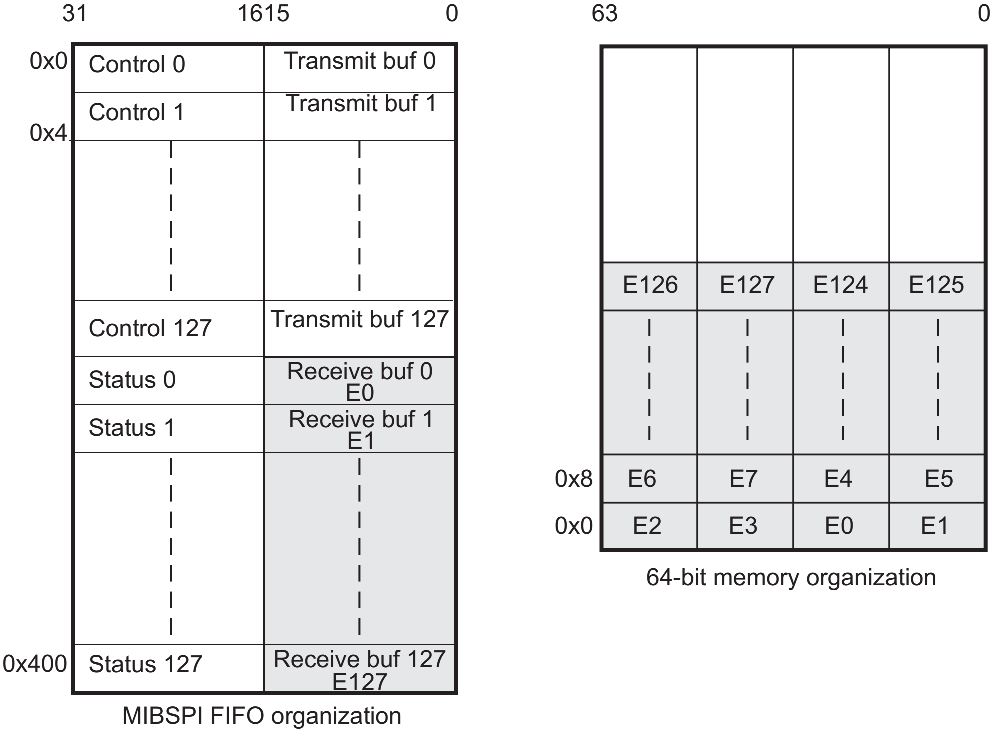
In this example, initialization of the MIBSPI FIFO is illustrated and assumes the following setup:

Read Element Size = 64 bit
Write Element Size = 16 bit
Element Count = 32
Frame Count = 1
Source Element Index = n/a, use post increment addressing mode
Source Frame Index = n/a, use post increment addressing mode
Destination Element Index = 4
Destination Frame Index = 0

When the read element size is smaller than the write element size, the DMA controller needs to perform data packing. The number of elements to pack is equal to the ratio between the write element size and read element size. In the example in Figure 16-13, the read element size is 16 bits and the write element size is 64 bits. The DMA controller would first pack the first four elements by performing four consecutive 16-bit read accesses of E0, E1, E2, and E3 into the first word of the DMA's internal FIFO. The DMA controller would then perform one single 64-bit write operation to transfer the data to the 64-bit destination memory.

Normally, the DMA controller carries out bus transactions on the bus according to the element size. For example, the DMA controller would perform a 16-bit read transaction if the read element size is programmed as 16 bits, or an 8-bit write transaction if the write element size is programmed as 8 bit. The exception is when the total transfer size is as defined in Equation 26 is not a multiple of the write element size.

Figure 16-13. Example of DMA Data Packing



In this example, a read of the MIBSPI FIFO is illustrated and assumes the following setup:

Read Element Size = 16 bit
Write Element Size = 64 bit
Element Count = 128
Frame Count = 1
Source Element Index = 4
Source Frame Index = 0
Destination Element Index = n/a, use post increment addressing mode
Destination Frame Index = n/a, use post increment addressing mode

For example, if the read element size is 8 bits, the element transfer count is equal to 9, and the write element size is 64 bit. The DMA controller would first perform eight 8-bit read transactions from the source. It would then perform a 64-bit write to the destination. When the same channel wins arbitration again, the DMA controller would first perform one 8-bit read from the source, followed by one 8-bit write to the destination, even though the write element size is 64 bit.

NOTE: Since peripherals are slower, it is advised to use data packing feature with caution for reading data from peripherals. Improper use might delay servicing other pending DMA channels.

16.2.7 DMA Request

There are three ways to start a DMA transfer:

- **Software request:** The transfer will be triggered by writing to SW Channel Enable Set and Status Register (Section 16.3.1.6). The software request can trigger either a block or a frame transfer depending on the setting of the TTYPE bit in the Channel Control Register (Section 16.3.2.4).
- **Hardware request**: The DMA controller can handle up to 32 DMA Request lines. A hardware request can trigger either a frame or a block transfer depending on the setting of the TTYPE bit in the Channel Control Register (Section 16.3.2.4).
- **Triggered by other control packet:** When a control packet finishes the programmed number of transfers it can trigger another channel to initiate its transfers.

Each time a DMA request is made, either one frame transfer or one block transfer can be chosen. An active DMA request signal will trigger a DMA transaction.

The DMA controller has a two-level buffer to capture HW requests per channel. When a HW request is generated and the channel is enabled, the corresponding bit in the DMA Status Register (Section 16.3.1.3) is set. The pending register acts as a first-level buffer. Typically, a peripheral acting as a source of a transfer would initiate another request after its data registers have been read out by DMA, even though that data has not been completely transferred to the destination. If a second HW request is generated by the peripheral, the DMA controller has an extra request buffer to capture this second request and service it after the first request is complete.

NOTE: The DMA cannot capture more than three requests if its request buffers are already full. If any request occur during this moment DMA will discard it.

The DMA controller also supports a mix of hardware and software requests on the same channel. Note that such interchangeable usage may result into an out of sync for DMA channel and peripheral. The application needs to be careful as the DMA does not have a built-in mechanism to protect against this loss of synchronization.

If a software request is generated, the corresponding bit in the Channel Pending Register (Section 16.3.1.2) is set accordingly. If the pending request is not completely serviced by the DMA and a hardware request is generated by a peripheral onto the same channel, the DMA will capture and recognize this hardware request into its request buffer.

NOTE: The DMA controller cannot recognize two software requests on the same channel if the first software request is still pending. If such request occur DMA will discard it. Therefore the user software should check the pending register before issuing a new software request.

The DMA module has 16 channels and up to 32 hardware DMA requests. The module contains DREQASIx registers that are used to map the DMA requests to the DMA channels. By default, channel 0 is mapped to request 0, channel 1 to request 1, and so on.

Some DMA requests have multiple sources, as shown in Table 16-2. The application must ensure that only one of these DMA request sources is enabled at any time.

Table 16-2. DMA Request Line Connection

Modules	DMA Request Sources	DMA Request
MIBSPI1	MIBSPI1[1][1]	DMAREQ[0]
MIBSPI1	MIBSPI1[0][2]	DMAREQ[1]
SPI2	SPI2 receive	DMAREQ[2]
SPI2	SPI2 transmit	DMAREQ[3]
MIBSPI1 / MIBSPI3 / DCAN2	MIBSPI1[2] / MIBSPI3[2] / DCAN2 IF3	DMAREQ[4]
MIBSPI1 / MIBSPI3 / DCAN2	MIBSPI1[3] / MIBSPI3[3] / DCAN2 IF2	DMAREQ[5]
DCAN1 / MIBSPI5	DCAN1 IF2 / MIBSPI5[2]	DMAREQ[6]

[1] SPI1, SPI3, SPI5 receive in standard SPI mode

[2] SPI1, SPI3, SPI5 transmit in standard SPI mode

Table 16-2. DMA Request Line Connection (continued)

Modules	DMA Request Sources	DMA Request
MIBADC1 / MIBSPI5	MIBADC1 event / MIBSPI5[3]	DMAREQ[7]
MIBSPI1 / MIBSPI3 / DCAN1	MIBSPI1[4] / MIBSPI3[4] / DCAN1 IF1	DMAREQ[8]
MIBSPI1 / MIBSPI3 / DCAN2	MIBSPI1[5] / MIBSPI3[5] / DCAN2 IF1	DMAREQ[9]
MIBADC1 / I2C / MIBSPI5	MIBADC1 G1 / I2C receive / MIBSPI5[4]	DMAREQ[10]
MIBADC1 / I2C / MIBSPI5	MIBADC1 G2 / I2C transmit / MIBSPI5[5]	DMAREQ[11]
RTI / MIBSPI1 / MIBSPI3	RTI DMAREQ0 / MIBSPI1[6] / MIBSPI3[6]	DMAREQ[12]
RTI / MIBSPI1 / MIBSPI3	RTI DMAREQ1 / MIBSPI1[7] / MIBSPI3[7]	DMAREQ[13]
MIBSPI3 / USB Device / MibADC2 / MIBSPI5	MIBSPI3[1][1] / USB_FUNC.DMATXREQ_ON[0] / MibADC2 event / MIBSPI5[6]	DMAREQ[14]
MIBSPI3 / USB Device / MIBSPI5	MIBSPI3[0][2] / USB_FUNC.DMARXREQ_ON[0] / MIBSPI5[7]	DMAREQ[15]
MIBSPI1 / MIBSPI3 / DCAN1 / MibADC2	MIBSPI1[8] / MIBSPI3[8] / DCAN1 IF3 / MibADC2 G1	DMAREQ[16]
MIBSPI1 / MIBSPI3 / DCAN3 / MibADC2	MIBSPI1[9] / MIBSPI3[9] / DCAN3 IF1 / MibADC2 G2	DMAREQ[17]
RTI / USB Device / MIBSPI5	RTI DMAREQ2 / USB_FUNC.DMATXREQ_ON[1] / MIBSPI5[8]	DMAREQ[18]
RTI / USB Device / MIBSPI5	RTI DMAREQ3 / USB_FUNC.DMARXREQ_ON[1] / MIBSPI5[9]	DMAREQ[19]
N2HET1 / N2HET2 / DCAN3	N2HET1 DMAREQ[4] / N2HET2 DMAREQ[4] / DCAN3 IF2	DMAREQ[20]
N2HET1 / N2HET2 / DCAN3	N2HET1 DMAREQ[5] / N2HET2 DMAREQ[5] / DCAN3 IF3	DMAREQ[21]
MIBSPI1 / MIBSPI3 / MIBSPI5	MIBSPI1[10] / MIBSPI3[10] / MIBSPI5[10]	DMAREQ[22]
MIBSPI1 / MIBSPI3 / MIBSPI5	MIBSPI1[11] / MIBSPI3[11] / MIBSPI5[11]	DMAREQ[23]
N2HET1 / N2HET2 / SPI4 / MIBSPI5	N2HET1 DMAREQ[6] / N2HET2 DMAREQ[6] / SPI4 receive / MIBSPI5[12]	DMAREQ[24]
N2HET1 / N2HET2 / SPI4 / MIBSPI5	N2HET1 DMAREQ[7] / N2HET2 DMAREQ[7] / SPI4 transmit / MIBSPI5[13]	DMAREQ[25]
CRC / MIBSPI1 / MIBSPI3	CRC DMAREQ[0] / MIBSPI1[12] / MIBSPI3[12]	DMAREQ[26]
CRC / MIBSPI1 / MIBSPI3	CRC DMAREQ[1] / MIBSPI1[13] / MIBSPI3[13]	DMAREQ[27]
LIN / USB Device / MIBSPI5	LIN receive / USB_FUNC.DMATXREQ_ON[2] / MIBSPI5[14]	DMAREQ[28]
LIN / USB Device / MIBSPI5	LIN transmit / USB_FUNC.DMARXREQ_ON[2] / MIBSPI5[15]	DMAREQ[29]
MIBSPI1 / MIBSPI3 / SCI / MIBSPI5	MIBSPI1[14] / MIBSPI3[14] / SCI receive / MIBSPI5[1][3]	DMAREQ[30]
MIBSPI1 / MIBSPI3 / SCI / MIBSPI5	MIBSPI1[15] / MIBSPI3[15] / SCI transmit / MIBSPI5[0][4]	DMAREQ[31]

[3] SPI1, SPI3, SPI5 receive in standard SPI mode

[4] SPI1, SPI3, SPI5 transmit in standard SPI mode

16.2.8 Auto-Initiation

When Auto-initiation Mode (AIM) bit of Channel Control Register (Section 16.3.2.4) is enabled for a channel and the channel is triggered by a software request for a block transfer, the channel will restart again using the same channel information stored at the respective control packet after one block transfer is completed. In the case of Hardware Request, the channel needs to be retriggered each time after a block is complete even if auto-initiation is enabled.

16.2.9 Interrupts

Each channel can be configured to generate interrupts on several transfer conditions:

- Frame transfer complete (FTC) interrupt: an interrupt is issued after the last element of a frame has been transferred.
- Last frame transfer started (LFS) interrupt: an interrupt is issued before the first element of the last frame of a block transfer has started.
- First half of block complete (HBC) interrupt: an interrupt is issued if more than half of the block is transferred.
  - If the number of frames *n* is odd, then the HBC interrupt is generated at the end of the frame when *(n+1) / 2* number of frames are left in the block.
  - If the number of frames *n* is even, then the HBC interrupt is generated at the end of the frame after *n/2* number of frames are left in the block.
- Block transfer complete (BTC) interrupt: an interrupt is issued after the last element of the last frame has been transferred.
- External imprecise error on read: an interrupt can be issued when a bus error (Illegal transaction with ok response) is detected. The imprecise read error is connected to the ESM module.
- External imprecise error on write: an interrupt can be issued when a bus error (Illegal transaction with ok response) is detected. The imprecise write error is connected to the ESM module.
- Memory Protection Unit error (MPU): an interrupt is issued when the DMA detects that the access falls outside of a memory region programmed in the MPU registers of the DMA. The MPU interrupt is connected to the ESM module.
- Parity error (PAR): an interrupt is issued when the DMA detects a parity error when reading one of the control packets. The PAR interrupt is connected to the ESM module.

The DMA outputs 5 interrupt lines for control packet handling, a parity interrupt and a memory protection interrupt (Figure 16-14). Each type of transfer interrupt condition is grouped together. For example, all block-transfer complete interrupts that are routed to a port are combined (ORed). The channel that caused the interrupt is given in the corresponding interrupt channel offset register. Priority between interrupts among the same interrupt type is resolved by a fixed priority scheme. Priority between different interrupt types is resolved in the Vector Interrupt Manager. Figure 16-15 explains the Frame Transfer Complete Interrupt structure in detail.

NOTE: Each Channel Specific interrupts in DMA module are routed towards Group A or B to support two different CPUs individually. For devices with Single CPU or Dual CPU where both CPUs are running same code in delayed lock-step as safety feature:

Group A - Interrupts (FTC, LFS, HBC, and BTC) are routed to the ARM CPU.

Group B - Interrupts (FTC, LFS, HBC, and BTC) are not routed out.

User software should configure only Group A interrupts.

Figure 16-14. DMA Interrupts

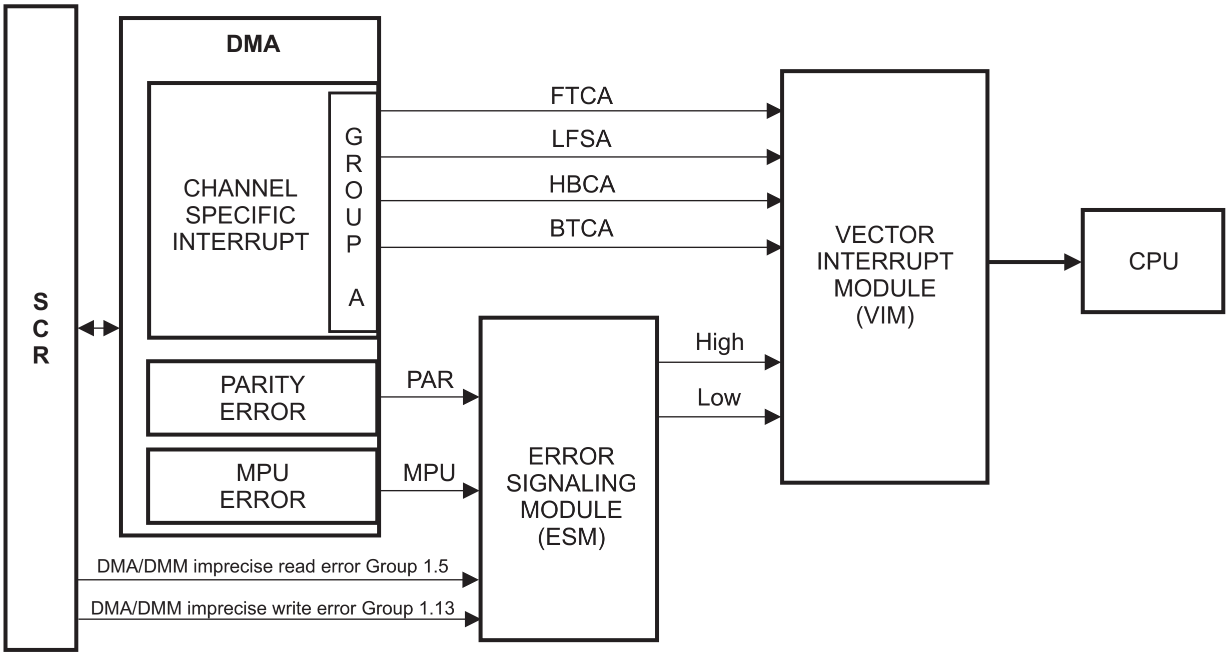
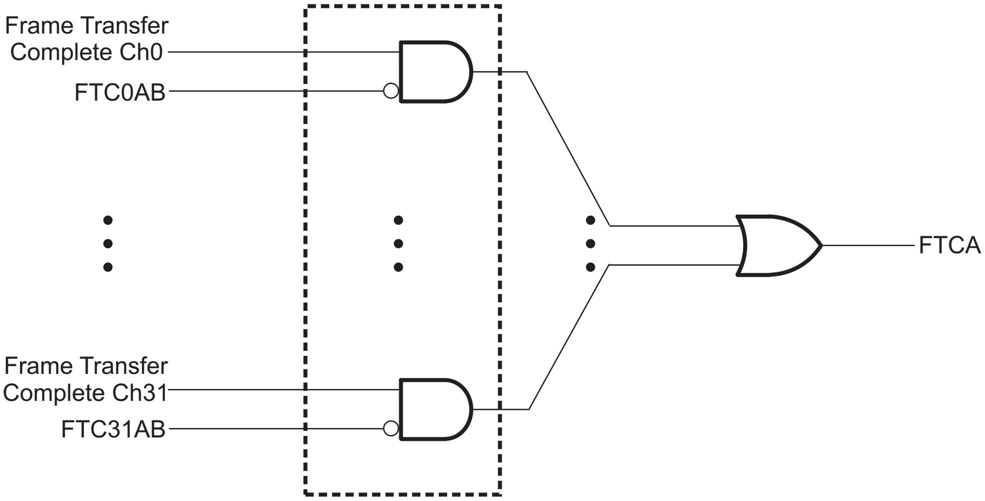

S
C
R
DMA
CHANNEL SPECIFIC INTERRUPT
G R O U P A
PARITY ERROR
MPU ERROR
FTCA
LFSA
HBCA
BTCA
PAR
MPU
High
Low
ERROR SIGNALING MODULE (ESM)
VECTOR INTERRUPT MODULE (VIM)
CPU
DMA/DMM imprecise read error Group 1.5
DMA/DMM imprecise write error Group 1.13

Figure 16-15. Detailed Interrupt Structure (Frame Transfer Complete Path)



This figure is applicable for the HBC, LFS, and BTC interrupt.

16.2.10 Debugging

The DMA supports four different behaviors in suspend mode. These behaviors can be configured by the user as per the application requirement.

- Immediate stop at a DMA channel arbitration boundary. Please refer to Table 16-3 and Table 16-4 for arbitration boundary definition.
- Finish current frame transfer and continue after suspend ends.
- Finish current block transfer and continue after suspend ends.
- Ignore the suspend. The DMA continues to be operational as in functional mode when debug mode is active.

When the DMA controller enters suspend mode, it continues to sample incoming hardware DMA requests, but the Channel Pending Register (Section 16.3.1.2) is frozen from being updated. After the suspend ends, all new requests that were received during suspend mode are reflected in the Channel Pending Register (Section 16.3.1.2).

Except when the DMA controller is configured to ignore suspend mode, no channel arbitration is performed during suspend mode. The current channel under which suspend mode was entered will finish its entire frame or block-transfer after suspend mode ends, depending how the debug option was chosen.

To facilitate debugging, a Watch Point Register (Section 16.3.1.47) and a Watch Mask Register (Section 16.3.1.48) are used. The watch point register together with the watch mask register can be configured to watch for a unique address or a range of addresses. When the condition to watch is true, the DMA freezes its state and generates a debug request signal to the host CPU so the state of the DMA can be examined.

16.2.11 Power Management

The DMA offers two power-management modes: run and sleep. In run mode, the DMA is fully operational.

The sleep mode shuts down the DMA if no pending channels are waiting to be serviced. If a DMA request is received or a software request is generated by the user software, then the DMA wakes up immediately.

The sleep mode may be used to optimize the DMA module power consumption.

When the system module issues a global low power mode request, the DMA will respond to the system module with an acknowledge if no DMA requests are pending.

NOTE: When the DMA is in global low power mode, the clock is stopped and therefore it cannot detect any DMA request. The device must be woken up before a peripheral can generate a DMA request.

16.2.12 FIFO Buffer

DMA FIFO is 4 levels deep and 64-bit wide (can hold up to 4 x 64-bits of data). They are used for Data packing and unpacking.

The DMA FIFO has two states:

- EMPTY : The FIFO contains no data.
- FULL : The FIFO is filled or the element count has reached zero; the read operation has to be stopped.

DMA channels can only be switched when the FIFO is empty. This also implies that arbitration between channels is done when the FIFO is empty.

The FIFO buffer may be bypassed through the use of the bypass feature in the port control register; see Port Control Register (Section 16.3.1.44) for register details. Writing 1 to this bit limits the FIFO depth to the size of one element. That means if the read element size is equal to or larger than the write element size, after one element is read the write out to the destination starts. Otherwise, the write out to the destination starts after enough reads have completed to do one write of the write element size. This feature is particularly useful to minimize switching latency in-between channels. When bypass mode is enabled, the DMA performs minimal bus cycles on AHB bus. In addition, the bypass feature allows arbitration between channels that can be carried out at a source element granularity.

However, it has to be considered that while in bypass mode, the DMA controller does not make optimal use of the bus bandwidth. Since the read and write element sizes can be different, then the number of read and write transactions will be different. Table 16-3 and Table 16-4 show a comparison between the number of read and write transactions performed by the DMA controller from one channel to another before arbitration in non-bypass and bypass mode.

Table 16-3. Maximum Number of DMA Transactions per Channel in Non-Bypass Mode

	Write Element Size	**8 bit**		**16 bit**		**32 bit**		**64 bit**	
Read Element Size	**8 bit**	4 read	4 write	4 read	2 write	4 read	1 write	8 read	1 write
	16 bit	2 read	4 write	4 read	4 write	4 read	2 write	4 read	1 write
	32 bit	1 read	4 write	2 read	4 write	4 read	4 write	4 read	2 write
	64 bit	1 read	8 write	1 read	4 write	2 read	4 write	4 read	4 write

Table 16-4. Maximum Number of DMA Transactions per Channel in Bypass Mode

	Write Element Size	**8 bit**		**16 bit**		**32 bit**		**64 bit**	
Read Element Size	**8 bit**	1 read	1 write	2 read	1 write	4 read	1 write	8 read	1 write
	16 bit	1 read	2 write	1 read	1 write	2 read	1 write	4 read	1 write
	32 bit	1 read	4 write	1 read	2 write	1 read	1 write	2 read	1 write
	64 bit	1 read	8 write	1 read	4 write	1 read	2 write	1 read	1 write

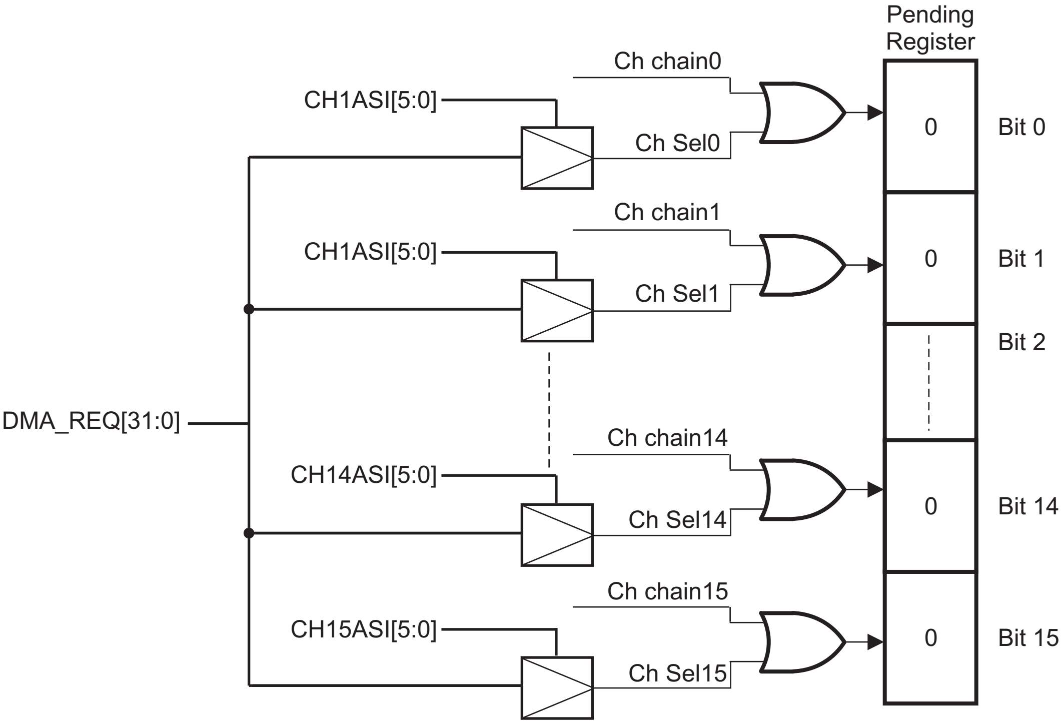
16.2.13 Channel Chaining

Channel chaining is used to trigger a single or multiple channels with out an external DMA request. This is possible by chaining one control packet to other. Chain[5:0] field of the Channel Control Register (Section 16.3.2.4) is used to program the chaining control packet. Chained control packets follow arbitration rules within the pending register. For example if CH1, CH2, CH4, CH5 are triggered together and CH3 is chained with CH1. The order of channels serviced in spite of chaining will be CH1 -> CH2 -> CH3 -> CH4 -> CH5.

In order to setup up channel chain feature, the Channel Control Register (Section 16.3.2.4) needs to be enabled for all chained channels before triggering first DMA request.

Figure 16-16 illustrates how internally chained request is generated after completing the required transfers and stored in pending register. In this example CH1 is Chained to CH0. When CH0 is triggered CH1 is captured as pending in the Channel Pending Register (Section 16.3.1.2) even when it is not triggered.

Figure 16-16. Example of Channel Chaining



16.2.14 Memory Protection

The DMA controller is capable of access to the full address range of the device. The protection mechanism allows the protection of up to four memory regions to restrict accesses to those address ranges. This will allow the application to protect critical application data from unintentionally being accessed by the DMA controller.

16.2.14.1 Protection Mechanism

The memory protection mechanism consists of the access privilege for a given memory region, the start and end address for the region, and notification of an access violation for the protected region.

Each region to be protected is configured by software by writing the start address and end address for each region into the DMA Memory Protection Registers, DMAMPRxS and DMAMPRxE. The definition of these registers can be found starting at Section 16.3.1.54. Any region in the valid address space can be protected from inappropriate accesses.

The access privileges can be set to one of four permission settings:

- Full access
- Read only access
- Write only access
- No access

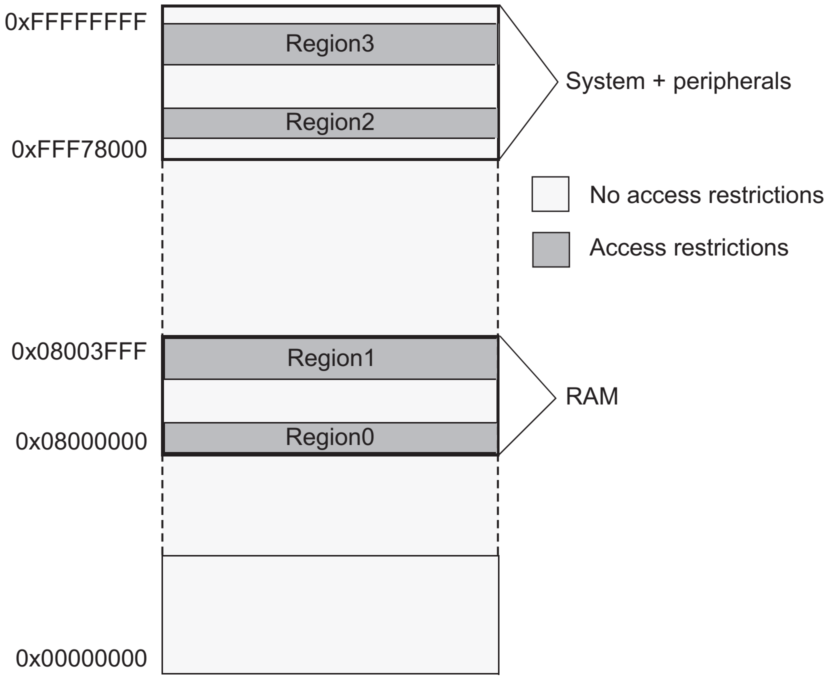
The permissions for a given region are selected by writing the appropriate values in the DMA Memory Protection Control Register (Section 16.3.1.54).

NOTE: If the regions defined by the start and end addresses overlap, the region defined first in the register space determines the access privilege. For example, if region 0 and region 1 overlap, the access permissions defined for region 0 will take precedence since region 0 registers are before region 1.

In a case where a memory protection violation occurs, a flag will be set and an interrupt will be generated, if interrupts are enabled. The DMA Memory Protection Status Register (Section 16.3.1.55) contains the status flags for the memory protection mechanism, and the DMA Memory Protection Control Register (Section 16.3.1.54) contains the interrupt enable bits. Upon detection of the memory protection violation, the DMA Channel that caused the violation will be stopped and the next available DMA channel will be serviced.

Figure 16-17 Illustrates a protection mechanism.

Figure 16-17. Example of Protection Mechanism



16.2.15 Parity Checking

Parity checking is implemented using parity on a per-byte basis for DMA Control Packets in the RAM. Checking for even or odd parity can be programmed by a 4-bit key located in the system module that controls the parity configuration on a global basis. This ensures that all modules using parity are acting in the same manner. The default setup after reset is odd parity.

In addition, parity checking can be enabled and disabled within the module by a 4-bit key. The key is located in the Parity Control Register (Section 16.3.1.52).

During a read access, regardless if it was read by the DMA state machine or another master (CPU), the parity is calculated based on the data read from the RAM and compared with the good parity value stored in the parity bits. If any word fails the parity check, then a parity error interrupt is generated. The address that generated the error is detected and is captured for host system debugging in the DMA Parity Error Address Register (Section 16.3.1.53). The address is frozen from being updated until it is read by the bus master.

Additional error handling is dependent on the requestor.

- DMA reading from a control packet RAM: The transmission requested by DMA request will not take place.
- CPU reading from the control packet RAM: The data will be retrieved by the CPU and a parity error interrupt will be generated.

In both cases, the control packet will be left active or the DMA will be switched off dependent on the ERRA bit in the Parity Control Register (Section 16.3.1.52).

16.2.16 Parity Testing

The parity RAM is accessible to allow manually inserting faults so that the parity checking feature can be tested. Test mode is entered by asserting the TEST bit in the Parity Control Register (Section 16.3.1.52). Once the bit is set, the parity bits are mapped to the control packet RAM starting address A00h.

NOTE: When in test mode, no parity checking will be done when reading from parity memory, but parity checking will be performed on the normal memory.

Each byte in Control Packet RAM has its own parity bit in the Control Packet Parity RAM as shown in Table 16-5, Table 16-6, and Table 16-7. P0 is the parity bit for byte 0, P1 is the parity bit for byte 1 and so on.

Each byte in the control packet RAM has its own parity bit in the control packet parity RAM as shown in Table 16-5 and Table 16-6.

Table 16-5. Control Packet RAM

Bit	31 24	23 16	15 8	7 0
Word0	Byte 0	Byte 1	Byte 2	Byte 3
Word1	Byte 4	Byte 5	Byte 6	Byte 7
Word2	Byte 8	Byte 9	Byte 10	Byte 11
Word3	Byte 12	Byte 13	Byte 14	Byte 15

Table 16-6. Control Packet RAM

Bit	127 96	95 64	63 32	31 0
	Word 3	Word 2	Word 1	Word 0

Table 16-7. Parity RAM

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

16.2.17 Initializing RAM with Parity

After power up, the RAM content including the parity bit cannot be guaranteed. To avoid parity failures when reading RAM, the RAM has to be initialized. The RAM can be initialized by writing known values into it. When the known value is written, the corresponding parity bit will be automatically calculated and updated.

Another possibility to initialize the memory is to follow the Auto-Initialization of On-Chip SRAM Modules subsection in the *Architecture* chapter. The RAM will be initialized to 0. Depending on the even/odd parity selection, the parity bit will be calculated accordingly.

To allow for parity calculation during initialization, the parity functionality has to be enabled as discussed in Section 16.2.15.

16.3 Control Registers and Control Packets

The DMA control registers are summarized in Table 16-8. The base address for the control registers is FFFF F000h. The control packets are summarized in Table 16-9. The base address for the control packets is FFF8 0000h. Each register begins on a word boundary. All registers and control packets are accessible in 8, 16, and 32 bit.

NOTE: The register definitions are given for a full DMA module configuration (32 channels, 64 requests, 2 Ports, Dual CPU support). Writes and Reads of bits pertaining to features not included in the DMA implementation as defined in the device-specific data manual are possible without error; however, they will have no affect on device operation.

Table 16-8. DMA Control Registers

Offset	Acronym	Register Description	Section
00h	GCTRL	Global Control Register	Section 16.3.1.1
04h	PEND	Channel Pending Register	Section 16.3.1.2
0Ch	DMASTAT	DMA Status Register	Section 16.3.1.3
14h	HWCHENAS	HW Channel Enable Set and Status Register	Section 16.3.1.4
1Ch	HWCHENAR	HW Channel Enable Reset and Status Register	Section 16.3.1.5
24h	SWCHENAS	SW Channel Enable Set and Status Register	Section 16.3.1.6
2Ch	SWCHENAR	SW Channel Enable Reset and Status Register	Section 16.3.1.7
34h	CHPRIOS	Channel Priority Set Register	Section 16.3.1.8
3Ch	CHPRIOR	Channel Priority Reset Register	Section 16.3.1.9
44h	GCHIENAS	Global Channel Interrupt Enable Set Register	Section 16.3.1.10
4Ch	GCHIENAR	Global Channel Interrupt Enable Reset Register	Section 16.3.1.11
54h	DREQASI0	DMA Request Assignment Register 0	Section 16.3.1.12
58h	DREQASI1	DMA Request Assignment Register 1	Section 16.3.1.13
5Ch	DREQASI2	DMA Request Assignment Register 2	Section 16.3.1.14
60h	DREQASI3	DMA Request Assignment Register 3	Section 16.3.1.15
94h	PAR0	Port Assignment Register 0	Section 16.3.1.16
98h	PAR1	Port Assignment Register 1	Section 16.3.1.17
B4h	FTCMAP	FTC Interrupt Mapping Register	Section 16.3.1.18
BCh	LFSMAP	LFS Interrupt Mapping Register	Section 16.3.1.19
C4h	HBCMAP	HBC Interrupt Mapping Register	Section 16.3.1.20
CCh	BTCMAP	BTC Interrupt Mapping Register	Section 16.3.1.21
DCh	FTCINTENAS	FTC Interrupt Enable Set Register	Section 16.3.1.22
E4h	FTCINTENAR	FTC Interrupt Enable Reset Register	Section 16.3.1.23
ECh	LFSINTENAS	LFS Interrupt Enable Set Register	Section 16.3.1.24
F4h	LFSINTENAR	LFS Interrupt Enable Reset Register	Section 16.3.1.25
FCh	HBCINTENAS	HBC Interrupt Enable Set Register	Section 16.3.1.26
104h	HBCINTENAR	HBC Interrupt Enable Reset Register	Section 16.3.1.27
10Ch	BTCINTENAS	BTC Interrupt Enable Set Register	Section 16.3.1.28
114h	BTCINTENAR	BTC Interrupt Enable Reset Register	Section 16.3.1.29
11Ch	GINTFLAG	Global Interrupt Flag Register	Section 16.3.1.30
124h	FTCFLAG	FTC Interrupt Flag Register	Section 16.3.1.31
12Ch	LFSFLAG	LFS Interrupt Flag Register	Section 16.3.1.32
134h	HBCFLAG	HBC Interrupt Flag Register	Section 16.3.1.33
13Ch	BTCFLAG	BTC Interrupt Flag Register	Section 16.3.1.34
144h	BERFLAG	BER Interrupt Flag Register	Section 16.3.1.35
14Ch	FTCAOFFSET	FTCA Interrupt Channel Offset Register	Section 16.3.1.36
150h	LFSAOFFSET	LFSA Interrupt Channel Offset Register	Section 16.3.1.37

Table 16-8. DMA Control Registers (continued)

Offset	Acronym	Register Description	Section
154h	HBCAOFFSET	HBCA Interrupt Channel Offset Register	Section 16.3.1.38
158h	BTCAOFFSET	BTCA Interrupt Channel Offset Register	Section 16.3.1.39
160h	FTCBOFFSET	FTCB Interrupt Channel Offset Register	Section 16.3.1.40
164h	LFSBOFFSET	LFSB Interrupt Channel Offset Register	Section 16.3.1.41
168h	HBCBOFFSET	HBCB Interrupt Channel Offset Register	Section 16.3.1.42
16Ch	BTCBOFFSET	BTCB Interrupt Channel Offset Register	Section 16.3.1.43
178h	PTCRL	Port Control Register	Section 16.3.1.44
17Ch	RTCTRL	RAM Test Control Register	Section 16.3.1.45
180h	DCTRL	Debug Control Register	Section 16.3.1.46
184h	WPR	Watch Point Register	Section 16.3.1.47
188h	WMR	Watch Mask Register	Section 16.3.1.48
198h	PBACSADDR	Port B Active Channel Source Address Register	Section 16.3.1.49
19Ch	PBACDADDR	Port B Active Channel Destination Address Register	Section 16.3.1.50
1A0h	PBACTC	Port B Active Channel Transfer Count Register	Section 16.3.1.51
1A8h	DMAPCR	Parity Control Register	Section 16.3.1.52
1ACh	DMAPAR	DMA Parity Error Address Register	Section 16.3.1.53
1B0h	DMAMPCTRL	DMA Memory Protection Control Register	Section 16.3.1.54
1B4h	DMAMPST	DMA Memory Protection Status Register	Section 16.3.1.55
1B8h	DMAMPR0S	DMA Memory Protection Region 0 Start Address Register	Section 16.3.1.56
1BCh	DMAMPR0E	DMA Memory Protection Region 0 End Address Register	Section 16.3.1.57
1C0h	DMAMPR1S	DMA Memory Protection Region 1 Start Address Register	Section 16.3.1.58
1C4h	DMAMPR1E	DMA Memory Protection Region 1 End Address Register	Section 16.3.1.59
1C8h	DMAMPR2S	DMA Memory Protection Region 2 Start Address Register	Section 16.3.1.60
1CCh	DMAMPR2E	DMA Memory Protection Region 2 End Address Register	Section 16.3.1.61
1D0h	DMAMPR3S	DMA Memory Protection Region 3 Start Address Register	Section 16.3.1.62
1D4h	DMAMPR3E	DMA Memory Protection Region 3 End Address Register	Section 16.3.1.63

Table 16-9. Control Packet Memory Map

Offset	Acronym	Register Description	Section
		Primary Control Packet 0	
00h	ISADDR	Initial Source Address Register	Section 16.3.2.1
04h	IDADDR	Initial Destination Address Register	Section 16.3.2.2
08h	ITCOUNT	Initial Transfer Count Register	Section 16.3.2.3
10h	CHCTRL	Channel Control Register	Section 16.3.2.4
14h	EIOFF	Element Index Offset Register	Section 16.3.2.5
18h	FIOFF	Frame Index Offset Register	Section 16.3.2.6
		Working Control Packet 0	
800h	CSADDR	Current Source Address Register	Section 16.3.2.7
804h	CDADDR	Current Destination Address Register	Section 16.3.2.8
808h	CTCOUNT	Current Transfer Count Register	Section 16.3.2.9

16.3.1 Global Configuration Registers

These registers control the overall behavior of the DMA controller.

16.3.1.1 Global Control Register (GCTRL)

Figure 16-18. Global Control Register (GCTRL) [offset = 00]

31 17	16
Reserved	DMA_EN
R-0	R/WP-0

15	14	13 10	9 8
Reserved	BUS_BUSY	Reserved	DEBUG_MODE
R-0	R-0	R-0	R/WP-0

7 1	0
Reserved	DMA_RES
R-0	R/WP-0

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

Table 16-10. Global Control Register (GCTRL) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reads return 0. Writes have no effect.
16	DMA_EN		DMA enable bit. The configuration registers and channel control packets should be setup first before DMA_EN bit is set to 1 to prevent state machines from carrying out bus transactions. If DMA_EN bit is cleared in the middle of an bus transaction, the state machine will stop at an arbitration boundary.
		0	The DMA is disabled.
		1	The DMA is enabled.
15	Reserved	0	Reads return 0. Writes have no effect.
14	BUS_BUSY		This bit indicates status of DMA external AHB bus status.
		0	DMAs external bus is not busy in data transfers.
		1	DMAs external bus is busy in data transfers.
13-10	Reserved	0	Reads return 0. Writes have no effect.
9-8	DEBUG_MODE		Debug Mode.
		0	Ignore suspend.
		1h	Finish current block transfer.
		2h	Finish current frame transfer.
		3h	Immediately stop at an DMA arbitration boundary and continue after suspend.
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	DMA_RES		DMA software reset **Note: In the event a DMA slave does not respond, the DMA module will respond to the software reset upon reaching an arbitration boundary.**
		0	Read: Software reset is disabled. Write: No effect.
		1	Read and write: The DMA state machine and all control registers are in software reset. Control packets are not reset when DMA software reset is active.

16.3.1.2 Channel Pending Register (PEND)

Figure 16-19. Channel Pending Register (PEND) [offset = 04h]

31	16
Reserved	
R-0	

15	0
PEND[15:0]	
R-0	

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

Table 16-11. Channel Pending Register (PEND) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	PEND[*n*]		Channel pending register. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on. Reading from PEND gives the channel pending information if the channel was initiated by SW or HW. Once set, it remains set even if the corresponding channel is disabled via HWCHENA or SWCHENA. The pending bit is automatically cleared for the following conditions: • At the end of a frame or a block transfer depending on how the channel is triggered as programmed in the TTYPE bit field of CHCTRL. • The control packet is modified after the pending bit is set. • An AHB bus error occurs.
		0	The corresponding channel is inactive.
		1	The corresponding channel is pending and is waiting for service.

16.3.1.3 DMA Status Register (DMASTAT)

Figure 16-20. DMA Status Register (DMASTAT) [offset = 0Ch]

31	16
Reserved	
R-0	

15	0
STCH[15:0]	
R-0	

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

Table 16-12. DMA Status Register (DMASTAT) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	STCH[*n*]		Status of DMA channels. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on.
		0	The channel is inactive.
		1	The channel is active; that is, the channel is currently in the DMA's execution queue. **Note: The status of a channel currently in DMA's execution queue remains active even if emulation mode is entered or DMA is disabled via DMA_EN bit.**

16.3.1.4 HW Channel Enable Set and Status Register (HWCHENAS)

Figure 16-21. HW Channel Enable Set and Status Register (HWCHENAS) [offset = 14h]

31	16
Reserved	
R-0	

15	0
HWCHENA[15:0]	
R/WP-0	

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

Table 16-13. HW Channel Enable Set and Status Register (HWCHENAS) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	HWCHENA[*n*]		Hardware channel enable bit. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on. An active hardware DMA request cannot initiate a DMA transfer unless the corresponding hardware enable bit is set. The corresponding hardware enable bit is cleared automatically for the following conditions: • At the end of a block transfer if the auto-initiation bit AIM (see CHCTRL) is not active. • If an AHB bus error is detected for an active channel. Reading from HWCHENAS gives the status (enabled/disabled) of channels 0 through 15.
		0	The corresponding channel is disabled for hardware triggering.
		1	The corresponding channel is enabled for hardware triggering.

16.3.1.5 HW Channel Enable Reset and Status Register (HWCHENAR)

Figure 16-22. HW Channel Enable Reset and Status Register (HWCHENAR) [offset = 1Ch]

31	16
Reserved	
R-0	

15	0
HWCHDIS[15:0]	
R/WP-0	

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

Table 16-14. HW Channel Enable Reset and Status Register (HWCHENAR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	HWCHDIS[*n*]		HW channel disable bit. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on.
		0	Read: The corresponding channel is disabled for HW triggering. Write: No effect.
		1	Read: The corresponding channel is enabled for HW triggering. Write: The corresponding channel is disabled.

16.3.1.6 SW Channel Enable Set and Status Register (SWCHENAS)

Figure 16-23. SW Channel Enable Set and Status Register (SWCHENAS) [offset = 24h]

31	16
Reserved	
R-0	

15	0
SWCHENA[15:0]	
R/WP-0	

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

Table 16-15. SW Channel Enable Set and Status Register (SWCHENAS) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	SWCHENA[*n*]		SW channel enable bit. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on. Writing a 1 to a bit triggers a SW request on the corresponding channel to start a DMA transaction. The corresponding bit is automatically cleared by the following conditions. • The corresponding bit is cleared after one frame transfer if the TTYPE bit in Channel Control Register (CHCTRL) is programmed for frame transfer. • The corresponding bit is cleared after one block transfer if the corresponding TTYPE bit is programmed for block transfer and the auto-initiation bit is not enabled. • The control packet is modified after the pending bit is set. • The corresponding bit is cleared after one block transfer when TTYPE bit is programmed for blocks transfer and if the corresponding bit in HW channel enable register (HWCHENAS) is enabled. When a channel is enabled for both HW and SW, the state machine will initiate transfers based on the SW first. After one block transfer is complete, the corresponding bit in the SWCHENA register is then cleared. The same channel is serviced again by a HW DMA request. • The corresponding bit is cleared if an AHB bus error is detected. Reading from SWCHENAS gives the status (enabled/disabled) of channels 0 through 15.
		0	The corresponding channel is not triggered by SW request.
		1	The corresponding channel is triggered by SW request.

16.3.1.7 SW Channel Enable Reset and Status Register (SWCHENAR)

Figure 16-24. SW Channel Enable Reset and Status Register (SWCHENAR) [offset = 2Ch]

31	16
Reserved	
R-0	

15	0
SWCHDIS[15:0]	
R/WP-0	

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

Table 16-16. SW Channel Enable Reset and Status Register (SWCHENAR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	SWCHDIS[*n*]		SW channel disable bit. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on.
		0	Read: The corresponding channel was not triggered by SW. Write: No effect.
		1	Read: The corresponding channel was triggered by SW. Write: The corresponding channel is disabled.

16.3.1.8 Channel Priority Set Register (CHPRIOS)

Figure 16-25. Channel Priority Set Register (CHPRIOS) [offset = 34h]

31	16
Reserved	
R-0	

15	0
CPS[15:0]	
R/WP-0	

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

Table 16-17. Channel Priority Set Register (CHPRIOS) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	CPS[*n*]		Channel priority set bit. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on. Writing a 1 to a bit assigns the corresponding channel to the high-priority queue.
		0	Read: The corresponding channel is assigned to the low-priority queue. Write: No effect.
		1	Read and write: The corresponding channel is assigned to high-priority queue.

16.3.1.9 Channel Priority Reset Register (CHPRIOR)

Figure 16-26. Channel Priority Reset Register (CHPRIOR) [offset = 3Ch]

31	16
Reserved	
R-0	

15	0
CPR[15:0]	
R/WP-0	

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

Table 16-18. Channel Priority Reset Register (CHPRIOR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	CPR[*n*]		Channel priority reset bit. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on. Writing a 1 to a bit assigns the according channel to the low-priority queue.
		0	Read: The corresponding channel is assigned to the low-priority queue. Write: No effect.
		1	Read: The corresponding channel is assigned to the high-priority queue. Write: The corresponding channel is assigned to the low-priority queue.

16.3.1.10 Global Channel Interrupt Enable Set Register (GCHIENAS)

Figure 16-27. Global Channel Interrupt Enable Set Register (GCHIENAS) [offset = 44h]

31	16
Reserved	
R-0	

15	0
GCHIE[15:0]	
R/WP-0	

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

Table 16-19. Global Channel Interrupt Enable Set Register (GCHIENAS) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	GCHIE[*n*]		Global channel interrupt enable bit. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on.
		0	Read: The corresponding channel is disabled for interrupt. Write: No effect.
		1	Read and write: The corresponding channel is enabled for interrupt.

16.3.1.11 Global Channel Interrupt Enable Reset Register (GCHIENAR)

Figure 16-28. Global Channel Interrupt Enable Reset Register (GCHIENAR) [offset = 4Ch]

31	16
Reserved	
R-0	

15	0
GCHID[15:0]	
R/WP-0	

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

Table 16-20. Global Channel Interrupt Enable Reset Register (GCHIENAR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	GCHID[*n*]		Global channel interrupt disable bit. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on.
		0	Read: The corresponding channel is disabled for interrupt. Write: No effect.
		1	Read: The corresponding channel is enabled for interrupt. Write: The corresponding channel is disabled for interrupt.

16.3.1.12 DMA Request Assignment Register 0 (DREQASI0)

Figure 16-29. DMA Request Assignment Register 0 (DREQASI0) [offset = 54h]

31 30	29 24	23 22	21 16
Reserved	CH0ASI	Reserved	CH1ASI
R-0	R/WP-0	R-0	R/WP-1h

15 14	13 8	7 6	5 0
Reserved	CH2ASI	Reserved	CH3ASI
R-0	R/WP-2h	R-0	R/WP-3h

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

Table 16-21. DMA Request Assignment Register 0 (DREQASI0) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Reads return 0. Writes have no effect.
29-24	CH0ASI		Channel 0 assignment. This bit field chooses the DMA request assignment for channel 0.
		0	DMA request line 0 triggers channel 0.
		:	:
		1Fh	DMA request line 31 triggers channel 0.
		20h-3Fh	Reserved
23-22	Reserved	0	Reads return 0. Writes have no effect.
21-16	CH1ASI		Channel 1 assignment. This bit field chooses the DMA request assignment for channel 1.
		0	DMA request line 0 triggers channel 1.
		:	:
		1Fh	DMA request line 31 triggers channel 1.
		20h-3Fh	Reserved
15-14	Reserved	0	Reads return 0. Writes have no effect.
13-8	CH2ASI		Channel 2 assignment. This bit field chooses the DMA request assignment for channel 2.
		0	DMA request line 0 triggers channel 2.
		:	:
		1Fh	DMA request line 31 triggers channel 2.
		20h-3Fh	Reserved
7-6	Reserved	0	Reads return 0. Writes have no effect.
5-0	CH3ASI		Channel 3 assignment. This bit field chooses the DMA request assignment for channel 3.
		0	DMA request line 0 triggers channel 3.
		:	:
		1Fh	DMA request line 31 triggers channel 3.
		20h-3Fh	Reserved

16.3.1.13 DMA Request Assignment Register 1 (DREQASI1)

Figure 16-30. DMA Request Assignment Register 1 (DREQASI1) [offset = 58h]

31 30	29 24	23 22	21 16
Reserved	CH4ASI	Reserved	CH5ASI
R-0	R/WP-4h	R-0	R/WP-5h

15 14	13 8	7 6	5 0
Reserved	CH6ASI	Reserved	CH7ASI
R-0	R/WP-6h	R-0	R/WP-7h

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

Table 16-22. DMA Request Assignment Register 1 (DREQASI1) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Reads return 0. Writes have no effect.
29-24	CH4ASI		Channel 4 assignment. This bit field chooses the DMA request assignment for channel 4.
		0	DMA request line 0 triggers channel 4.
		:	:
		1Fh	DMA request line 31 triggers channel 4.
		20h-3Fh	Reserved
23-22	Reserved	0	Reads return 0. Writes have no effect.
21-26	CH5ASI		Channel 5 assignment. This bit field chooses the DMA request assignment for channel 5.
		0	DMA request line 0 triggers channel 5.
		:	:
		1Fh	DMA request line 31 triggers channel 5.
		20h-3Fh	Reserved
15-14	Reserved	0	Reads return 0. Writes have no effect.
13-8	CH6ASI		Channel 6 assignment. This bit field chooses the DMA request assignment for channel 6.
		0	DMA request line 0 triggers channel 6.
		:	:
		1Fh	DMA request line 31 triggers channel 6.
		20h-3Fh	Reserved
7-6	Reserved	0	Reads return 0. Writes have no effect.
5-0	CH7ASI		Channel 7 assignment. This bit field chooses the DMA request assignment for channel 7.
		0	DMA request line 0 triggers channel 7.
		:	:
		1Fh	DMA request line 31 triggers channel 7.
		20h-3Fh	Reserved

16.3.1.14 DMA Request Assignment Register 2 (DREQASI2)

Figure 16-31. DMA Request Assignment Register 2 (DREQASI2) [offset = 5Ch]

31 30	29 24	23 22	21 16
Reserved	CH8ASI	Reserved	CH9ASI
R-0	R/WP-8h	R-0	R/WP-9h

15 14	13 8	7 6	5 0
Reserved	CH10ASI	Reserved	CH11ASI
R-0	R/WP-Ah	R-0	R/WP-Bh

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

Table 16-23. DMA Request Assignment Register 2 (DREQASI2) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Reads return 0. Writes have no effect.
29-24	CH8ASI		Channel 8 assignment. This bit field chooses the DMA request assignment for channel 8.
		0	DMA request line 0 triggers channel 8.
		:	:
		1Fh	DMA request line 31 triggers channel 8.
		20h-3Fh	Reserved
23-22	Reserved	0	Reads return 0. Writes have no effect.
21-16	CH9ASI		Channel 9 assignment. This bit field chooses the DMA request assignment for channel 9.
		0	DMA request line 0 triggers channel 9.
		:	:
		1Fh	DMA request line 31 triggers channel 9.
		20h-3Fh	Reserved
15-14	Reserved	0	Reads return 0. Writes have no effect.
13-8	CH10ASI		Channel 10 assignment. This bit field chooses the DMA request assignment for channel 10.
		0	DMA request line 0 triggers channel 10.
		:	:
		1Fh	DMA request line 31 triggers channel 10.
		20h-3Fh	Reserved
7-6	Reserved	0	Reads return 0. Writes have no effect.
5-0	CH11ASI		Channel 11 assignment. This bit field chooses the DMA request assignment for channel 11.
		0	DMA request line 0 triggers channel 11.
		:	:
		1Fh	DMA request line 31 triggers channel 11.
		20h-3Fh	Reserved

16.3.1.15 DMA Request Assignment Register 3 (DREQASI3)

Figure 16-32. DMA Request Assignment Register 3 (DREQASI3) [offset = 60h]

31 30	29 24	23 22	21 16
Reserved	CH12ASI	Reserved	CH13ASI
R-0	R/WP-Ch	R-0	R/WP-Dh

15 14	13 8	7 6	5 0
Reserved	CH14ASI	Reserved	CH15ASI
R-0	R/WP-Eh	R-0	R/WP-Fh

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

Table 16-24. DMA Request Assignment Register 3 (DREQASI3) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Reads return 0. Writes have no effect.
29-24	CH12ASI		Channel 12 assignment. This bit field chooses the DMA request assignment for channel 12.
		0	DMA request line 0 triggers channel 12.
		:	:
		1Fh	DMA request line 31 triggers channel 12.
		20h-3Fh	Reserved
23-22	Reserved	0	Reads return 0. Writes have no effect.
21-16	CH13ASI		Channel 13 assignment. This bit field chooses the DMA request assignment for channel 13.
		0	DMA request line 0 triggers channel 13.
		:	:
		1Fh	DMA request line 31 triggers channel 13.
		20h-3Fh	Reserved
15-14	Reserved	0	Reads return 0. Writes have no effect.
13-8	CH14ASI		Channel 14 assignment. This bit field chooses the DMA request assignment for channel 14.
		0	DMA request line 0 triggers channel 14.
		:	:
		1Fh	DMA request line 31 triggers channel 14.
		20h-3Fh	Reserved
7-6	Reserved	0	Reads return 0. Writes have no effect.
5-0	CH15ASI		Channel 15 assignment. This bit field chooses the DMA request assignment for channel 15.
		0	DMA request line 0 triggers channel 15.
		:	:
		1Fh	DMA request line 31 triggers channel 15.
		20h-3Fh	Reserved

16.3.1.16 Port Assignment Register 0 (PAR0)

Figure 16-33. Port Assignment Register 0 (PAR0) [offset = 94h]

31	30 28	27	26 24	23	22 20	19	18 16
Rsvd	CH0PA	Rsvd	CH1PA	Rsvd	CH2PA	Rsvd	CH3PA
R-0	R/WP-0	R-0	R/WP-0	R-0	R/WP-0	R-0	R/WP-0

15	14 12	11	10 8	7	6 4	3	2 0
Rsvd	CH4PA	Rsvd	CH5PA	Rsvd	CH6PA	Rsvd	CH7PA
R-0	R/WP-0	R-0	R/WP-0	R-0	R/WP-0	R-0	R/WP-0

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

Table 16-25. Port Assignment Register 0 (PAR0) Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reads return 0. Writes have no effect.
30-28	CH0PA		These bit fields determine to which port channel 0 is assigned.
		1xx	Port B
		0xx	Reserved
27	Reserved	0	Reads return 0. Writes have no effect.
26-24	CH1PA		These bit fields determine to which port channel 1 is assigned.
		1xx	Port B
		0xx	Reserved
23	Reserved	0	Reads return 0. Writes have no effect.
22-20	CH2PA		These bit fields determine to which port channel 2 is assigned.
		1xx	Port B
		0xx	Reserved
19	Reserved	0	Reads return 0. Writes have no effect.
18-16	CH3PA		These bit fields determine to which port channel 3 is assigned.
		1xx	Port B
		0xx	Reserved
15	Reserved	0	Reads return 0. Writes have no effect.
14-12	CH4PA		These bit fields determine to which port channel 4 is assigned.
		1xx	Port B
		0xx	Reserved
11	Reserved	0	Reads return 0. Writes have no effect.
10-8	CH5PA		These bit fields determine to which port channel 5 is assigned.
		1xx	Port B
		0xx	Reserved
7	Reserved	0	Reads return 0. Writes have no effect.
6-4	CH6PA		These bit fields determine to which port channel 6 is assigned.
		1xx	Port B
		0xx	Reserved
3	Reserved	0	Reads return 0. Writes have no effect.
2-0	CH7PA		These bit fields determine to which port channel 7 is assigned.
		1xx	Port B
		0xx	Reserved

16.3.1.17 Port Assignment Register 1 (PAR1)

Figure 16-34. Port Assignment Register 1 (PAR1) [offset = 98h]

31	30 28	27	26 24	23	22 20	19	18 16
Rsvd	CH8PA	Rsvd	CH9PA	Rsvd	CH10PA	Rsvd	CH11PA
R-0	R/WP-0	R-0	R/WP-0	R-0	R/WP-0	R-0	R/WP-0

15	14 12	11	10 8	7	6 4	3	2 0
Rsvd	CH12PA	Rsvd	CH13PA	Rsvd	CH14PA	Rsvd	CH15PA
R-0	R/WP-0	R-0	R/WP-0	R-0	R/WP-0	R-0	R/WP-0

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

Table 16-26. Port Assignment Register 1 (PAR1) Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reads return 0. Writes have no effect.
30-28	CH8PA		These bit fields determine to which port channel 8 is assigned.
		1xx	Port B
		0xx	Reserved
27	Reserved	0	Reads return 0. Writes have no effect.
26-24	CH9PA		These bit fields determine to which port channel 9 is assigned.
		1xx	Port B
		0xx	Reserved
23	Reserved	0	Reads return 0. Writes have no effect.
22-20	CH10PA		These bit fields determine to which port channel 10 is assigned.
		1xx	Port B
		0xx	Reserved
19	Reserved	0	Reads return 0. Writes have no effect.
18-16	CH11PA		These bit fields determine to which port channel 11 is assigned.
		1xx	Port B
		0xx	Reserved
15	Reserved	0	Reads return 0. Writes have no effect.
14-12	CH12PA		These bit fields determine to which port channel 12 is assigned.
		1xx	Port B
		0xx	Reserved
11	Reserved	0	Reads return 0. Writes have no effect.
10-8	CH13PA		These bit fields determine to which port channel 13 is assigned.
		1xx	Port B
		0xx	Reserved
7	Reserved	0	Reads return 0. Writes have no effect.
6-4	CH14PA		These bit fields determine to which port channel 14 is assigned.
		1xx	Port B
		0xx	Reserved
3	Reserved	0	Reads return 0. Writes have no effect.
2-0	CH15PA		These bit fields determine to which port channel 15 is assigned.
		1xx	Port B
		0xx	Reserved

16.3.1.18 FTC Interrupt Mapping Register (FTCMAP)

NOTE: On this device Group B interrupts are not implemented; hence, user software should configure only Group A interrupts.

Figure 16-35. FTC Interrupt Mapping Register (FTCMAP) [offset = B4h]

31	16
Reserved	
R-0	

15	0
FTCAB[15:0]	
R/WP-0	

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

Table 16-27. FTC Interrupt Mapping Register (FTCMAP) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	FTCAB[*n*]		Frame transfer complete (FTC) interrupt to Group A or Group B. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on.
		0	The FTC interrupt of the corresponding channel is routed to Group A.
		1	The FTC interrupt of the corresponding channel is routed to Group B.

16.3.1.19 LFS Interrupt Mapping Register (LFSMAP)

NOTE: On this device Group B interrupts are not implemented; hence, user software should configure only Group A interrupts.

Figure 16-36. LFS Interrupt Mapping Register (LFSMAP) [offset = BCh]

31	16
Reserved	
R-0	

15	0
LFSAB[15:0]	
R/WP-0	

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

Table 16-28. LFS Interrupt Mapping Register (LFSMAP) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	LFSAB[*n*]		Last frame started (LFS) interrupt to Group A or Group B. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on.
		0	The LFS interrupt of the corresponding channel is routed to Group A.
		1	The LFS interrupt of the corresponding channel is routed to Group B.

16.3.1.20 HBC Interrupt Mapping Register (HBCMAP)

NOTE: On this device Group B interrupts are not implemented; hence, user software should configure only Group A interrupts.

Figure 16-37. HBC Interrupt Mapping Register (HBCMAP) [offset = C4h]

31 — 16
Reserved
R-0

15 — 0
HBCAB[15:0]
R/WP-0

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

Table 16-29. HBC Interrupt Mapping Register (HBCMAP) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	HBCAB[*n*]		Half block complete (HBC) interrupt to Group A or Group B. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on.
		0	The HBC interrupt of the corresponding channel is routed to Group A.
		1	The HBC interrupt of the corresponding channel is routed to Group B.

16.3.1.21 BTC Interrupt Mapping Register (BTCMAP)

NOTE: On this device Group B interrupts are not implemented; hence, user software should configure only Group A interrupts.

Figure 16-38. BTC Interrupt Mapping Register (BTCMAP) [offset = CCh]

31 — 16
Reserved
R-0

15 — 0
BTCAB[15:0]
R/WP-0

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

Table 16-30. BTC Interrupt Mapping Register (BTCMAP) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	BTCAB[*n*]		Block transfer complete (BTC) interrupt to Group A or Group B. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on.
		0	The BTC interrupt of the corresponding channel is routed to Group A.
		1	The BTC interrupt of the corresponding channel is routed to Group B.

16.3.1.22 FTC Interrupt Enable Set (FTCINTENAS)

Figure 16-39. FTC Interrupt Enable Set (FTCINTENAS) [offset = DCh]

31	16
Reserved	
R-0	

15	0
FTCINTENA[15:0]	
R/WP-0	

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

Table 16-31. FTC Interrupt Enable Set (FTCINTENAS) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	FTCINTENA[*n*]		Frame transfer complete (FTC) interrupt enable. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on.
		0	Read: The corresponding FTC interrupt of a channel is disabled. Write: No effect.
		1	Read or write: The FTC interrupt of the corresponding channel is enabled.

16.3.1.23 FTC Interrupt Enable Reset (FTCINTENAR)

NOTE: On this device Group B interrupts are not implemented hence user software should configure only Group A interrupts.

Figure 16-40. FTC Interrupt Enable Reset (FTCINTENAR) [offset = E4h]

31	16
Reserved	
R-0	

15	0
FTCINTDIS[15:0]	
R/WP-0	

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

Table 16-32. FTC Interrupt Enable Reset (FTCINTENAR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	FTCINTDIS[*n*]		Frame transfer complete (FTC) interrupt disable. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on.
		0	Read: The corresponding FTC interrupt of a channel is disabled. Write: No effect.
		1	Read: The corresponding FTC interrupt of a channel is enabled. Write: The corresponding FTC interrupt is disabled.

16.3.1.24 LFS Interrupt Enable Set (LFSINTENAS)

Figure 16-41. LFS Interrupt Enable Set (LFSINTENAS) [offset = ECh]

31	16
Reserved	
R-0	

15	0
LFSINTENA[15:0]	
R/WP-0	

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

Table 16-33. LFS Interrupt Enable Set (LFSINTENAS) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	LFSINTENA[*n*]		Last frame started (LFS) interrupt enable. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on.
		0	Read: The corresponding LFS interrupt of a channel is disabled. Write: No effect.
		1	Read or write: The LFS interrupt of the corresponding channel is enabled.

16.3.1.25 LFS Interrupt Enable Reset (LFSINTENAR)

Figure 16-42. LFS Interrupt Enable Reset (LFSINTENAR) [offset = F4h]

31	16
Reserved	
R-0	

15	0
LFSINTDIS[15:0]	
R/WP-0	

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

Table 16-34. LFS Interrupt Enable Reset (LFSINTENAR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	LFSINTDIS[*n*]		Last frame started (LFS) interrupt disable. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on.
		0	Read: The LFS interrupt of the corresponding channel is disabled. Write: No effect.
		1	Read: The LFS interrupt of the corresponding channel is enabled. Write: The LFS interrupt of the corresponding channel is disabled.

16.3.1.26 HBC Interrupt Enable Reset (HBCINTENAS)

Figure 16-43. HBC Interrupt Enable Set (HBCINTENAS) [offset = FCh]

31	16
Reserved	
R-0	

15	0
HBCINTENA[15:0]	
R/WP-0	

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

Table 16-35. HBC Interrupt Enable Set (HBCINTENAS) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	HBCINTENA[*n*]		Half block complete (HBC) interrupt enable. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on.
		0	Read: The HBC interrupt of the corresponding channel is disabled. Write: No effect.
		1	Read or write: The HBC interrupt of the corresponding channel is enabled.

16.3.1.27 HBC Interrupt Enable Reset (HBCINTENAR)

Figure 16-44. HBC Interrupt Enable Reset (HBCINTENAR) [offset = 104h]

31	16
Reserved	
R-0	

15	0
HBCINTDIS[15:0]	
R/WP-0	

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

Table 16-36. HBC Interrupt Enable Reset (HBCINTENAR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	HBCINTDIS[*n*]		Half block complete (HBC) interrupt disable. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on.
		0	Read: The HBC interrupt of the corresponding channel is disabled. Write: No effect.
		1	Read: The HBC interrupt of the corresponding channel is enabled. Write: The HBC interrupt of the corresponding channel is disabled.

16.3.1.28 BTC Interrupt Enable Set (BTCINTENAS)

Figure 16-45. BTC Interrupt Enable Set (BTCINTENAS) [offset = 10Ch]

31	16
Reserved	
R-0	

15	0
BTCINTENA[15:0]	
R/WP-0	

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

Table 16-37. BTC Interrupt Enable Reset (BTCINTENAS) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	BTCINTENA[*n*]		Block transfer complete (BTC) interrupt enable. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on.
		0	Read: The BTC interrupt of the corresponding channel is disabled. Write: No effect.
		1	Read or write: The BTC interrupt of the corresponding channel is enabled.

16.3.1.29 BTC Interrupt Enable Reset (BTCINTENAR)

Figure 16-46. BTC Interrupt Enable Reset (BTCINTENAR) [offset = 114h]

31	16
Reserved	
R-0	

15	0
BTCINTDIS[15:0]	
R/WP-0	

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

Table 16-38. BTC Interrupt Enable Reset (BTCINTENAR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	BTCINTDIS[*n*]		Block transfer complete (BTC) interrupt disable. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on.
		0	Read: The BTC interrupt of the corresponding channel is disabled. Write: No effect.
		1	Read: The BTC interrupt of the corresponding channel is enabled. Write: The BTC interrupt of the corresponding channel is disabled.

16.3.1.30 Global Interrupt Flag Register (GINTFLAG)

Figure 16-47. Global Interrupt Flag Register (GINTFLAG) [offset = 11Ch]

31	16
Reserved	
R-0	

15	0
GINT[15:0]	
R/WP-0	

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

Table 16-39. Global Interrupt Flag Register (GINTFLAG) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	GINT[*n*]		Global interrupt flags. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on. A global interrupt flag bit is an OR function of FTC, LFS, HBC, and BTC interrupt flags.
		0	No interrupt is pending on the corresponding channel.
		1	One or more of the interrupt types (FTC, LFS, HBC, or BTC) is pending on the corresponding channel.

16.3.1.31 FTC Interrupt Flag Register (FTCFLAG)

Figure 16-48. FTC Interrupt Flag Register (FTCFLAG) [offset = 124h]

31	16
Reserved	
R-0	

15	0
FTCI[15:0]	
R/W1CP-0	

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 in privilege mode to clear the bit; -*n* = value after reset

Table 16-40. FTC Interrupt Flag Register (FTCFLAG) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	FTCI[*n*]		Frame transfer complete (FTC) flags. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on. **Note: Reading from the respective interrupt channel offset register also clears the corresponding flag (see Section 16.3.1.36 and Section 16.3.1.40).** **Note: The state of the flag bit can be polled even if the corresponding interrupt enable bit is cleared.**
		0	Read: An FTC interrupt of the corresponding channel is not pending. Write: No effect.
		1	Read: An FTC interrupt of the corresponding channel is pending. Write: The flag is cleared.

16.3.1.32 LFS Interrupt Flag Register (LFSFLAG)

Figure 16-49. LFS Interrupt Flag Register (LFSFLAG) [offset = 12Ch]

31	16
Reserved	
R-0	

15	0
LFSI[15:0]	
R/W1CP-0	

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 in privilege mode to clear the bit; -*n* = value after reset

Table 16-41. LFS Interrupt Flag Register (LFSFLAG) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	LFSI[*n*]		Last frame started (LFS) flags. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on. **Note: Reading from the respective interrupt channel offset register also clears the corresponding flag (see Section 16.3.1.37 and Section 16.3.1.41).** **Note: The state of the flag bit can be polled even if the corresponding interrupt enable bit is cleared.**
		0	Read: An LFS interrupt of the corresponding channel is not pending. Write: No effect.
		1	Read: An LFS interrupt of the corresponding channel is pending. Write: The flag is cleared.

16.3.1.33 HBC Interrupt Flag Register (HBCFLAG)

Figure 16-50. HBC Interrupt Flag Register (HBCFLAG) [offset = 134h]

31	16
Reserved	
R-0	

15	0
HBCI[15:0]	
R/W1CP-0	

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 in privilege mode to clear the bit; -*n* = value after reset

Table 16-42. HBC Interrupt Flag (HBCFLAG) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	HBCI[*n*]		Half block transfer (HBC) complete flags. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on. **Note: Reading from the respective interrupt channel offset register also clears the corresponding flag (see Section 16.3.1.38 and Section 16.3.1.42).** **Note: The state of the flag bit can be polled even if the corresponding interrupt enable bit is cleared.**
		0	Read: An HBC interrupt of the corresponding channel is not pending. Write: No effect.
		1	Read: An HBC interrupt of the corresponding channel is pending. Write: The flag is cleared.

16.3.1.34 BTC Interrupt Flag Register (BTCFLAG)

Figure 16-51. BTC Interrupt Flag Register (BTCFLAG) [offset = 13Ch]

31	16
Reserved	
R-0	

15	0
BTCI[15:0]	
R/W1CP-0	

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 in privilege mode to clear the bit; -*n* = value after reset

Table 16-43. BTC Interrupt Flag Register (BTCFLAG) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	BTCI[*n*]		Block transfer complete (BTC) flags. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on. **Note: Reading from the respective interrupt channel offset register also clears the corresponding flag (see Section 16.3.1.39 and Section 16.3.1.43).** **Note: The state of the flag bit can be polled even if the corresponding interrupt enable bit is cleared.**
		0	Read: A BTC interrupt of the corresponding channel is not pending. Write: No effect.
		1	Read: A BTC interrupt of the corresponding channel is pending. Write: The flag is cleared.

16.3.1.35 BER Interrupt Flag Register (BERFLAG)

The BERFLAG will never be set in this device. The bus error reporting is handled by the DMA Read Imprecise Error and DMA Write Imprecise Error asserted to the ESM module directly, which are detected at the device level. See the ESM error mapping for the DMA Read/Write Imprecise Error.

16.3.1.36 FTCA Interrupt Channel Offset Register (FTCAOFFSET)

Figure 16-52. FTCA Interrupt Channel Offset Register (FTCAOFFSET) [offset = 14Ch]

31-16
Reserved
R-0

15-8	7	6	5-0
Reserved	sbz	sbz	FTCA
R-0	R-0	R-0	R-0

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

Table 16-44. FTCA Interrupt Channel Offset Register (FTCAOFFSET) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-6	sbz	0	These bits should always be programmed as 0.
5-0	FTCA		Channel causing FTC interrupt Group A. These bits contain the channel number of the pending interrupt for Group A if the corresponding interrupt enable is set. **Note: Reading this location clears the corresponding interrupt pending flag (see Section 16.3.1.31) with the highest priority.**
		0	No interrupt is pending.
		1h	Channel 0 is causing the pending interrupt Group A.
		:	:
		10h	Channel 15 is causing the pending interrupt Group A.
		11h-3Fh	Reserved

16.3.1.37 LFSA Interrupt Channel Offset Register (LFSAOFFSET)

Figure 16-53. LFSA Interrupt Channel Offset Register (LFSAOFFSET) [offset = 150h]

31-16
Reserved
R-0

15-8	7	6	5-0
Reserved	sbz	sbz	LFSA
R-0	R-0	R-0	R-0

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

Table 16-45. LFSA Interrupt Channel Offset Register (LFSAOFFSET) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-6	sbz	0	These bits should always be programmed as 0.

Table 16-45. LFSA Interrupt Channel Offset Register (LFSAOFFSET) Field Descriptions (continued)

Bit	Field	Value	Description
5-0	LFSA		Channel causing LFS interrupt Group A. These bits contain the channel number of the pending interrupt for Group A if the corresponding interrupt enable is set. **Note: Reading this location clears the corresponding interrupt pending flag (see Section 16.3.1.32) with the highest priority.**
		0	No interrupt is pending.
		1h	Channel 0 is causing the pending interrupt Group A.
		:	:
		10h	Channel 15 is causing the pending interrupt Group A.
		11h-3Fh	Reserved

16.3.1.38 HBCA Interrupt Channel Offset Register (HBCAOFFSET)

Figure 16-54. HBCA Interrupt Channel Offset Register (HBCAOFFSET) [offset = 154h]

31 16
Reserved
R-0

15 8	7	6	5 0
Reserved	sbz	sbz	HBCA
R-0	R-0	R-0	R-0

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

Table 16-46. HBCA Interrupt Channel Offset Register (HBCAOFFSET) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-6	sbz	0	These bits should always be programmed as 0.
5-0	HBCA		Channel causing HBC interrupt Group A. These bits contain the channel number of the pending interrupt for Group A if the corresponding interrupt enable is set. **Note: Reading this location clears the corresponding interrupt pending flag (see Section 16.3.1.33) with the highest priority.**
		0	No interrupt is pending.
		1h	Channel 0 is causing the pending interrupt Group A.
		:	:
		10h	Channel 15 is causing the pending interrupt Group A.
		11h-3Fh	Reserved

16.3.1.39 BTCA Interrupt Channel Offset Register (BTCAOFFSET)

Figure 16-55. BTCA Interrupt Channel Offset Register (BTCAOFFSET) [offset = 158h]

31 16
Reserved
R-0

15 8	7	6	5 0
Reserved	sbz	sbz	BTCA
R-0	R-0	R-0	R-0

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

Table 16-47. BTCA Interrupt Channel Offset Register (BTCAOFFSET) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-6	sbz	0	These bits should always be programmed as 0.

Table 16-47. BTCA Interrupt Channel Offset Register (BTCAOFFSET) Field Descriptions (continued)

Bit	Field	Value	Description
5-0	BTCA		Channel causing BTC interrupt Group A. These bits contain the channel number of the pending interrupt for Group A if the corresponding interrupt enable is set. **Note: Reading this location clears the corresponding interrupt pending flag (see Section 16.3.1.34) with the highest priority.**
		0	No interrupt is pending.
		1h	Channel 0 is causing the pending interrupt Group A.
		:	:
		10h	Channel 15 is causing the pending interrupt Group A.
		11h-3Fh	Reserved

16.3.1.40 FTCB Interrupt Channel Offset Register (FTCBOFFSET)

Figure 16-56. FTCB Interrupt Channel Offset Register (FTCBOFFSET) [offset = 160h]

31	16
Reserved	
R-0	

15	8	7	6	5	0
Reserved		sbz	sbz	FTCB	
R-0		R-0	R-0	R-0	

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

Table 16-48. FTCB Interrupt Channel Offset Register (FTCBOFFSET) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-6	sbz	0	These bits should always be programmed as 0.
5-0	FTCB		Channel causing FTC interrupt Group B. These bits contain the channel number of the pending interrupt for Group B if the corresponding interrupt enable is set. **Note: Reading this location clears the corresponding interrupt pending flag (see Section 16.3.1.31) with the highest priority.**
		0	No interrupt is pending.
		1h	Channel 0 is causing the pending interrupt Group B.
		:	:
		10h	Channel 15 is causing the pending interrupt Group B.
		11h-3Fh	Reserved

16.3.1.41 LFSB Interrupt Channel Offset Register (LFSBOFFSET)

Figure 16-57. LFSB Interrupt Channel Offset Register (LFSBOFFSET) [offset = 164h]

31	16
Reserved	
R-0	

15	8	7	6	5	0
Reserved		sbz	sbz	LFSB	
R-0		R-0	R-0	R-0	

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

Table 16-49. LFSB Interrupt Channel Offset Register (LFSBOFFSET) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-6	sbz	0	These bits should always be programmed as 0.

Table 16-49. LFSB Interrupt Channel Offset Register (LFSBOFFSET) Field Descriptions (continued)

Bit	Field	Value	Description
5-0	LFSB		Channel causing LFS interrupt Group B. These bits contain the channel number of the pending interrupt for Group B if the corresponding interrupt enable is set. **Note: Reading this location clears the corresponding interrupt pending flag (see Section 16.3.1.32) with the highest priority.**
		0	No interrupt is pending.
		1h	Channel 0 is causing the pending interrupt Group B.
		:	:
		10h	Channel 15 is causing the pending interrupt Group B.
		11h-3Fh	Reserved

16.3.1.42 HBCB Interrupt Channel Offset Register (HBCBOFFSET)

Figure 16-58. HBCB Interrupt Channel Offset Register (HBCBOFFSET) [offset = 168h]

<table>
<tr><td>31</td><td>16</td></tr>
<tr><td colspan="2">Reserved</td></tr>
<tr><td colspan="2">R-0</td></tr>
</table>

<table>
<tr><td>15</td><td>8</td><td>7</td><td>6</td><td>5</td><td>0</td></tr>
<tr><td colspan="2">Reserved</td><td>sbz</td><td>sbz</td><td colspan="2">HBCB</td></tr>
<tr><td colspan="2">R-0</td><td>R-0</td><td>R-0</td><td colspan="2">R-0</td></tr>
</table>

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

Table 16-50. HBCB Interrupt Channel Offset Register (HBCBOFFSET) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-6	sbz	0	These bits should always be programmed as 0.
5-0	HBCB		Channel causing HBC interrupt Group B. These bits contain the channel number of the pending interrupt for Group B if the corresponding interrupt enable is set. **Note: Reading this location clears the corresponding interrupt pending flag (see Section 16.3.1.33) with the highest priority.**
		0	No interrupt is pending.
		1h	Channel 0 is causing the pending interrupt Group B.
		:	:
		10h	Channel 15 is causing the pending interrupt Group B.
		11h-3Fh	Reserved

16.3.1.43 BTCB Interrupt Channel Offset Register (BTCBOFFSET)

Figure 16-59. BTCB Interrupt Channel Offset Register (BTCBOFFSET) [offset = 16Ch]

<table>
<tr><td>31</td><td>16</td></tr>
<tr><td colspan="2">Reserved</td></tr>
<tr><td colspan="2">R-0</td></tr>
</table>

<table>
<tr><td>15</td><td>8</td><td>7</td><td>6</td><td>5</td><td>0</td></tr>
<tr><td colspan="2">Reserved</td><td>sbz</td><td>sbz</td><td colspan="2">BTCB</td></tr>
<tr><td colspan="2">R-0</td><td>R-0</td><td>R-0</td><td colspan="2">R-0</td></tr>
</table>

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

Table 16-51. BTCB Interrupt Channel Offset Register (BTCBOFFSET) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-6	sbz	0	These bits should always be programmed as 0.

Table 16-51. BTCB Interrupt Channel Offset Register (BTCBOFFSET) Field Descriptions (continued)

Bit	Field	Value	Description
5-0	BTCB		Channel causing BTC interrupt Group B. These bits contain the channel number of the pending interrupt for Group B if the corresponding interrupt enable is set. **Note: Reading this location clears the corresponding interrupt pending flag (see Section 16.3.1.34) with the highest priority.**
		0	No interrupt is pending.
		1h	Channel 0 is causing the pending interrupt Group B.
		:	:
		10h	Channel 15 is causing the pending interrupt Group B.
		11h-3Fh	Reserved

16.3.1.44 Port Control Register (PTCRL)

Figure 16-60. Port Control Register (PTCRL) [offset = 178h]

31 25	24
Reserved	PENDB
R-0	R-0

23 19	18	17	16
Reserved	BYB	PSFRHQPB	PSFRLQPB
R-0	R/WP-0	R/WP-0	R/WP-0

15 0
Reserved
R-0

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

Table 16-52. Port Control Register (PTCRL) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	PENDB		Transfers pending for Port B. This flag determines if transfers are ongoing on port B. The flag will be cleared if no transfers are performed. It can be used to determine if there is still data transferred while DMA_EN is set to 0 in GCTCRL. In this case, once all transfers are finished, the flag will be set to 0.
		0	No transfers are pending.
		1	Transfers are pending.
23-19	Reserved	0	Reads return 0. Writes have no effect.
18	BYB		Bypass FIFO B.
		0	FIFO B is not bypassed.
		1	FIFO B is bypassed. Writing 1 to this bit limits the FIFO depth to the size of one element. That means that after one element is read, the write-out to the destination will begin. This feature is particularly useful to minimize switching latency between channels. **Note: This feature does not make optimal use of bus bandwidth.**
17	PSFRHQPB		Priority scheme fix or rotate for high priority queue of Port B.
		0	Fixed priority is used.
		1	Rotation priority is used.
16	PSFRLQPB		Priority scheme fix or rotate for low priority queue of Port B.
		0	The fixed priority scheme is used.
		1	The rotation priority scheme is used.
15-0	Reserved	0	Reads return 0. Writes have no effect.

16.3.1.45 RAM Test Control (RTCTRL)

Figure 16-61. RAM Test Control (RTCTRL) [offset = 17Ch]

31 … 16
Reserved
R-0

15 … 1	0
Reserved	RTC
R-0	R/WP-0

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

Table 16-53. RAM Test Control (RTCTRL) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	RTC		RAM Test Control. Writing a 1 to this bit opens the write access to the reserved locations of control packet RAM as defined in the memory map.
			Note: This bit should be cleared to 0 during normal operation.
		0	RAM Test Control is disabled.
		1	RAM Test Control is enabled.

16.3.1.46 Debug Control (DCTRL)

Figure 16-62. Debug Control (DCTRL) [offset = 180h]

31 29	28 24	23 17	16
Reserved	CHNUM	Reserved	DMADBGS
R-0	R-0	R-0	R/W1C-0

15 1	0
Reserved	DBGEN
R-0	R/WC-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 16-54. Debug Control (DCTRL) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Reads return 0. Writes have no effect.
28-24	CHNUM	0-1Fh	Channel Number. This bit field indicates the channel number that causes the watch point to match.
23-17	Reserved	0	Reads return 0. Writes have no effect.
16	DMADBGS		DMA debug status. When a watch point is set up to watch for a unique bus address or a range of addresses is true on one of the three bus ports, then the DMA debug status bit is set to 1 and a debug request signal is asserted to the main CPU. The CPU must write a 1 to clear this bit for the DMA controller to release the debug request signal.
		0	Read: No watch point condition is detected. Write: No effect.
		1	Read: The watch point condition is detected. Write: The bit is cleared.
15-1	Reserved	0	Reads return 0. Writes have no effect.
0	DBGEN		Debug Enable. **Note: This bit can only be set when using a debugger.** **Note: This bit is reset when Test reset (nTRST) is low.**
		0	Debug is disabled.
		1	The watch point checking logics is enabled.

16.3.1.47 Watch Point Register (WPR)

Figure 16-63. Watch Point Register (WPR) [offset = 184h]

31	16
WP	
R/W-0	

15	0
WP	
R/W-0	

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

Table 16-55. Watch Point Register (WPR) Field Descriptions

Bit	Field	Value	Description
31-0	WP	0-FFFF FFFFh	Watch point. **Note: These bits can only be set when using a debugger.** **Note: This register is only reset by a when Test reset (nTRST).** A 32-bit address can be programmed into this register as a watch point. This register is used with the watch mask register (WMR). When the DBGEN bit in the DCTRL register is set and a unique address or a range of addresses are detected on the AHB address bus of Port B, a debug request signal is sent to the ARM CPU. The state machine of the port in which the watch point condition is true is frozen.

16.3.1.48 Watch Mask Register (WMR)

Figure 16-64. Watch Mask Register (WMR) [offset = 188h]

31	16
WM[31:16]	
R/W-0	

15	0
WM[15:0]	
R/W-0	

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

Table 16-56. Watch Mask Register (WMR) Field Descriptions

Bit	Field	Value	Description
31-0	WM[*n*]		Watch mask. **Note: These bits can only be set when using a debugger.** **Note: This register is only reset by a when Test reset (nTRST).**
		0	Allows the corresponding bit in the WPR register to be used for address matching for a watch point.
		1	Masks the corresponding bit in the WPR register and is disregarded in the comparison.

16.3.1.49 Port B Active Channel Source Address Register (PBACSADDR)

Figure 16-65. Port B Active Channel Source Address Register (PBACDADDR) [offset = 198h]

31 — 16
PBACSA
R-0

15 — 0
PBACSA
R-0

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

Table 16-57. Port B Active Channel Source Address Register (PBACDADDR) Field Descriptions

Bit	Field	Value	Description
31-0	PBACSA	0-FFFF FFFFh	Port B Active Channel Source Address. This register contains the current source address of the active channel as broadcasted in Section 16.3.1.3 for Port B.

16.3.1.50 Port B Active Channel Destination Address Register (PBACDADDR)

Figure 16-66. Port B Active Channel Destination Address Register (PBACDADDR) [offset = 19Ch]

31 — 16
PBACDA
R-0

15 — 0
PBACDA
R-0

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

Table 16-58. Port B Active Channel Destination Address Register (PBACDADDR) Field Descriptions

Bit	Field	Value	Description
31-0	PBACDA	0-FFFF FFFFh	Port B Active Channel Destination Address. This register contains the current destination address of the active channel as broadcasted in Section 16.3.1.3 for Port B.

16.3.1.51 Port B Active Channel Transfer Count Register (PBACTC)

Figure 16-67. Port B Active Channel Transfer Count Register (PBACTC) [offset = 1A0h]

31 29	28 16
Reserved	PBFTCOUNT
R-0	R-0

15 13	12 0
Reserved	PBETCOUNT
R-0	R-0

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

Table 16-59. Port B Active Channel Transfer Count Register (PBACTC) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Reads return 0. Writes have no effect.
28-16	PBFTCOUNT	0-1FFFh	Port B active channel frame count. These bits contain the current frame count value of the active channel as broadcasted in Section 16.3.1.3 for Port B.
15-13	Reserved	0	Reads return 0. Writes have no effect.
12-0	PBETCOUNT	0-1FFFh	Port B active channel element count. These bits contain the current element count value of the active channel as broadcasted in Section 16.3.1.3 for Port B.

16.3.1.52 Parity Control Register (DMAPCR)

Figure 16-68. Parity Control Register (DMAPCR) [offset = 1A8h]

31	15	16
Reserved		ERRA
R-0		R/WP-0

15	9	8	7	4	3	0
Reserved		TEST	Reserved		PARITY_ENA	
R-0		R/WP-0	R-0		R/WP-5h	

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

Table 16-60. Parity Control Register (DMAPCR) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reads return 0. Writes have no effect.
16	ERRA		Error action.
		0	If a parity error is detected on control packet x (x = 0, 1, ... n), then the enable/disable state of control packet x remains unchanged.
		1	If a parity error is detected on control packet x (x = 0, 1, ...n), then the DMA controller is disabled immediately. If a frame on control packet x is processed at the time the parity error is detected, then remaining elements of this frame will not be transferred anymore. The DMA will be disabled regardless of whether the error was detected during a read to the control packet RAM performed by the DMA state machine or by a different master.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	TEST		When this bit is set, the parity bits are memory mapped to make them accessible by the CPU.
		0	The parity bits are not memory mapped.
		1	The parity bits are memory mapped.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	PARITY_ENA		Parity error detection enable. This bit field enables or disables the parity check on read operations and the parity calculation on write operations. If parity checking is enabled and a parity error is detected, the DMA_UERR signal is activated.
		5h	The parity check is disabled.
		All other values	The parity check is enabled. **Note: It is recommended to write Ah to enable parity check, to guard against soft error from flipping PARITY_ENA to a disable state.**

16.3.1.53 DMA Parity Error Address Register (DMAPAR)

Figure 16-69. DMA Parity Error Address Register (DMAPAR) [offset = 1ACh]

31 25	24	23 16
Reserved	EDFLAG	Reserved
R-0	R/W1C-0	R-0

15 12	11 0
Reserved	ERRORADDRESS
R-0	R-X

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; X= Undefined; -*n* = value after reset

Table 16-61. DMA Parity Error Address Register (DMAPAR) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	EDFLAG		Parity Error Detection Flag. This flag indicates if a parity error occurred on reading DMA Control packet RAM.
		0	Read: No error occurred. Write: No effect.
		1	Read: Error detected and the address is captured in DMAPAR's ERROR_ADDRESS field Write: Clears the bit.
23-12	Reserved	0	Reads return 0. Writes have no effect.
11-0	ERRORADDRESS	0-FFFh	Error address. These bits hold the address of the first parity error generated in the RAM. This error address is frozen from being updated until it is read by the CPU. During emulation mode when SUSPEND is high, this address is frozen even when read. **Note: The error address register will not be reset by PORRST nor by any other reset source.**

16.3.1.54 DMA Memory Protection Control Register (DMAMPCTRL)

Figure 16-70. DMA Memory Protection Control Register (DMAMPCTRL) [offset = 1B0h]

31		29	28	27	26	25	24
Reserved			INT3AB	INT3ENA	REG3AP		REG3ENA
R-0			R/WP-0	R/WP-0	R/WP-0		R/WP-0

23		21	20	19	18	17	16
Reserved			INT2AB	INT2ENA	REG2AP		REG2ENA
R-0			R/WP-0	R/WP-0	R/WP-0		R/WP-0

15		13	12	11	10	9	8
Reserved			INT1AB	INT1ENA	REG1AP		REG1ENA
R-0			R/WP-0	R/WP-0	R/WP-0		R/WP-0

7		5	4	3	2	1	0
Reserved			INT0AB	INT0ENA	REG0AP		REG0ENA
R-0			R/WP-0	R/WP-0	R/WP-00		R/WP-0

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

Table 16-62. DMA Memory Protection Control Register (DMAMPCTRL) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Reads return 0. Writes have no effect.
28	INT3AB		Interrupt assignment of region 3 to Group A or Group B.
		0	The interrupt is routed to the VIM (Group A).
		1	The interrupt is routed to the second CPU (Group B).
27	INT3ENA		Interrupt enable of region 3.
		0	The interrupt is disabled.
		1	The interrupt is enabled.
26-25	REG3AP		Region 3 access permission. These bits determine the access permission for region 3.
		0	All accesses are allowed.
		1h	Read only accesses are allowed.
		2h	Write only accesses are allowed.
		3h	No accesses are allowed.
24	REG3ENA		Region 3 enable.
		0	The region is disabled (no address checking done).
		1	The region is enabled (address and access permission checking done).
23-21	Reserved	0	Reads return 0. Writes have no effect.
20	INT2AB		Interrupt assignment of region 2 to Group A or Group B.
		0	The interrupt is routed to the VIM (Group A).
		1	The interrupt is routed to the second CPU (Group B).
19	INT2ENA		Interrupt enable of region 2.
		0	The interrupt is disabled.
		1	The interrupt is enabled.
18-17	REG2AP		Region 2 access permission. These bits determine the access permission for region 2.
		0	All accesses are allowed.
		1h	Read only accesses are allowed.
		2h	Write only accesses are allowed.
		3h	No accesses are allowed.
16	REG2ENA		Region 2 enable.
		0	The region is disabled (no address checking done).
		1	The region is enabled (address and access permission checking done).

Table 16-62. DMA Memory Protection Control Register (DMAMPCTRL) Field Descriptions (continued)

Bit	Field	Value	Description
15-13	Reserved	0	Reads return 0. Writes have no effect.
12	INT1AB		Interrupt assignment of region 1 to Group A or Group B.
		0	The interrupt is routed to the VIM (Group A).
		1	The interrupt is routed to the second CPU (Group B).
11	INT1ENA		Interrupt enable of region 1.
		0	The interrupt is disabled.
		1	The interrupt is enabled.
10-9	REG1AP		Region 1 access permission. These bits determine the access permission for region 1.
		0	All accesses are allowed.
		1h	Read only accesses are allowed.
		2h	Write only accesses are allowed.
		3h	No accesses are allowed.
8	REG1ENA		Region 1 enable.
		0	The region is disabled (no address checking done).
		1	The region is enabled (address and access permission checking done).
7-5	Reserved	0	Reads return 0. Writes have no effect.
4	INT0AB		Interrupt assignment of region 0 to Group A or Group B.
		0	The interrupt is routed to the VIM (Group A).
		1	The interrupt is routed to the DSP CPU (Group B).
3	INT0ENA		Interrupt enable of region 0.
		0	The interrupt is disabled.
		1	The interrupt is enabled.
2-1	REG0AP		Region 0 access permission. These bits determine the access permission for region 0.
		0	All accesses are allowed.
		1h	Read only accesses are allowed.
		2h	Write only accesses are allowed.
		3h	No accesses are allowed.
0	REG0ENA		Region 0 enable.
		0	The region is disabled (no address checking done).
		1	The region is enabled (address and access permission checking done).

16.3.1.55 DMA Memory Protection Status Register (DMAMPST)

Figure 16-71. DMA Memory Protection Status Register (DMAMPST) [offset = 1B4h]

31 25	24	23 17	16
Reserved	REG3FT	Reserved	REG2FT
R-0	R/W1C-0	R-0	R/W1C-0

15 9	8	7 1	0
Reserved	REG1FT	Reserved	REG0FT
R-0	R/W1C-0	R-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 16-63. DMA Memory Protection Status Register (DMAMPST) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	REG3FT		Region 3 fault. This bit determines whether an access permission violation was detected in this region.
		0	Read: No fault was detected. Write: No effect.
		1	Read: A fault was detected. Write: The bit was cleared.
23-17	Reserved	0	Reads return 0. Writes have no effect.
16	REG2FT		Region 2 fault. This bit determines whether an access permission violation was detected in this region.
		0	Read: No fault was detected. Write: No effect.
		1	Read: A fault was detected. Write: The bit was cleared.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	REG1FT		Region 1 fault. This bit determines whether an access permission violation was detected in this region.
		0	Read: No fault was detected. Write: No effect.
		1	Read: A fault was detected. Write: The bit was cleared.
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	REG0FT		Region 0 fault. This bit determines whether an access permission violation was detected in this region.
		0	Read: No fault was detected. Write: No effect.
		1	Read: A fault was detected. Write: The bit was cleared.

16.3.1.56 DMA Memory Protection Region 0 Start Address Register (DMAMPR0S)

Figure 16-72. DMA Memory Protection Region 0 Start Address Register (DMAMPR0S) [offset = 1B8h]

31	16
STARTADDRESS	
R/WP-0	

15	0
STARTADDRESS	
R/WP-0	

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

Table 16-64. DMA Memory Protection Region 0 Start Address Register (DMAMPR0S) Field Descriptions

Bit	Field	Value	Description
31-0	STARTADDRESS	0-FFFF FFFFh	Start Address defines the address at which the region begins. The effective start address is truncated to the nearest word address, that is, 0x103 = 0x100.

16.3.1.57 DMA Memory Protection Region 0 End Address Register (DMAMPR0E)

Figure 16-73. DMA Memory Protection Region 0 End Address Register (DMAMPR0E) [offset = 1BCh]

31	16
ENDADDRESS	
R/WP-0	

15	0
ENDADDRESS	
R/WP-0	

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

Table 16-65. DMA Memory Protection Region 0 End Address Register (DMAMPR0E) Field Descriptions

Bit	Field	Value	Description
31-0	ENDADDRESS	0-FFFF FFFFh	End Address defines the address at which the region ends. The end address usually is larger than the start address for this region; otherwise, the region will wrap around at the end of the address space. The end address is the start address plus the region length minus 1. The effective end address is rounded up to the nearest 32-bit word end address, that is, 0x200 = 0x203. **Note: When using 64-bit transfers, the address is rounded up to the nearest 64-bit word end address, that is, 0x200 = 0x207. All other transfers are rounded up to the nearest 32-bit word end address.**

16.3.1.58 DMA Memory Protection Region 1 Start Address Register (DMAMPR1S)

Figure 16-74. DMA Memory Protection Region 1 Start Address Register (DMAMPR1S) [offset = 1C0h]

31 — 16
STARTADDRESS
R/WP-0

15 — 0
STARTADDRESS
R/WP-0

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

Table 16-66. DMA Memory Protection Region 1 Start Address Register (DMAMPR1S) Field Descriptions

Bit	Field	Value	Description
31-0	STARTADDRESS	0-FFFF FFFFh	Start Address defines the address at which the region begins. The effective start address is truncated to the nearest word address, that is, 0x103 = 0x100.

16.3.1.59 DMA Memory Protection Region 1 End Address Register (DMAMPR1E)

Figure 16-75. DMA Memory Protection Region 1 End Address Register (DMAMPR1E) [offset = 1C4h]

31 — 16
ENDADDRESS
R/WP-0

15 — 0
ENDADDRESS
R/WP-0

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

Table 16-67. DMA Memory Protection Region 1 End Address Register (DMAMPR1E) Field Descriptions

Bit	Field	Value	Description
31-0	ENDADDRESS	0-FFFF FFFFh	End Address defines the address at which the region ends. The end address usually is larger than the start address for this region; otherwise, the region will wrap around at the end of the address space. The end address is the start address plus the region length minus 1. The effective end address is rounded up to the nearest 32-bit word end address, that is, 0x200 = 0x203. **Note: When using 64-bit transfers, the address is rounded up to the nearest 64-bit word end address, that is, 0x200 = 0x207. All other transfers are rounded up to the nearest 32-bit word end address.**

16.3.1.60 DMA Memory Protection Region 2 Start Address Register (DMAMPR2S)

Figure 16-76. DMA Memory Protection Region 2 Start Address Register (DMAMPR2S) [offset = 1C8h]

31	16
STARTADDRESS	
R/WP-0	

15	0
STARTADDRESS	
R/WP-0	

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

Table 16-68. DMA Memory Protection Region 2 Start Address Register (DMAMPR2S) Field Descriptions

Bit	Field	Value	Description
31-0	STARTADDRESS	0-FFFF FFFFh	Start Address defines the address at which the region begins. The effective start address is truncated to the nearest word address, that is, 0x103 = 0x100.

16.3.1.61 DMA Memory Protection Region 2 End Address Register (DMAMPR2E)

Figure 16-77. DMA Memory Protection Region 2 End Address Register (DMAMPR2E) [offset = 1CCh]

31	16
ENDADDRESS	
R/WP-0	

15	0
ENDADDRESS	
R/WP-0	

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

Table 16-69. DMA Memory Protection Region 2 End Address Register (DMAMPR2E) Field Descriptions

Bit	Field	Value	Description
31-0	ENDADDRESS	0-FFFF FFFFh	End Address defines the address at which the region ends. The end address usually is larger than the start address for this region; otherwise, the region will wrap around at the end of the address space. The end address is the start address plus the region length minus 1. The effective end address is rounded up to the nearest 32-bit word end address, that is, 0x200 = 0x203. **Note: When using 64-bit transfers, the address is rounded up to the nearest 64-bit word end address, that is, 0x200 = 0x207. All other transfers are rounded up to the nearest 32-bit word end address.**

16.3.1.62 DMA Memory Protection Region 3 Start Address Register (DMAMPR3S)

Figure 16-78. DMA Memory Protection Region 3 Start Address Register (DMAMPR3S) [offset = 1D0h]

31	16
STARTADDRESS	
R/WP-0	

15	0
STARTADDRESS	
R/WP-0	

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

Table 16-70. DMA Memory Protection Region 3 Start Address Register (DMAMPR3S) Field Descriptions

Bit	Field	Value	Description
31-0	STARTADDRESS	0-FFFF FFFFh	Start Address defines the address at which the region begins. The effective start address is truncated to the nearest word address, that is, 0x103 = 0x100.

16.3.1.63 DMA Memory Protection Region 3 End Address Register (DMAMPR3E)

Figure 16-79. DMA Memory Protection Region 3 End Address Register (DMAMPR3E) [offset = 1D4h]

31	16
ENDADDRESS	
R/WP-0	

15	0
ENDADDRESS	
R/WP-0	

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

Table 16-71. DMA Memory Protection Region 3 End Address Register (DMAMPR3E) Field Descriptions

Bit	Field	Value	Description
31-0	ENDADDRESS	0-FFFF FFFFh	End Address defines the address at which the region ends. The end address usually is larger than the start address for this region; otherwise, the region will wrap around at the end of the address space. The end address is the start address plus the region length minus 1. The effective end address is rounded up to the nearest 32-bit word end address, that is, 0x200 = 0x203. **Note: When using 64-bit transfers, the address is rounded up to the nearest 64-bit word end address, that is, 0x200 = 0x207. All other transfers are rounded up to the nearest 32-bit word end address.**

16.3.2 Channel Configuration

The channel configuration is defined by the channel control packet: channel control, transfer count, offset values, source/destination address.

- It is stored in local RAM, which is protected by parity.
- Each control packet contains a total of nine fields.
- The first six fields are programmable, while the last three fields are read only.
- The RAM is accessible by queue A and queue B state machines as well as CPU.
- When there are simultaneous accesses, the priority is resolved in a fixed priority scheme with the CPU having the highest priority.

All the control packets look the same. Following, there is the detailed layout of these registers shown for control packet 0.

16.3.2.1 Initial Source Address (ISADDR)

Figure 16-80. Initial Source Address (ISADDR) [offset = 00]

31	16
ISADDR	
R/WP-X	

15	0
ISADDR	
R/WP-X	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset; X = Unknown

Table 16-72. Initial Source Address (ISADDR) Field Descriptions

Bit	Field	Value	Description
31-0	ISADDR	0-FFFF FFFFh	Initial source address. These bits give the absolute 32-bit source address (physical).

16.3.2.2 Initial Destination Address Register (IDADDR)

Figure 16-81. Initial Destination Address Register (IDADDR) [offset = 04h]

31	16
IDADDR	
R/WP-X	

15	0
IDADDR	
R/WP-X	

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset; X = Unknown

Table 16-73. Initial Destination Address Register (IDADDR) Field Descriptions

Bit	Field	Value	Description
31-0	IDADDR	0-FFFF FFFFh	Initial destination address. These bits give the absolute 32-bit destination address (physical).

16.3.2.3 Initial Transfer Count Register (ITCOUNT)

Figure 16-82. Initial Transfer Count Register (ITCOUNT) [offset = 08h]

31 — 29	28 — 16
Reserved	IFTCOUNT
R-X	R/WP-X

15 — 13	12 — 0
Reserved	IETCOUNT
R-X	R/WP-X

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset; X = Unknown

Table 16-74. Initial Transfer Count Register (ITCOUNT) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Reads are undefined. Writes have no effect.
28-16	IFTCOUNT	0-1FFFh	Initial frame transfer count. These bits define the number of frame transfers.
15-13	Reserved	0	Reads are undefined. Writes have no effect.
12-0	IETCOUNT	0-1FFFh	Initial element transfer count. These bits define the number of element transfers. The block transfer size will be IETCOUNT x IFTCOUNT

16.3.2.4 Channel Control Register (CHCTRL)

Figure 16-83. Channel Control Register (CHCTRL) [offset = 10h]

31 — 22	21 — 16
Reserved	CHAIN
R-X	R/WP-X

15 — 14	13 — 12	11 — 9	8	7 — 5	4 — 3	2 — 1	0
RES	WES	Reserved	TTYPE	Reserved	ADDMR	ADDMW	AIM
R/WP-X	R/WP-X	R-X	R/WP-X	R-X	R/WP-X	R/WP-X	R/WP-X

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset; X = Unknown

Table 16-75. Channel Control Register (CHCTRL) Field Descriptions

Bit	Field	Value	Description
31-22	Reserved	0	Reads are undefined. Writes have no effect.
21-16	CHAIN		Next channel to be triggered. At the end of the programmed number of frames, the specified channel will be triggered. Programmed value x means channel (x-1) is chained.
			Note: The programmer must program the CHAIN bits before initiating a DMA transfer.
		0	No channel is selected.
		1h	Channel 0 is selected.
		:	:
		10h	Channel 15 is selected.
		11h-3Fh	Reserved
15-14	RES		Read element size.
		0	The element is byte, 8-bit.
		1h	The element is half-word, 16-bit.
		2h	The element is word, 32-bit.
		3h	The element is double-word, 64-bit.
13-12	WES		Write element size.
		0	The element is byte, 8-bit.
		1h	The element is half-word, 16-bit.
		2h	The element is word, 32-bit.
		3h	The element is double-word, 64-bit.
11-9	Reserved	0	Reads are undefined. Writes have no effect.
8	TTYPE		Transfer type.
		0	A request triggers one frame transfer.
		1	A request triggers one block transfer.
7-5	Reserved	0	Reads are undefined. Writes have no effect.
4-3	ADDMR		Addressing mode read.
		0	Constant
		1h	Post-increment
		2h	Reserved
		3h	Indexed
2-1	ADDMW		Addressing mode write.
		0	Constant
		1h	Post-increment
		2h	Reserved
		3h	Indexed
0	AIM		Auto-initiation mode.
		0	Auto-initiation mode is disabled.
		1	Auto-initiation mode is enabled.

16.3.2.5 Element Index Offset Register (EIOFF)

Figure 16-84. Element Index Offset Register (EIOFF) [offset = 14h]

31-29	28-16
Reserved	EIDXD
R-X	R/WP-X

15-13	12-0
Reserved	EIDXS
R-X	R/WP-X

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset; X = Unknown

Table 16-76. Element Index Offset Register (EIOFF) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Reads are undefined. Writes have no effect.
28-16	EIDXD	0-1FFFh	Destination address element index. These bits define the offset to be added to the destination address after each element transfer.
15-13	Reserved	0	Reads are undefined. Writes have no effect.
12-0	EIDXS	0-1FFFh	Source address element index. These bits define the offset to be added to the source address after each element transfer.

16.3.2.6 Frame Index Offset Register (FIOFF)

Figure 16-85. Frame Index Offset Register (FIOFF) [offset = 18h]

31-29	28-16
Reserved	FIDXD
R-X	R/WP-X

15-13	12-0
Reserved	FIDXS
R-X	R/WP-X

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset; X = Unknown

Table 16-77. Frame Index Offset Register (FIOFF) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Reads are undefined. Writes have no effect.
28-16	FIDXD	0-1FFFh	Destination address frame index. These bits define the offset to be added to the destination address after element count reached 1.
15-13	Reserved	0	Reads are undefined. Writes have no effect.
12-0	FIDXS	0-1FFFh	Source address frame index. These bits define the offset to be added to the source address after element count reached 1.

16.3.2.7 Current Source Address Register (CSADDR)

Figure 16-86. Current Source Address Register (CSADDR) [offset = 800h]

31	16
CSADDR	
R-X	

15	0
CSADDR	
R-X	

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

Table 16-78. Current Source Address Register (CSADDR) Field Descriptions

Bit	Field	Value	Description
31-0	CSADDR	0-FFFF FFFFh	Current source address. These bits contain the current working absolute 32-bit source address (physical). These bits are only updated after a channel is arbitrated out from the priority queue.

16.3.2.8 Current Destination Address Register (CDADDR)

Figure 16-87. Current Destination Address Register (CDADDR) [offset = 804h]

31	16
CDADDR	
R-X	

15	0
CDADDR	
R-X	

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

Table 16-79. Current Destination Address Register (CDADDR) Field Descriptions

Bit	Field	Value	Description
31-0	CDADDR	0-FFFF FFFFh	Current destination address. These bits contain the current working absolute 32-bit destination address (physical). These bits are only updated after a channel is arbitrated out of the priority queue.

16.3.2.9 Current Transfer Count Register (CTCOUNT)

Figure 16-88. Current Transfer Count Register (CTCOUNT) [offset = 808h]

31 29	28 16
Reserved	CFTCOUNT
R-X	R-X

15 13	12 0
Reserved	CETCOUNT
R-X	R-X

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

Table 16-80. Current Transfer Count Register (CTCOUNT) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Reads are undefined. Writes have no effect.
28-16	CFTCOUNT	0-1FFFh	Current frame transfer count. Returned the current remaining frame counts.
15-13	Reserved	0	Reads are undefined. Writes have no effect.
12-0	CETCOUNT	0-1FFFh	Current element transfer count. These bits return the current remaining element counts. CTCOUNT register is only updated after a channel is arbitrated out of the priority queue.

External Memory Interface (EMIF)

This chapter describes the external memory Interface (EMIF).

Topic **Page**

17.1 Introduction 649
17.2 EMIF Module Architecture 651
17.3 EMIF Registers 683
17.4 Example Configuration 695

17.1 Introduction

17.1.1 Purpose of the Peripheral

This EMIF memory controller is compliant with the JESD21-C SDR SDRAM memories utilizing a 16-bit data bus. The purpose of this EMIF 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 and SRAM

The most common use for the EMIF is to interface with both a flash device and an SDRAM device simultaneously. Section 17.4 contains an example of operating the EMIF in this configuration.

17.1.2 Features

The EMIF includes many features to enhance the ease and flexibility of connecting to external SDR SDRAM and asynchronous devices.

17.1.2.1 Asynchronous Memory Support

EMIF supports asynchronous:

- SRAM memories
- NOR Flash memories

The EMIF data bus width is up to 16 bits and there are up to 13 address lines. There is an external wait input that allows slower asynchronous memories to extend the memory access. The EMIF module supports up to 3 chip selects (EMIF_nCS[4:2]). Each chip select has the following individually programmable attributes:

- Data Bus Width
- Read cycle timings: setup, hold, strobe
- Write cycle timings: setup, hold, strobe
- Bus turn-around time
- Extended Wait Option with Programmable Timeout
- Select Strobe option

17.1.2.2 Synchronous DRAM Memory Support

The EMIF module supports 16-bit SDRAM in addition to the asynchronous memories listed in Section 17.1.2.1. It has a single SDRAM chip select (EMIF_nCS[0]). SDRAM configurations that are supported are:

- One, Two, and Four Bank SDRAM devices
- Devices with Eight, Nine, Ten, and Eleven Column Address
- CAS Latency of two or three clock cycles
- 16-bit Data Bus Width
- 3.3V LVCMOS Interface

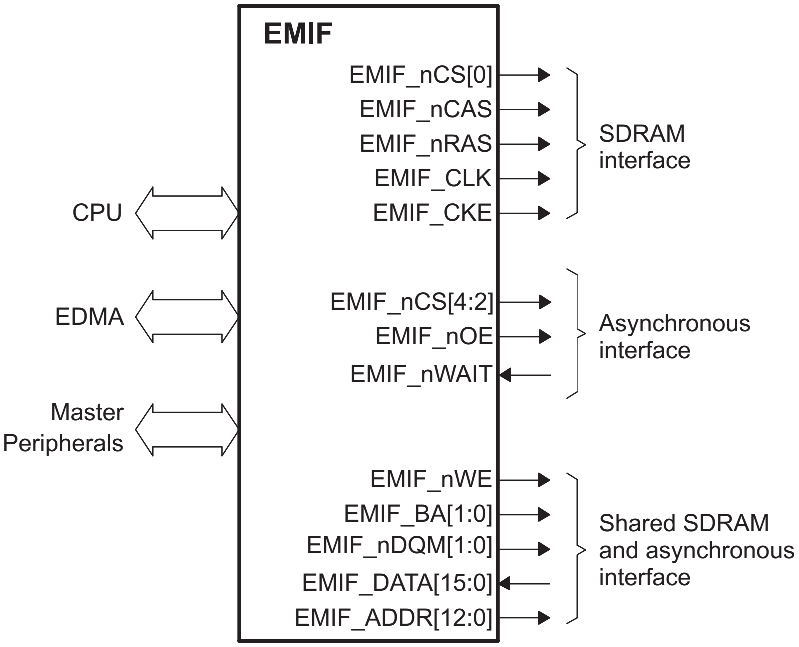
Additionally, the EMIF supports placing the SDRAM in Self-Refresh and Powerdown modes. Self-refresh mode allows the SDRAM to be put in a low-power state while still retaining memory contents; since the SDRAM will continue to refresh itself even without clocks from the microcontroller. Powerdown mode achieves even lower power, except the microcontroller must periodically wake up and issue refreshes if data retention is required.

Note that the EMIF module does not support Mobile SDRAM devices.

17.1.3 Functional Block Diagram

Figure 17-1 illustrates the connections between the EMIF and its internal requesters, along with the external EMIF pins. Section 17.2.2 contains a description of the entities internal to the SoC that can send requests to the EMIF, along with their prioritization. Section 17.2.3 describes the EMIF external pins and summarizes their purpose when interfacing with SDRAM and asynchronous devices.

Figure 17-1. EMIF Functional Block Diagram



17.2 EMIF Module Architecture

This section provides details about the architecture and operation of the EMIF. Both, SDRAM and asynchronous Interface are covered, along with other system-related issues such as clock control.

17.2.1 EMIF Clock Control

The EMIF clock is output on the EMIF_CLK pin and should be used when interfacing to external SDRAM devices. The EMIF module gets the VCLK3 clock domain as the input. This clock domain is running at half the frequency of the main oscillator by default, that is, between 2.5MHz to 10MHz. The VCLK3 frequency is divided down from the HCLK domain frequency by a programmable divider (/1 to /16). Refer the Architecture chapter of the device technical reference manual for more information on configuring the VCLK3 domain frequency.

17.2.2 EMIF Requests

Different sources within the SoC can make requests to the EMIF. These requests consist of accesses to SDRAM memory, asynchronous memory, and EMIF registers. The EMIF can process only one request at a time. Therefore a high performance crossbar switch exists within the SoC to provide prioritized requests from the different sources to the EMIF. The sources are:

1. CPU
2. DMA
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 EMIF first. Upon completion of a request, the crossbar switch again evaluates the pending requests and forwards the highest priority pending request to the EMIF.

When the EMIF receives a request, it may or may not be immediately processed. In some cases, the EMIF will perform one or more auto refresh cycles before processing the request. For details on the EMIF's internal arbitration between performing requests and performing auto refresh cycles, see Section 17.2.13.

17.2.3 EMIF Signal Descriptions

This section describes the function of each of the EMIF signals.

Table 17-1. EMIF Pins Used to Access Both SDRAM and Asynchronous Memories

Pins(s)	I/O	Description
EMIF_DATA[15:0]	I/O	**EMIF data bus.**
EMIF_ADDR[12:0]	O	**EMIF address bus.** 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 Table 17-13. EMIF_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 EMIF_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 17.2.6.1.
EMIF_BA[1:0]	O	**EMIF 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 inTable 17-13. When interfacing to an asynchronous device, these pins are used in conjunction with the EMIF_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 17.2.6.1.
EMIF_nDQM[1:0]	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 17.2.6 for details.

Table 17-1. EMIF Pins Used to Access Both SDRAM and Asynchronous Memories (continued)

Pins(s)	I/O	Description
EMIF_nWE	O	**Active-low write enable.** When interfacing to SDRAM, this pin is connected to the nWE 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.

Table 17-2. EMIF Pins Specific to SDRAM

Pin(s)	I/O	Description
EMIF_nCS[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 EMIF keeps this SDRAM chip select active, even if the EMIF 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 access.
EMIF_nRAS	O	**Active-low row address strobe pin.** This pin is connected to the nRAS pin of the attached SDRAM device and is used for sending commands to the device.
EMIF_nCAS	O	**Active-low column address strobe pin.** This pin is connected to the nCAS pin of the attached SDRAM device and is used for sending commands to the device.
EMIF_CKE	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 17.2.5.7 for details.
EMIF_CLK	O	**SDRAM clock pin.** This pin is connected to the CLK pin of the attached SDRAM device. See Section 17.2.1 for details on the clock signal.

Table 17-3. EMIF Pins Specific to Asynchronous Memory

Pin(s)	I/O	Description
EMIF_nCS[4:2]	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.
EMIF_nWAIT	I	**Wait input with programmable polarity.** A connected asynchronous device can extend the strobe period of an access cycle by asserting the EMIF_nWAIT input to the EMIF as described in Section 17.2.6.6. To enable this functionality, the EW bit in the asynchronous 1 configuration register (CE2CFG) must be set to 1. In addition, the WP0 bit in CE2CFG must be configured to define the polarity of the EMIF_nWAIT pin.
EMIF_nOE	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.

17.2.4 EMIF Signal Multiplexing Control

Several EMIF signals are multiplexed with other functions on this microcontroller. Please refer to the I/O Multiplexing Module chapter of the technical reference manual for more information on how to enable the output of these EMIF signals.

17.2.5 SDRAM Controller and Interface

The EMIF 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 EMIF to perform read and write operations to externally connected SDR SDRAM devices. Also, Section 17.4 provides a detailed example of interfacing the EMIF to a common SDRAM device.

17.2.5.1 SDRAM Commands

The EMIF supports the SDRAM commands described in Table 17-4. Table 17-5 shows the truth table for the SDRAM commands, and an example timing waveform of the PRE command is shown in Figure 17-2. EMIF_A[10] is pulled low in this example to deactivate only the bank specified by the EMIF_BA pins.

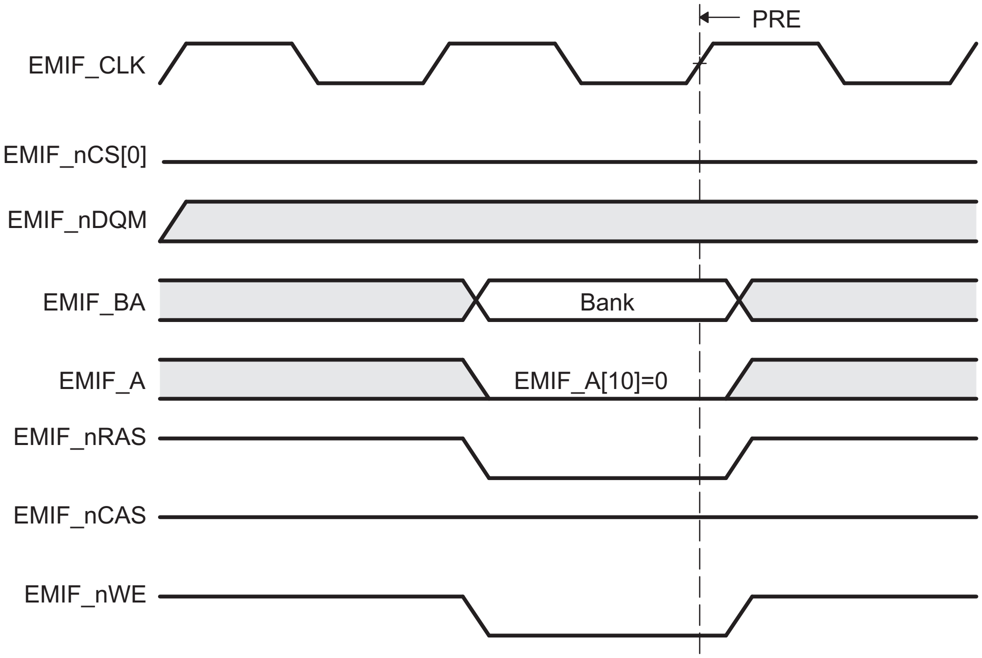
Table 17-4. EMIF SDRAM Commands

Command	Function
PRE	**Precharge.** Depending on the value of EMIF_A[10], the PRE command either deactivates the open row in all banks (EMIF_A[10] = 1) or only the bank specified by the EMIF_BA[1:0] pins (EMIF_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 EMIF_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 EMIF_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 17.2.5.4.
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 17-5. Truth Table for SDRAM Commands

SDRAM Pins:	CKE	nCS	nRAS	nCAS	nWE	BA[1:0]	A[12:11]	A[10]	A[9:0]
EMIF Pins:	EMIF_CKE	EMIF_nCS[0]	EMIF_nRAS	EMIF_nCAS	EMIF_nWE	EMIF_BA[1:0]	EMIF_A[12:11]	EMIF_A[10]	EMIF_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

Figure 17-2. Timing Waveform of SDRAM PRE Command



17.2.5.2 Interfacing to SDRAM

The EMIF 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.
- The number of row address bits is 13, 14, 15, or 16.
- The number of internal banks is 1, 2, or 4.

Figure 17-3 shows an interface between the EMIF and a 2M × 16 × 4 bank SDRAM device, and Figure 17-4 shows an interface between the EMIF and a 512K × 16 × 2 bank SDRAM device. For devices supporting 16-bit interface, refer to Table 17-6 for list of commonly-supported SDRAM devices and the required connections for the address pins.

Figure 17-3. EMIF to 2M × 16 × 4 bank SDRAM Interface

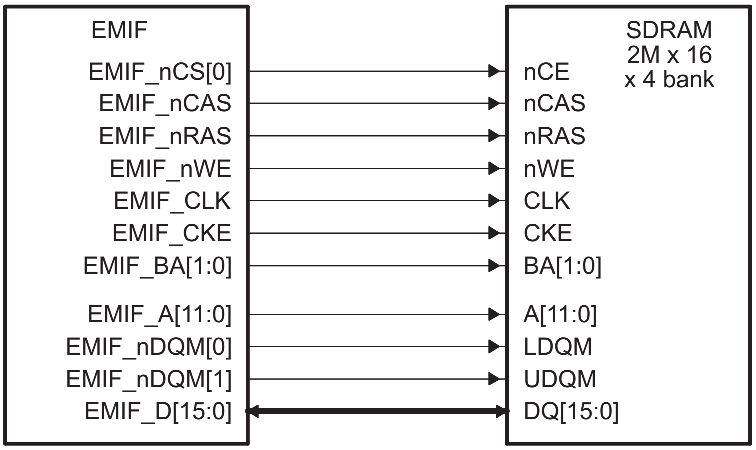


Figure 17-4. EMIF to 512K × 16 × 2 bank SDRAM Interface

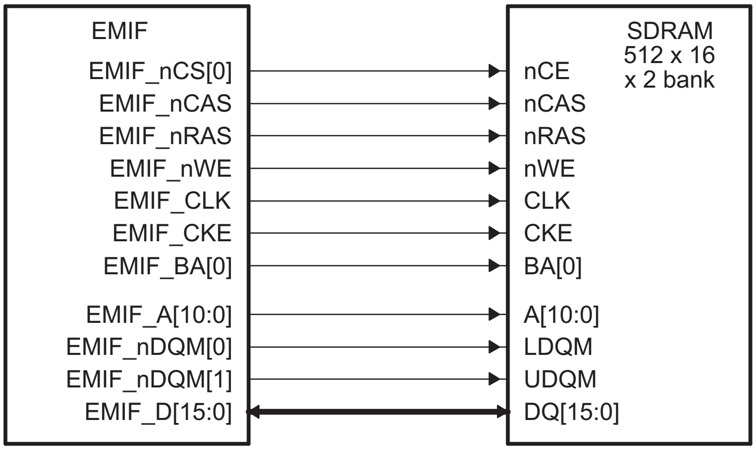


Table 17-6. 16-bit EMIF Address Pin Connections

SDRAM Size	Width	Banks	Device	Address Pins
16M bits	×16	2	SDRAM	A[10:0]
			EMIF	EMIF_A[10:0]
64M bits	×16	4	SDRAM	A[11:0]
			EMIF	EMIF_A[11:0]
128M bits	×16	4	SDRAM	A[11:0]
			EMIF	EMIF_A[11:0]
256M bits	x16	4	SDRAM	A[12:0]
			EMIF	EMIF_A[12:0]
512M bits	x16	4	SDRAM	A[12:0]
			EMIF	EMIF_A[12:0]

17.2.5.3 SDRAM Configuration Registers

The operation of the EMIF's SDRAM interface is controlled by programming the appropriate configuration registers. This section describes the purpose and function of each configuration register, but Section 17.3 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 EMIF to abandon whatever it is currently doing and trigger the SDRAM initialization procedure described in Section 17.2.5.4.

Table 17-7. 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 EMIF will go into Self Refresh.
PDWR	Perform refreshes during Power Down. Writing a 1 to this bit will cause the EMIF 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 EMIF 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 17.2.5.4. 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 17.2.5.11 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 17.2.5.11 for details.

Table 17-8. 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_{EMIF_CLK} / (Required SDRAM Refresh Rate) More information about the operation of the SDRAM refresh controller can be found in Section 17.2.5.6.

Table 17-9. 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 EMIF to comply with the AC timing requirements of the attached SDRAM devices. This allows the EMIF 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 17.3.6. These parameters should be set to satisfy the corresponding timing requirements found in the SDRAM's datasheet.

Table 17-10. 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 EMIF about the minimum number of EMIF_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.

17.2.5.4 SDRAM Auto-Initialization Sequence

The EMIF 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 EMIF 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 EMIF issues a PRE command with EMIF_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 EMIF drives EMIF_CKE 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 EMIF_CLK (RR/f_{EMIF_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 EMIF_CLK, this step may or may not be sufficient to avoid violating the SDRAM constraint. See Section 17.2.5.5 for more information.
3. After the refresh intervals have elapsed, the EMIF issues a PRE command with EMIF_A[10] held high to indicate all banks.
4. The EMIF issues eight AUTO REFRESH commands.
5. The EMIF issues the LMR command with the EMIF_A[9:0] pins set as described in Table 17-11.
6. Finally, the EMIF performs a refresh cycle, which consists of the following steps:
   a. Issuing a PRE command with EMIF_A[10] held high if any banks are open
   b. Issuing an REF command

Table 17-11. SDRAM LOAD MODE REGISTER Command

EMIF_A[9:7]	EMIF_A[6:4]	EMIF_A[3]	EMIF_A[2:0]
0 (Write bursts are of the programmed burst length in EMIF_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, EMIF_A[6:4] = 2h (CAS latency = 2) • If CL = 3, EMIF_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, EMIF_A[2:0] = 2h (Burst Length = 4) • If NM = 1, EMIF_A[2:0] = 3h (Burst Length = 8)

17.2.5.5 SDRAM Configuration Procedure

There are two different SDRAM configuration procedures. Although EMIF automatically performs the SDRAM initialization sequence described in Section 17.2.5.4 when coming out of reset, it is recommended to follow one of the procedures listed below before performing any EMIF 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 17.2.5.4 on coming out of Reset. The SDRAM Power-up constraint specifies that 200 μs (sometimes 100 μs) should exist 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 EMIF_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 not 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 17.2.5.4. The SDRAM should be placed into Self-Refresh mode when changing the frequency of EMIF_CLK to avoid incurring the 200 μs Power-up constraint again.
2. Configure the desired EMIF_CLK clock frequency. 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 datasheet.
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 17.2.5.4.
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 17.2.5.6.1 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 17.2.5.4 to be re-run. This second initialization generally takes much less time due to the increased frequency of EMIF_CLK.

Procedure B — Following is the procedure to be followed if the SDRAM Power-up constraint was violated:

1. Configure the desired EMIF_CLK clock frequency. 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 datasheet.
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.
3. Program the RR field of SDRCR such that the following equation is satisfied: $(RR \times 8)/(f_{EMIF\_CLK}) >$ 200 μs (sometimes 100 μs). For example, an EMIF_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 17.2.5.4 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 17.2.5.6.1 details on determining the appropriate value.

After following the above procedure, the EMIF is ready to perform accesses to the attached SDRAM device. See Section 17.4 for an example of configuring the SDRAM interface.

17.2.5.6 EMIF Refresh Controller

An SDRAM device requires that each of its rows be refreshed at a minimum required rate. The EMIF 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 EMIF of the required rate for performing auto refresh cycles, the RR field of the SDRAM refresh control register (SDRCR) must be programmed. The EMIF 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 EMIF is not interfaced with an SDRAM. The remainder of this section details the EMIF'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 EMIF 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 EMIF 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 EMIF 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 17-12. This refresh scheme allows the required refreshes to be performed with minimal impact on access requests.

Table 17-12. Refresh Urgency Levels

Urgency Level	Refresh Backlog Counter Range	Action Taken
Refresh May	1-3	An auto-refresh cycle is performed only if the EMIF has no requests pending and none of the SDRAM banks are open.
Refresh Release	4-7	An auto-refresh cycle is performed if the EMIF 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 EMIF can begin servicing any new read or write requests.

17.2.5.6.1 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 EMIF_CLK signal (f_{EMIF_CLK}) and the required refresh rate of the SDRAM ($f_{Refresh}$). The following formula can be used:

$RR = f_{EMIF\_CLK} / f_{Refresh}$

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:

$RR = f_{EMIF\_CLK} \times t_{Refresh\ Period} / n_{cycles}$

The following example illustrates calculating the value of RR. Given that:

- f_{EMIF_CLK} = 100 MHz (frequency of the EMIF 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

17.2.5.7 Self-Refresh Mode

The EMIF can be programmed to enter the self-refresh state by setting the SR bit of SDCR to 1. This will cause the EMIF 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 EMIF continues to service asynchronous bank requests and register accesses as normal, with one caveat. The EMIF will not park the data bus following a read to asynchronous memory while in the self-refresh state. Instead, the EMIF tri-states the data bus. Therefore, it is not recommended to perform asynchronous read operations while the EMIF 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 17.2.7.

The EMIF 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 EMIF exits from the self-refresh state by driving EMIF_CKE high and performing an auto refresh cycle.

The attached SDRAM device should also be placed into Self-Refresh Mode when changing the frequency of EMIF_CLK. If the frequency of EMIF_CLK changes while the SDRAM is not in Self-Refresh Mode, Procedure B in Section 17.2.5.5 should be followed to reinitialize the device.

17.2.5.8 Power Down Mode

To support low-power modes, the EMIF 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 EMIF 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 EMIF will enter the power-down state. Upon entering this state, the EMIF will issue a POWER DOWN command (same as a NOP command but driving EMIF_CKE low on the same cycle). The EMIF then maintains EMIF_CKE low until it exits the power-down state.

Since the EMIF 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 EMIF only supports Precharge Power Down. The EMIF 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 EMIF 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 EMIF should perform refreshes in power-down state. If the PDWR bit is set, the EMIF 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 EMIF 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 EMIF will come out of the power-down state. The EMIF:

- Drives EMIF_CKE high.
- Enters its idle state.

17.2.5.9 SDRAM Read Operation

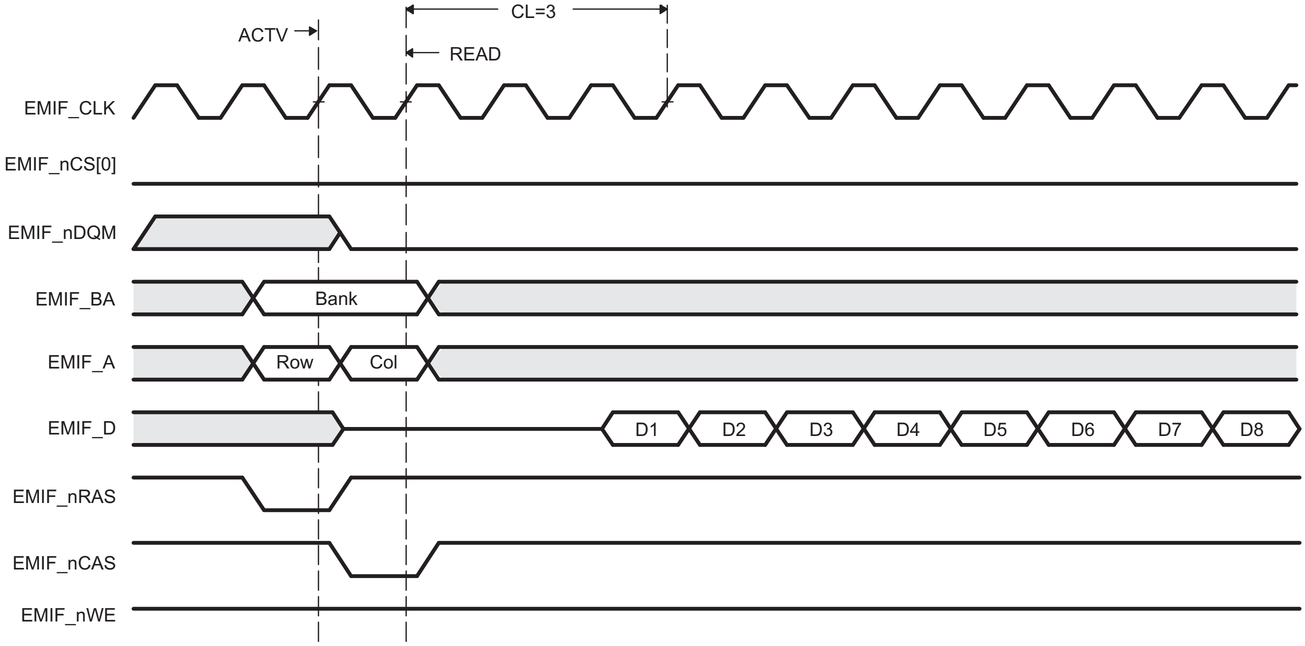
When the EMIF receives a read request to SDRAM from one of the requesters listed in Section 17.2.2, it performs one or more read access cycles. A read access cycle begins with the issuing of the ACTV command to select the desired bank and row of the SDRAM device. After the row has been opened, the EMIF proceeds to issue a READ command while specifying the desired bank and column address. EMIF_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 EMIF 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 17-5 shows the signal waveforms for a basic SDRAM read operation in which a burst of data is read from a single page. When the EMIF 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 17-5 shows a burst size of eight.

The EMIF will truncate a series of bursting data if the remaining addresses of the burst are not required to complete the request. The EMIF 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 17-5. Timing Waveform for Basic SDRAM Read Operation



Several other pins are also active during a read access. The EMIF_nDQM[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 EMIF pins during each command can be found in Table 17-5.

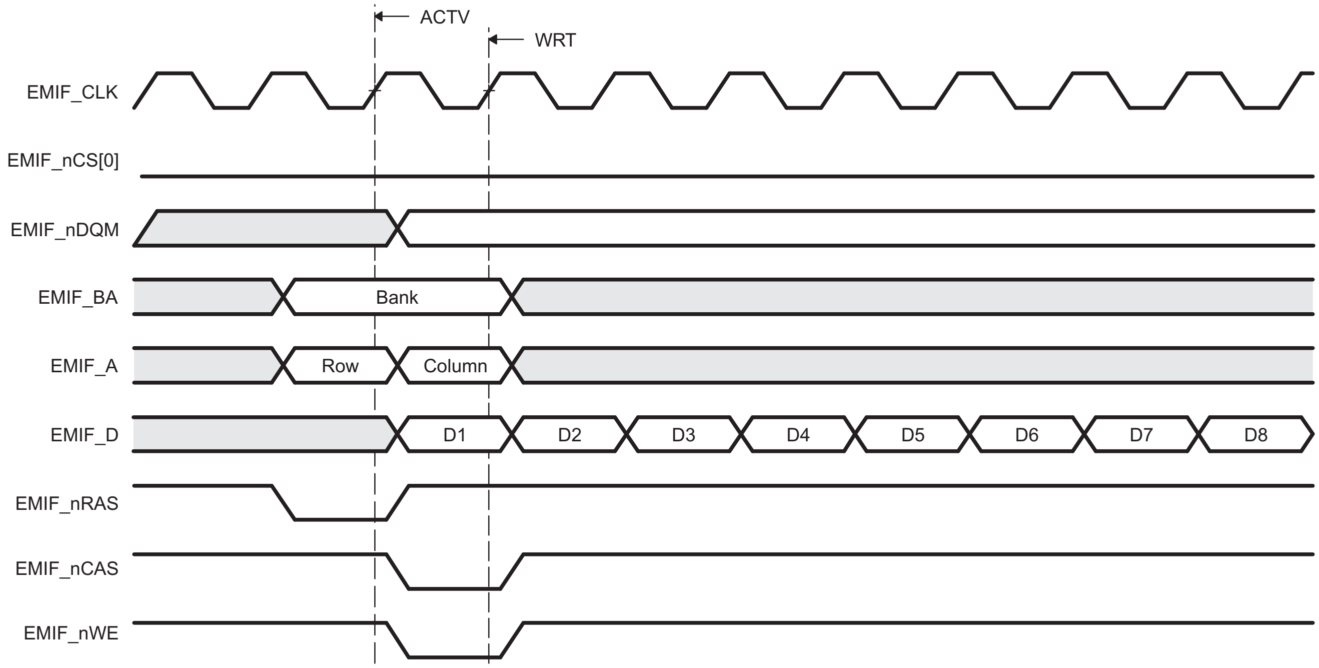
The EMIF schedules its commands based on the timing information that is provided to it in the SDRAM timing register (SDTIMR). The values for the timing parameters in this register should be chosen to satisfy the timing requirements listed in the SDRAM datasheet. The EMIF uses this timing information 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 17.3.6 for more details on the various timing parameters.

17.2.5.10 SDRAM Write Operations

When the EMIF receives a write request to SDRAM from one of the requesters listed in Section 17.2.2, 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 EMIF proceeds to issue a WRT command while specifying the desired bank and column address. EMIF_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 EMIF 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 17-6 shows the signal waveforms for a basic SDRAM write operation in which a burst of data is read from a single page. When the EMIF 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 17-6 shows a burst size of eight.

Figure 17-6. Timing Waveform for Basic SDRAM Write Operation



The EMIF will truncate a series of bursting data if the remaining addresses of the burst are not part of the write request. The EMIF 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 EMIF_nDQM[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 EMIF pins during each command can be found in Table 17-5.

The EMIF 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 EMIF 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 17.3.6 for more details on the various timing parameters.

17.2.5.11 Mapping from Logical Address to EMIF Pins

When the EMIF 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 17-13 for 16-bit operation. Using the settings of the IBANK and PAGESIZE fields of the SDRAM configuration register (SDCR), the EMIF determines which bits of the logical address will be 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 EMIF moves into the same page in the next bank of the attached device by incrementing the bank address EMIF_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 EMIF uses the EMIF_nDQM[1:0] pins during a WRT command to mask out selected bytes or entire words. The EMIF_nDQM[1:0] pins are always low during a READ command.

Table 17-13. Mapping from Logical Address to EMIF Pins for 16-bit SDRAM

<table>
<tr><th rowspan="2">IBANK</th><th rowspan="2">PAGESIZE</th><th colspan="6">Logical Address</th></tr>
<tr><th>12</th><th>11</th><th>10</th><th>9</th><th>8:1</th><th>0</th></tr>
<tr><td>0</td><td>0</td><td colspan="4">Row Address</td><td>Column Address</td><td>EMIF_nDQM[0]</td></tr>
<tr><td>1</td><td>0</td><td colspan="3">Row Address</td><td>EMIF_BA[0]</td><td>Column Address</td><td>EMIF_nDQM[0]</td></tr>
<tr><td>2</td><td>0</td><td colspan="2">Row Address</td><td colspan="2">EMIF_BA[1:0]</td><td>Column Address</td><td>EMIF_nDQM[0]</td></tr>
<tr><td>0</td><td>1</td><td colspan="3">Row Address</td><td colspan="2">Column Address</td><td>EMIF_nDQM[0]</td></tr>
<tr><td>1</td><td>1</td><td colspan="2">Row Address</td><td>EMIF_BA[0]</td><td colspan="2">Column Address</td><td>EMIF_nDQM[0]</td></tr>
<tr><td>2</td><td>1</td><td>Row Address</td><td colspan="2">EMIF_BA[1:0]</td><td colspan="2">Column Address</td><td>EMIF_nDQM[0]</td></tr>
<tr><td>0</td><td>2</td><td colspan="2">Row Address</td><td colspan="3">Column Address</td><td>EMIF_nDQM[0]</td></tr>
<tr><td>1</td><td>2</td><td>Row Address</td><td>EMIF_BA[0]</td><td colspan="3">Column Address</td><td>EMIF_nDQM[0]</td></tr>
<tr><td>2</td><td>2</td><td colspan="2">EMIF_BA[1:0]</td><td colspan="3">Column Address</td><td>EMIF_nDQM[0]</td></tr>
<tr><td>0</td><td>3</td><td>Row Address</td><td colspan="4">Column Address</td><td>EMIF_nDQM[0]</td></tr>
<tr><td>1</td><td>3</td><td>EMIF_BA[0]</td><td colspan="4">Column Address</td><td>EMIF_nDQM[0]</td></tr>
<tr><td>2</td><td>3</td><td>EMIF_BA[0]</td><td colspan="4">Column Address</td><td>EMIF_nDQM[0]</td></tr>
</table>

NOTE: The upper bit of the Row Address is used only when addressing 256-Mbit and 512-Mbit SDRAM memories.

17.2.6 Asynchronous Controller and Interface

The EMIF easily interfaces to a variety of asynchronous devices including NOR Flash and SRAM. It can be operated in two major modes (see Table 17-14):

- Normal Mode
- Select Strobe Mode

Table 17-14. Normal Mode vs. Select Strobe Mode

Mode	Function of EMIF_nDQM pins	Operation of EMIF_nCS[4:2]
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 EMIF_nDQM pins of the EMIF function as byte enables. In this mode, the EMIF_nCS[4:2] pins behaves as typical chip select signals, remaining active for the duration of the asynchronous access. See Section 17.2.6.1 for an example interface with multiple 8-bit devices.

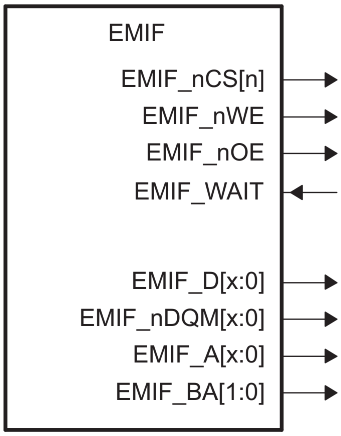
The second mode of operation is Select Strobe Mode, in which the EMIF_nCS[4:2] pins act as a strobe, active only during the strobe period of an access. In this mode, the EMIF_nDQM pins of the EMIF function as standard byte enables for reads and writes. A summary of the differences between the two modes of operation are shown in Table 17-14. Refer to Section 17.2.6.4 for the details of asynchronous operations in Normal Mode, and to Section 17.2.6.5 for the details of asynchronous operations in Select Strobe Mode. The EMIF 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, or 4). Throughout the chapter, *m* can hold the values 1, 2, 3 or 4; and *n* can hold the values 2, 3, or 4.

The EMIF 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.

17.2.6.1 Interfacing to Asynchronous Memory

Figure 17-7 shows the EMIF's external pins used in interfacing with an asynchronous device. In EMIF_nCS[n], n = 2, 3, or 4.

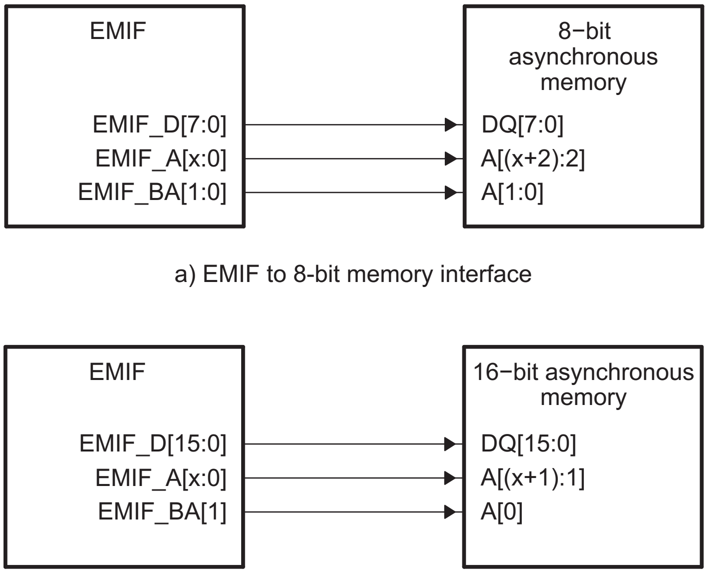
Figure 17-7. EMIF Asynchronous Interface



Of special note is the connection between the EMIF and the external device's address bus. The EMIF address pin EMIF_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 EMIF_BA[1] and EMIF_BA[0] pins provide the least-significant bits of the halfword or byte address, respectively. Additionally, when the EMIF interfaces to a 16-bit asynchronous device, the EMIF_BA[0] pin can serve as the upper address line EMIF_A[12]. Figure 17-8 and Figure 17-9 show the mapping between the EMIF 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 17-9 shows a common interface between the EMIF and external asynchronous memory. Figure 17-9 shows an interface between the EMIF and an external memory with byte enables. The EMIF should be operated in either Normal Mode or Select Strobe Mode when using this interface, so that the EMIF_nDQM signals operate as byte enables.

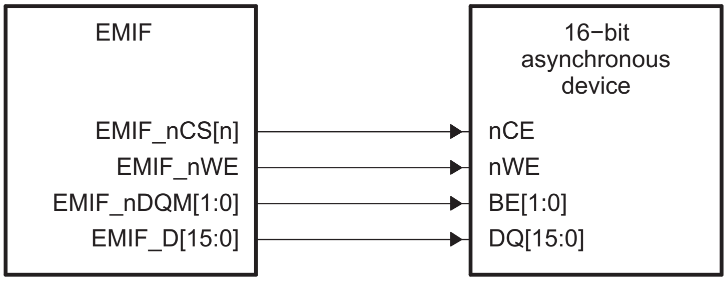
Figure 17-8. EMIF to 8-bit/16-bit Memory Interface



a) EMIF to 8-bit memory interface

b) EMIF to 16-bit memory interface

Figure 17-9. Common Asynchronous Interface



17.2.6.2 Accessing Larger Asynchronous Memories

The device has 13 dedicated EMIF address lines. If a device such as a large asynchronous flash needs to be attached to the EMIF, then GPIO pins may be used to control the flash device’s upper address lines.

17.2.6.3 Configuring the EMIF for Asynchronous Accesses

The operation of the EMIF'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 17.3. 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 17-15. Description of the Asynchronous *m* Configuration Register (CE*n*CFG)

Parameter	Description
SS	**Select Strobe mode.** This bit selects the EMIF's mode of operation in the following way: • SS = 0 selects Normal Mode – EMIF_nDQM pins function as byte enables – EMIF_nCS[4:2] active for duration of access • SS = 1 selects Select Strobe Mode – EMIF_nDQM pins function as byte enables – EMIF_nCS[4:2] 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 EMIF enables its Extended Wait Mode in which the strobe width of an access cycle can be extended in response to the assertion of the EMIF_nWAIT pin. The WP*n* bit in the asynchronous wait cycle configuration register (AWCC) controls to polarity of EMIF_nWAIT pin. See Section 17.2.6.6 for more details on this mode of operation.
W_SETUP/R_SETUP	**Read/Write setup widths.** These fields define the number of EMIF clock cycles of setup time for the address pins (EMIF_A and EMIF_BA), byte enables (EMIF_nDQM), and asynchronous chip enable (EMIF_nCS[4:2]) before the read strobe pin (EMIF_nOE) or write strobe pin (EMIF_nWE) falls, minus one cycle. For writes, the W_SETUP field also defines the setup time for the data pins (EMIF_D). Refer to the datasheet of the external asynchronous device to determine the appropriate setting for this field.
W_STROBE/R_STROBE	**Read/Write strobe widths.** These fields define the number of EMIF clock cycles between the falling and rising of the read strobe pin (EMIF_nOE) or write strobe pin (EMIF_nWE), minus one cycle. 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.** These fields define the number of EMIF clock cycles of hold time for the address pins (EMIF_A and EMIF_BA), byte enables (EMIF_nDQM), and asynchronous chip enable (EMIF_nCS[4:2]) after the read strobe pin (EMIF_nOE) or write strobe pin (EMIF_nWE) rises, minus one cycle. For writes, the W_HOLD field also defines the hold time for the data pins (EMIF_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 EMIF 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.

Table 17-15. Description of the Asynchronous *m* Configuration Register (CE*n*CFG) (continued)

Parameter	Description
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 EMIF_A and EMIF_BA pins as described in Section 17.2.6.1. This field also determines the number of external accesses required to fulfill a request generated by one of the sources mentioned in Section 17.2.2. 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.

Table 17-16. Description of the Asynchronous Wait Cycle Configuration Register (AWCC)

Parameter	Description
WP*n*	**EM_WAIT Polarity.** • WP*n* = 0 selects active-low polarity • WP*n* = 1 selects active-high polarity When set to 1, the EMIF will wait if the EMIF_nWAIT pin is high. When cleared to 0, the EMIF will wait if the EMIF_nWAIT pin is low. The EMIF must have the Extended Wait Mode enabled for the EMIF_nWAIT pin to affect the width of the strobe period.
MAX_EXT_WAIT	**Maximum Extended Wait Cycles.** This field configures the number of EMIF clock cycles the EMIF will wait for the EMIF_nWAIT pin to be deactivated during the strobe period of an access cycle. The maximum number of EMIF clock cycles it will wait is determined by the following formula: Maximum Extended Wait Cycles = (MAX_EXT_WAIT + 1) × 16 If the EMIF_nWAIT pin is not deactivated within the time specified by this field, the EMIF 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 EMIF interrupt mask set register (INTMSKSET). Refer to Section 17.2.9.1 for more information about the EMIF interrupts.

Table 17-17. Description of the EMIF 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 EMIF_nWAIT occurs
AT_MASK_SET	**Asynchronous Timeout Mask Set.** Writing a 1 to this bit enables an interrupt to be generated when an Asynchronous Timeout occurs.

Table 17-18. Description of the EMIF 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 EMIF 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.

17.2.6.4 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 EMIF_nDQM pins operate as byte enables. Section 17.2.6.4.1 and Section 17.2.6.4.2 explain the details of read and write operations while in Normal Mode.

17.2.6.4.1 Asynchronous Read Operations (Normal Mode)

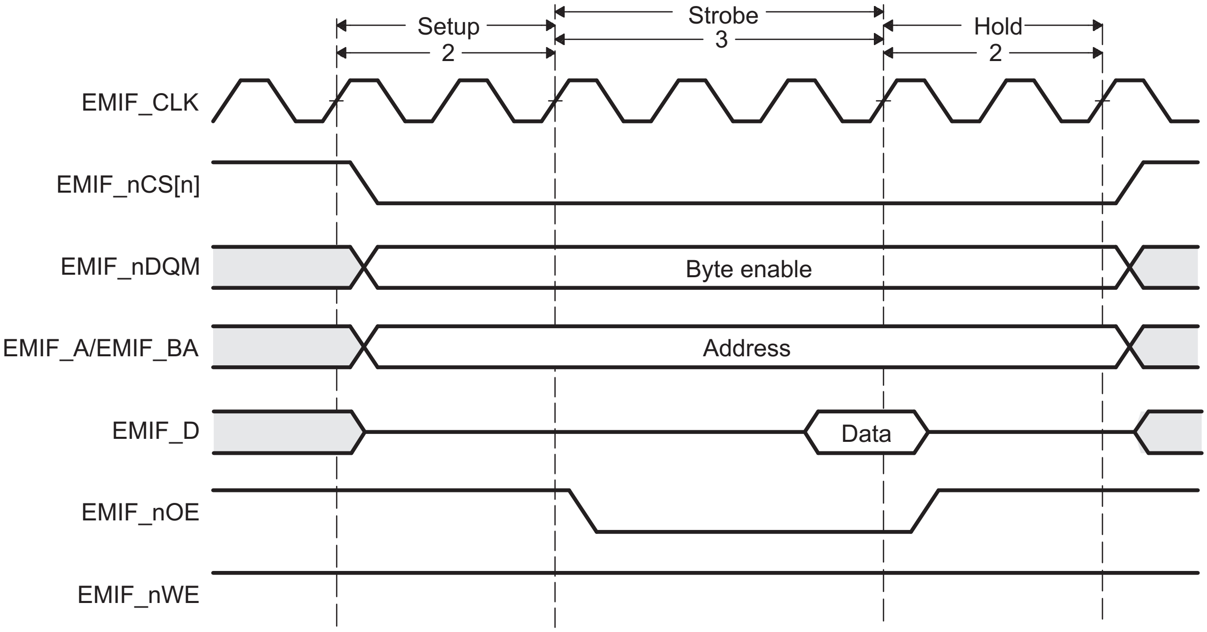
NOTE: During an entire asynchronous read operation, the EMIF_nWE pin is driven high.

An asynchronous read is performed when any of the requesters mentioned in Section 17.2.2 request a read from the attached asynchronous memory. After the request is received, a read operation is initiated once it becomes the EMIF's highest priority task, according to the priority scheme detailed in Section 17.2.13. 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 EMIF until the entire request is fulfilled. The details of an asynchronous read operation in Normal Mode are described in Table 17-19. Also, Figure 17-10 shows an example timing diagram of a basic read operation.

Table 17-19. Asynchronous Read Operation in Normal Mode

Time Interval	Pin Activity in Normal Mode
Turn-around period	Once the read operation becomes the highest priority task for the EMIF, the EMIF 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, no turnaround cycles are inserted. • If the current read operation was directly proceeded by a write operation and the TA field has been cleared to 0, one turn-around cycle will be inserted. After the EMIF 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 EMIF proceeds to the setup period of the operation. If it is no longer the highest priority task, the EMIF 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 EMIF_A and EMIF_BA become valid and carry the values described in Section 17.2.6.1. • EMIF_nCS[4:2] 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. EMIF_nOE 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: • EMIF_nOE rises • The data on the EMIF_D bus is sampled by the EMIF. In Figure 17-10, EMIF_nWAIT is inactive. If EMIF_nWAIT is instead activated, the strobe period can be extended by the external device to give it more time to provide the data. Section 17.2.6.6 contains more details on using the EMIF_nWAIT pin.
End of the hold period	At the end of the hold period: • The address pins EMIF_A and EMIF_BA become invalid • EMIF_nCS[4:2] rises (if no more operations are required to complete the current request) EMIF 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 EMIF 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 EMIF 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 EMIF instead enters directly into the turnaround period for the pending read or write operation.

Figure 17-10. Timing Waveform of an Asynchronous Read Cycle in Normal Mode



17.2.6.4.2 Asynchronous Write Operations (Normal Mode)

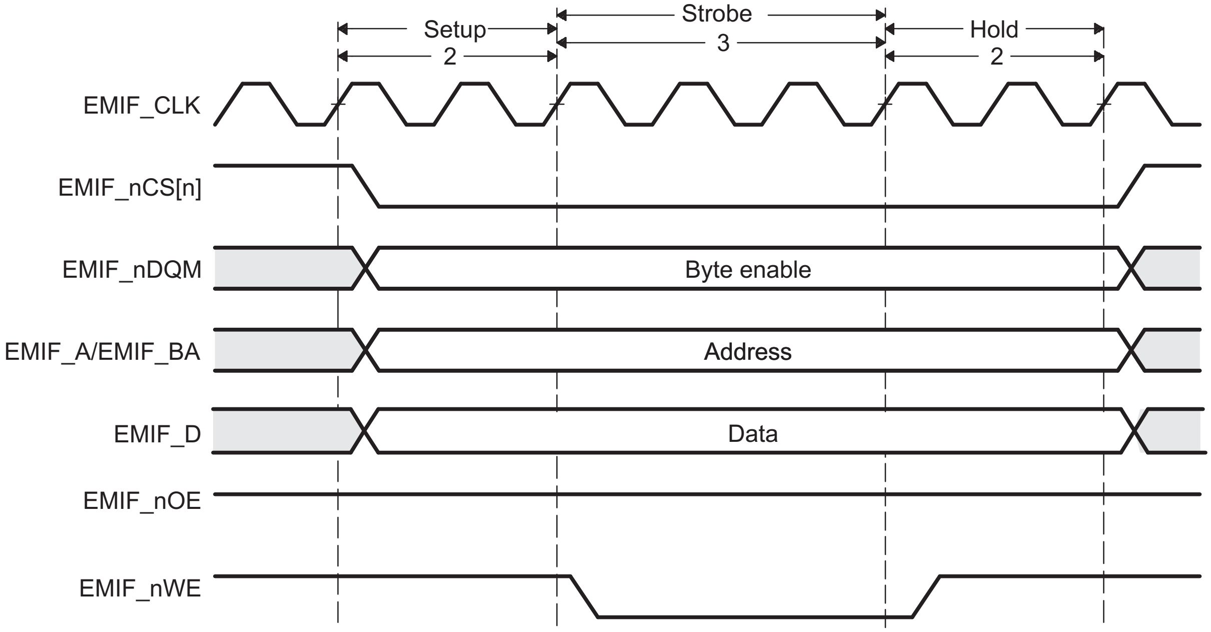
NOTE: During an entire asynchronous write operation, the EMIF_nOE pin is driven high.

An asynchronous write is performed when any of the requesters mentioned in Section 17.2.2 request a write to memory in the asynchronous bank of the EMIF. After the request is received, a write operation is initiated once it becomes the EMIF's highest priority task, according to the priority scheme detailed in Section 17.2.13. 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 EMIF until the entire request is fulfilled. The details of an asynchronous write operation in Normal Mode are described in Table 17-20. Also, Figure 17-11 shows an example timing diagram of a basic write operation.

Table 17-20. 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 EMIF, the EMIF 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, no turn-around cycles are inserted. • If the current write operation was directly proceeded by a read operation and the TA field has been cleared to 0, one turnaround cycle will be inserted. After the EMIF 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 EMIF proceeds to the setup period of the operation. If it is no longer the highest priority task, the EMIF 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 EMIF_A and EMIF_BA and the data pins EMIF_D become valid. The EMIF_A and EMIF_BA pins carry the values described in Section 17.2.6.1. • EMIF_nCS[4:2] 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. EMIF_nWE falls 2. The EMIF_nDQM 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. EMIF_nWE rises 2. The EMIF_nDQM pins deactivate In Figure 17-11, EMIF_nWAIT is inactive. If EMIF_nWAIT is instead activated, the strobe period can be extended by the external device to give it more time to accept the data. Section 17.2.6.6 contains more details on using the EMIF_nWAIT pin.
End of the hold period	At the end of the hold period: • The address pins EMIF_A and EMIF_BA become invalid • The data pins become invalid • EMIF_nCS[n] (n = 2, 3, or 4) rises (if no more operations are required to complete the current request) The EMIF 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 EMIF 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 EMIF 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 EMIF instead enters directly into the turnaround period for the pending read or write operation.

Figure 17-11. Timing Waveform of an Asynchronous Write Cycle in Normal Mode



17.2.6.5 Read and Write Operation in Select Strobe Mode

Select Strobe Mode is the EMIF'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 EMIF_nDQM pins operate as byte enables and the EMIF_nCS[n] (n = 2, 3, or 4) pin is only active during the strobe period of an access cycle. Section 17.2.6.4.1 and Section 17.2.6.4.2 explain the details of read and write operations while in Select Strobe Mode.

17.2.6.5.1 Asynchronous Read Operations (Select Strobe Mode)

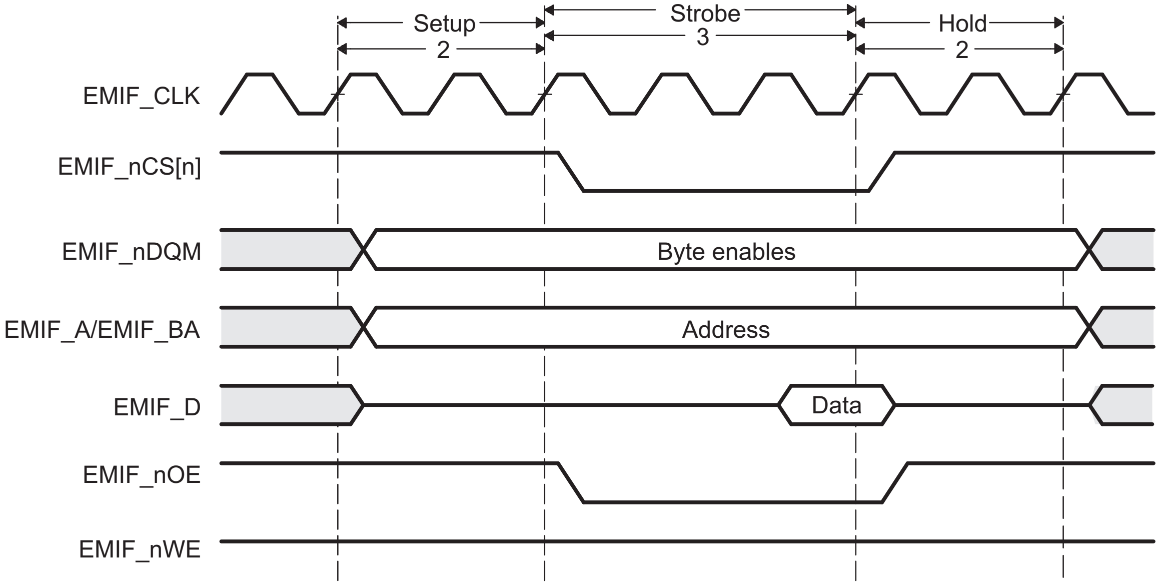
NOTE: During the entirety of an asynchronous read operation, the EMIF_nWE pin is driven high.

An asynchronous read is performed when any of the requesters mentioned in Section 17.2.2 request a read from the attached asynchronous memory. After the request is received, a read operation is initiated once it becomes the EMIF's highest priority task, according to the priority scheme detailed in Section 17.2.13. 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 EMIF until the entire request is fulfilled. The details of an asynchronous read operation in Select Strobe Mode are described in Table 17-21. Also, Figure 17-12 shows an example timing diagram of a basic read operation.

Table 17-21. 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 EMIF, the EMIF 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, no turn-around cycles are inserted. • If the current read operation was directly proceeded by a write operation and the TA field has been cleared to 0, one turn-around cycle will be inserted. After the EMIF 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 EMIF proceeds to the setup period of the operation. If it is no longer the highest priority task, the EMIF 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 EMIF_A and EMIF_BA become valid and carry the values described in Section 17.2.6.1. • The EMIF_nDQM pins become valid as byte enables.
Strobe period	The following actions occur during the strobe period of a read operation: 1. EMIF_nCS[n] (n = 2, 3, or 4) and EMIF_nOE 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: • EMIF_nCS[n] (n = 2, 3, or 4) and EMIF_nOE rise • The data on the EMIF_D bus is sampled by the EMIF. In Figure 17-12, EMIF_nWAIT is inactive. If EMIF_nWAIT is instead activated, the strobe period can be extended by the external device to give it more time to provide the data. Section 17.2.6.6 contains more details on using the EMIF_nWAIT pin.
End of the hold period	At the end of the hold period: • The address pins EMIF_A and EMIF_BA become invalid • The EMIF_nDQM pins become invalid The EMIF 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 EMIF 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 EMIF 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 EMIF instead enters directly into the turnaround period for the pending read or write operation.

Figure 17-12. Timing Waveform of an Asynchronous Read Cycle in Select Strobe Mode



17.2.6.5.2 Asynchronous Write Operations (Select Strobe Mode)

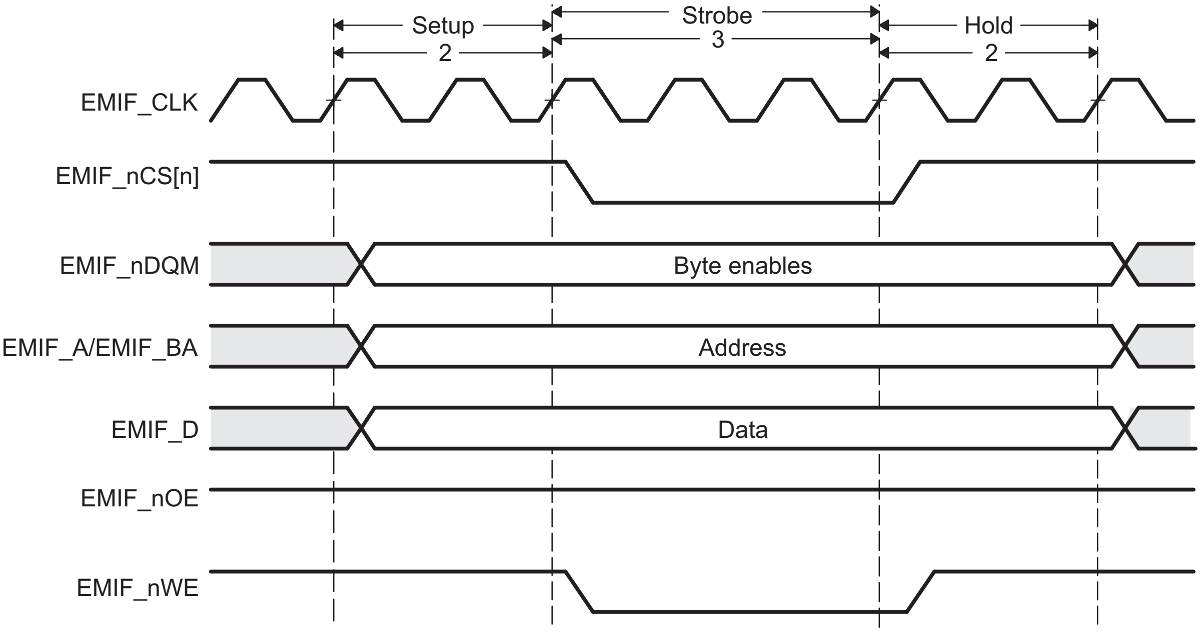
NOTE: During the entirety of an asynchronous write operation, the EMIF_nOE pin is driven high.

An asynchronous write is performed when any of the requesters mentioned in Section 17.2.2 request a write to memory in the asynchronous bank of the EMIF. After the request is received, a write operation is initiated once it becomes the EMIF's highest priority task, according to the priority scheme detailed in Section 17.2.13. 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 EMIF until the entire request is fulfilled. The details of an asynchronous write operation in Select Strobe Mode are described in Table 17-22. Also, Figure 17-13 shows an example timing diagram of a basic write operation.

Table 17-22. 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 EMIF, the EMIF waits for the programmed number of turnaround 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, no turn-around cycles are inserted. • If the current write operation was directly proceeded by a read operation and the TA field has been cleared to 0, one turnaround cycle will be inserted. After the EMIF 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 EMIF proceeds to the setup period of the operation. If it is no longer the highest priority task, the EMIF 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 EMIF_A and EMIF_BA and the data pins EMIF_D become valid. The EMIF_A and EMIF_BA pins carry the values described in Section 17.2.6.1. • The EMIF_nDQM pins become active as byte enables.
Strobe period	The following actions occur at the start of the strobe period of a write operation: • EMIF_nCS[n] (n = 2, 3, or 4) and EMIF_nWE fall The following actions occur on the rising edge of the clock which is concurrent with the end of the strobe period: • EMIF_nCS[n] (n = 2, 3, or 4) and EMIF_nWE rise In Figure 17-13, EMIF_nWAIT is inactive. If EMIF_nWAIT is instead activated, the strobe period can be extended by the external device to give it more time to accept the data. Section 17.2.6.6 contains more details on using the EMIF_nWAIT pin.
End of the hold period	At the end of the hold period: • The address pins EMIF_A and EMIF_BA become invalid • The data pins become invalid • The EMIF_nDQM pins become invalid The EMIF 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 EMIF 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 EMIF 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 EMIF instead enters directly into the turn-around period for the pending read or write operation.

Figure 17-13. Timing Waveform of an Asynchronous Write Cycle in Select Strobe Mode



17.2.6.6 Extended Wait Mode and the EMIF_nWAIT Pin

The EMIF 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, or 4). When this bit is set, the EMIF monitors the EMIF_nWAIT 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 EMIF detects that the EMIF_nWAIT pin has been asserted, it will begin inserting extra strobe cycles into the operation until the EMIF_nWAIT pin is deactivated by the external device. The EMIF 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 EMIF_nWAIT signal.

The EMIF_nWAIT 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 EMIF_CLK cycles the strobe period may be extended beyond the programmed length. When the counter expires, the EMIF proceeds to the hold period of the operation regardless of the state of the EMIF_nWAIT pin. The EMIF can also generate an interrupt upon expiration of this counter. See Section 17.2.9.1 for details on enabling this interrupt.

For the EMIF to function properly in the Extended Wait mode, the WP*n* bit of AWCC must be programmed to match the polarity of the EMIF_nWAIT pin. In its reset state of 1, the EMIF will insert wait cycles when the EMIF_nWAIT pin is sampled high. When set to 0, the EMIF will insert wait cycles only when EMIF_nWAIT is sampled low. This programmability allows for a glueless connection to larger variety of asynchronous devices.

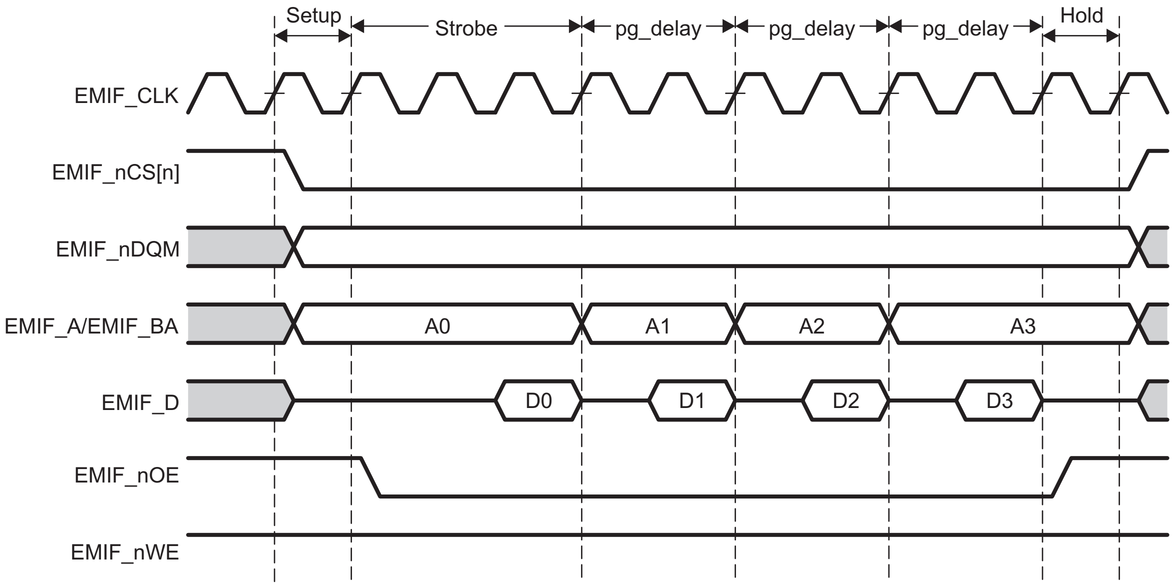
Finally, a restriction is placed on the strobe period timing parameters when operating in Extended Wait mode. Specifically, the sum of the W_SETUP and W_STROBE fields must be greater than 4, and the sum of the R_SETUP and R_STROBE fields must be greater than 4 for the EMIF to recognize the EMIF_nWAIT pin has been asserted. The W_SETUP, W_STROBE, R_SETUP, and R_STROBE fields are in CE*n*CFG.

17.2.6.7 NOR Flash Page Mode

EMIF supports Page mode reads for NOR Flash on its asynchronous memory chip selects. This mode can be enabled by writing a 1 to the CS*n*_PG_MD_EN (*n* = 2, 3, or 4) field in the Page Mode Control register for the chip select in consideration. Whenever Page Mode for reads is enabled for a particular chip select, the page size for the device connected must also be programmed in the CS*n*_PG_SIZE field of the Page Mode Control register. The address change to valid read data available timing must be programmed in the CS*n*_PG_DEL field of the Page Control register. All other asynchronous memory timings must be programmed in the asynchronous configuration register (CE*n*CFG). See Figure 17-14 for read in asynchronous page mode.

NOTE: The Extended Wait mode and the Select Strobe mode must be disabled when using the asynchronous interface in Page mode.

Figure 17-14. Asynchronous Read in Page Mode



17.2.7 Data Bus Parking

The EMIF 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 EMIF issues a read command to the external memory does it stop driving the data bus. After the EMIF latches the last read data, it immediately parks the data bus again.

The one exception to this behavior occurs after performing an asynchronous read operation while the EMIF is in the self-refresh state. In this situation, the read operation is not followed by the EMIF parking the data bus. Instead, the EMIF tri-states the data bus. Therefore, it is not recommended to perform asynchronous read operations while the EMIF 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 EMIF 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 17.2.5.7.

17.2.8 Reset and Initialization Considerations

The EMIF memory controller has two active-low reset signals, CHIP_RST_n and MOD_G_RST_n. Both these reset signals are driven by the device system reset signal. This device does not offer the flexibility to reset just the EMIF state machine without also resetting the EMIF controller's memory-mapped registers. As soon as the device system reset is released (driven High), the EMIF memory controller immediately begins its initialization sequence. Command and data stored in the EMIF memory controller FIFOs are lost. Refer the Architecture chapter of the tecnical reference manual (TRM) for more information on conditions that can cause a device system reset to be asserted.

When system reset is released, the EMIF automatically begins running the SDRAM initialization sequence described in Section 17.2.5.4. Even though the initialization procedure is automatic, a special procedure, found in Section 17.2.5.5 must still be followed.

17.2.9 Interrupt Support

The EMIF supports a single interrupt to the CPU. Section 17.2.9.1 details the generation and internal masking of EMIF interrupts.

17.2.9.1 Interrupt Events

There are three conditions that may cause the EMIF to generate an interrupt to the CPU. These conditions are:

- A rising edge on the EMIF_nWAIT 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 EMIF_nWAIT 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 EMIF_nWAIT pin within the number of cycles defined by the MAX_EXT_WAIT bit in AWCC (this happens only in extended wait mode). EMIF supports only linear incrementing and cache line wrap addressing modes . If an access request for an unsupported addressing mode is received, the EMIF will set the LT bit in the EMIF 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 EMIF 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 EMIF 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 EMIF interrupt raw register (INTRAW) and the EMIF 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 17-23 contains a brief summary of the interrupt status and control bit fields. See Section 17.3 for complete details on the register fields.

Table 17-23. Interrupt Monitor and Control Bit Fields

Register Name	Bit Name	Description
EMIF interrupt raw register (INTRAW)	WR	This bit is set when an rising edge on the EMIF_nWAIT 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.
EMIF interrupt mask register (INTMSK)	WR_MASKED	This bit is set only when a rising edge on the EMIF_nWAIT 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.
EMIF 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.
EMIF 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.

17.2.10 DMA Event Support

EMIF 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.

17.2.11 EMIF Signal Multiplexing

For details on EMIF signal multiplexing, see the I/O Multiplexing Module chapter of the technical reference manual.

17.2.12 Memory Map

For information describing the device memory-map, see your device-specific datasheet.

17.2.13 Priority and Arbitration

Section 17.2.2 describes the external prioritization and arbitration among requests from different sources within the microcontroller. The result of this external arbitration is that only one request is presented to the EMIF at a time. Once the EMIF completes a request, the external arbiter then provides the EMIF with the next pending request.

Internally, the EMIF 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 EMIF to immediately commence its initialization sequence as described in Section 17.2.5.4.

Once the EMIF has completed its initialization sequence, it performs memory transactions according to the following priority scheme (highest priority listed first):

1. If the EMIF's backlog refresh counter is at the Refresh Must urgency level, the EMIF 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 EMIF performs a read operation.
3. If the EMIF's backlog refresh counter is at the Refresh Need urgency level, the EMIF performs an SDRAM auto refresh cycle.
4. If an SDRAM or asynchronous write has been requested, the EMIF performs a write operation.
5. If the EMIF's backlog refresh counter is at the Refresh May or Refresh Release urgency level, the EMIF performs an SDRAM auto refresh cycle.
6. If the value of the SR bit in SDCR has been set to 1, the EMIF will enter the self-refresh state as described in Section 17.2.5.7.

After taking one of the actions listed above, the EMIF then returns to the top of the priority list to determine its next action.

Because the EMIF 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 17.2.5.7 for details on the operation of the EMIF when in the self-refresh state.

17.2.14 System Considerations

This section describes various system considerations to keep in mind when operating the EMIF.

17.2.14.1 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 EMIF can be sent is 16 words, therefore the maximum number of access cycles per memory request is 64 when the EMIF 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 EMIF_nWAIT 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.

17.2.14.2 Interface to External Peripheral or FIFO Memory

If EMIF is used to interface to an external peripheral or FIFO logic (for example, UHPI), it is recommended to use the host CPU's Memory Protection Unit (MPU) to define this external memory range as a region that is either strongly-ordered or of device type.

17.2.14.3 Interface to External SDRAM

If EMIF is used to interface to an external SDRAM, it is recommended to burst as much as possible to normal memory to improve the interface bandwidth.

17.2.15 Power Management

Power dissipation from the EMIF 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 EMIF memory controller achieves higher power savings when compared to the power savings of self-refresh or power down mode. The input clock VCLK3 can be turned off through the use of the Global Clock Module (GCM). Before gating clocks off, the EMIF memory controller must place the SDR SDRAM memory in self-refresh mode. If the external memory requires a continuous clock, the VCLK3 clock domain 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 EMIF memory controller clocks.

17.2.15.1 Power Management Using Self-Refresh Mode

The EMIF 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 17.2.5.7 for more details on placing the EMIF into the self-refresh state.

17.2.15.2 Power Management Using Power Down Mode

In the power down mode, EMIF drives EMIF_CKE low to lower the power consumption. EMIF_CKE goes high when there is a need to send refresh (REFR) commands, after which EMIF_CKE is again driven low. EMIF_CKE remains low until any request arrives. Refer to Section 17.2.5.8 for more details on placing EMIF in power down mode.

17.2.16 Emulation Considerations

EMIF memory controller remains fully functional during emulation halts in order to allow emulation access to external memory.

17.3 EMIF Registers

The external memory interface (EMIF) is controlled by programming its internal memory-mapped registers (MMRs). Table 17-24 lists the memory-mapped registers for the EMIF.

NOTE: All EMIF 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 EMIF registers must always be accessed using 32-bit accesses (unless otherwise specified in this chapter). The base address of the EMIF memory-mapped registers is FCFF E800h.

Table 17-24. External Memory Interface (EMIF) Registers

Offset	Acronym	Register Description	Section
00h	MIDR	Module ID Register	Section 17.3.1
04h	AWCC	Asynchronous Wait Cycle Configuration Register	Section 17.3.2
08h	SDCR	SDRAM Configuration Register	Section 17.3.3
0Ch	SDRCR	SDRAM Refresh Control Register	Section 17.3.4
10h	CE2CFG	Asynchronous 1 Configuration Register	Section 17.3.5
14h	CE3CFG	Asynchronous 2 Configuration Register	Section 17.3.5
18h	CE4CFG	Asynchronous 3 Configuration Register	Section 17.3.5
1Ch	CE5CFG	Asynchronous 4 Configuration Register	Section 17.3.5
20h	SDTIMR	SDRAM Timing Register	Section 17.3.6
3Ch	SDSRETR	SDRAM Self Refresh Exit Timing Register	Section 17.3.7
40h	INTRAW	EMIF Interrupt Raw Register	Section 17.3.8
44h	INTMSK	EMIF Interrupt Mask Register	Section 17.3.9
48h	INTMSKSET	EMIF Interrupt Mask Set Register	Section 17.3.10
4Ch	INTMSKCLR	EMIF Interrupt Mask Clear Register	Section 17.3.11
68h	PMCR	Page Mode Control Register	Section 17.3.12

17.3.1 Module ID Register (MIDR)

This is a read-only register indicating the module ID of the EMIF. The MIDR is shown in Figure 17-15 and described in Table 17-25.

Figure 17-15. Module ID Register (MIDR) [offset = 00]

31	0
REV	
R-x	

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

Table 17-25. Module ID Register (MIDR) Field Descriptions

Bit	Field	Value	Description
31-0	REV	x	Module ID of EMIF. See the device-specific data manual.

17.3.2 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 EMIF_nWAIT pin(s) and the maximum allowable number of extended wait cycles can be configured. The AWCC is shown in Figure 17-16 and described in Table 17-26. Not all devices support both EMIF_nWAIT[1] and EMIF_nWAIT[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 17-16. Asynchronous Wait Cycle Configuration Register (AWCCR) [offset = 04h]

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-3h	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

Table 17-26. Asynchronous Wait Cycle Configuration Register (AWCCR) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	3h	Reserved
29	WP1		EMIF_nWAIT[1] polarity bit. This bit defines the polarity of the EMIF_nWAIT[1] pin.
		0	Insert wait cycles if EMIF_nWAIT[1] pin is low.
		1	Insert wait cycles if EMIF_nWAIT[1] pin is high.
28	WP0		EMIF_nWAIT[0] polarity bit. This bit defines the polarity of the EMIF_nWAIT[0] pin.
		0	Insert wait cycles if EMIF_nWAIT[0] pin is low.
		1	Insert wait cycles if EMIF_nWAIT[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 EMIF_nWAIT[*n*] signal used for memory accesses to chip select 5 memory space. This device does not support chip select 5, so any value written to this field has no effect.
21-20	CS4_WAIT		Chip Select 4 WAIT signal selection. This signal determines which EMIF_nWAIT[*n*] signal used for memory accesses to chip select 4 memory space.
		0	EMIF_nWAIT[0] pin is used to control external wait states.
		1h	EMIF_nWAIT[1] pin is used to control external wait states.
		2h-3h	Reserved
19-18	CS3_WAIT		Chip Select 3 WAIT signal selection. This signal determines which EMIF_nWAIT[*n*] signal used for memory accesses to chip select 3 memory space.
		0	EMIF_nWAIT[0] pin is used to control external wait states.
		1h	EMIF_nWAIT[1] pin is used to control external wait states.
		2h-3h	Reserved
17-16	CS2_WAIT		Chip Select 2 WAIT signal selection. This signal determines which EMIF_nWAIT[*n*] signal used for memory accesses to chip select 2 memory space.
		0	EMIF_nWAIT[0] pin is used to control external wait states.
		1h	EMIF_nWAIT[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 EMIF waits 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.

17.3.3 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 17-17 and described in Table 17-27.

NOTE: Writing to the lower three bytes of this register causes the EMIF to start the SDRAM initialization sequence described in Section 17.2.5.4.

Figure 17-17. SDRAM Configuration Register (SDCR) [offset = 08h]

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 EMIF on your device only has 16 data bus pins.

Table 17-27. 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 17.2.5.7. 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 causes connected SDRAM devices and the EMIF to exit the Self-Refresh mode.
		1	Writing a 1 to this bit causes connected SDRAM devices and the EMIF 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 EMIF goes into Self Refresh.
		0	Writing a 0 to this bit causes connected SDRAM devices and the EMIF to exit the power-down mode.
		1	Writing a 1 to this bit causes connected SDRAM devices and the EMIF to enter the power-down mode.
29	PDWR		Perform refreshes during power down. Writing a 1 to this bit causes EMIF 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 EMIF. 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.

Table 17-27. SDRAM Configuration Register (SDCR) Field Descriptions (continued)

Bit	Field	Value	Description
11-9	CL		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 EMIF_CLK cycles
		3h	CAS latency = 3 EMIF_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		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		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

17.3.4 SDRAM Refresh Control Register (SDRCR)

The SDRAM refresh control register (SDRCR) is used to configure the rate at which connected SDRAM devices are automatically refreshed by the EMIF. Refer to Section 17.2.5.6 on the refresh controller for more details. The SDRCR is shown in Figure 17-18 and described in Table 17-28.

Figure 17-18. SDRAM Refresh Control Register (SDRCR) [offset = 0Ch]

31-16
Reserved
R-0

15-13	12-0
Reserved	RR
R-0	R/W-60h

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

Table 17-28. 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 EMIF_CLK cycles. Writing a value < 0x0020 to this field causes it to be loaded with (2 × T_RFC) + 1 value from the SDRAM timing register (SDTIMR).

17.3.5 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. CS5 is not available on this device. 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 uses the new configuration. The CE*n*CFG is shown in Figure 17-19 and described in Table 17-29.

Figure 17-19. Asynchronous *n* Configuration Register (CE*n*CFG) [offset = 10h - 1Ch]

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.

B. This bit field must be cleared to 0, if the EMIF on your device does not have an EMIF_nWAIT pin.

Table 17-29. 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 17.2.6 for details on the two modes of operation.
		0	Normal Mode is enabled.
		1	Select Strobe Mode is enabled.
30	EW		Extend Wait bit. This bit defines whether extended wait cycles are enabled. See Section 17.2.6.6 on extended wait cycles for details. This bit field must be cleared to 0, if the EMIF on your device does not have an EMIF_nWAIT pin.
		0	Extended wait cycles are disabled.
		1	Extended wait cycles are enabled.
29-26	W_SETUP	0-Fh	Write setup width in EMIF_CLK cycles, minus one cycle. See Section 17.2.6.3 for details.
25-20	W_STROBE	0-3Fh	Write strobe width in EMIF_CLK cycles, minus one cycle. See Section 17.2.6.3 for details.
19-17	W_HOLD	0-7h	Write hold width in EMIF_CLK cycles, minus one cycle. See Section 17.2.6.3 for details.
16-13	R_SETUP	0-Fh	Read setup width in EMIF_CLK cycles, minus one cycle. See Section 17.2.6.3 for details.
12-7	R_STROBE	0-3Fh	Read strobe width in EMIF_CLK cycles, minus one cycle. See Section 17.2.6.3 for details.
6-4	R_HOLD	0-7h	Read hold width in EMIF_CLK cycles, minus one cycle. See Section 17.2.6.3 for details.
3-2	TA	0-3h	Minimum Turn-Around time. This field defines the minimum number of EMIF_CLK cycles between reads and writes, minus 1 cycle. See Section 17.2.6.3 for details.
1-0	ASIZE		Asynchronous Data Bus Width. This field defines the width of the asynchronous device data bus.
		0	8-bit data bus
		1h	16-bit data bus
		2h-3h	Reserved

17.3.6 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 17-20 and described in Table 17-30.

Figure 17-20. SDRAM Timing Register (SDTIMR) [offset = 20h]

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 17-30. 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 EMIF_CLK cycles from Refresh (REFR) to Refresh (REFR), minus 1: T_RFC = (Trfc/t_{EMIF_CLK}) - 1
26-24	T_RP	0-7h	Specifies the Trp value of the SDRAM. This defines the minimum number of EMIF_CLK cycles from Precharge (PRE) to Activate (ACTV) or Refresh (REFR) command, minus 1: T_RP = (Trp/t_{EMIF_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 EMIF_CLK cycles from Active (ACTV) to Read (READ) or Write (WRT), minus 1: T_RCD = (Trcd/t_{EMIF_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 EMIF_CLK cycles from last Write (WRT) to Precharge (PRE), minus 1: T_WR = (Twr/t_{EMIF_CLK}) - 1
15-12	T_RAS	0-Fh	Specifies the Tras value of the SDRAM. This defines the minimum number of EMIF_CLK clock cycles from Activate (ACTV) to Precharge (PRE), minus 1: T_RAS = (Tras/t_{EMIF_CLK}) - 1
11-8	T_RC	0-Fh	Specifies the Trc value of the SDRAM. This defines the minimum number of EMIF_CLK clock cycles from Activate (ACTV) to Activate (ACTV), minus 1: T_RC = (Trc/t_{EMIF_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 EMIF_CLK clock cycles from Activate (ACTV) to Activate (ACTV) for a different bank, minus 1: T_RRD = (Trrd/t_{EMIF_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.

17.3.7 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 EMIF issues another command. The SDSRETR is shown in Figure 17-21 and described in Table 17-31.

Figure 17-21. SDRAM Self Refresh Exit Timing Register (SDSRETR) [offset = 3Ch]

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 17-31. 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 1. T_XS = Txsr / t_{EMIF_CLK} - 1

17.3.8 EMIF Interrupt Raw Register (INTRAW)

The EMIF interrupt raw register (INTRAW) is used to monitor and clear the EMIF's hardware-generated Asynchronous Timeout Interrupt. The AT bit in this register is set when an Asynchronous Timeout occurs regardless of the status of the EMIF interrupt mask set register (INTMSKSET) and EMIF interrupt mask clear register (INTMSKCLR). Writing a 1 to this bit clears it. The EMIF on some devices does not have the EMIF_nWAIT pin; therefore, these registers and fields are reserved on those devices. The INTRAW is shown in Figure 17-22 and described in Table 17-32.

Figure 17-22. EMIF Interrupt Raw Register (INTRAW) [offset = 40h]

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 17-32. EMIF 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 EMIF_nWAIT pin has occurred.
		0	Indicates that a rising edge has not occurred on the EMIF_nWAIT pin. Writing a 0 has no effect.
		1	Indicates that a rising edge has occurred on the EMIF_nWAIT pin. Writing a 1 clears this bit and the WR_MASKED bit in the EMIF 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 clears this bit as well as the LT_MASKED bit in the EMIF 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 EMIF_nWAIT 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 clears this bit as well as the AT_MASKED bit in the EMIF interrupt masked register (INTMSK).

17.3.9 EMIF Interrupt Masked Register (INTMSK)

Like the EMIF interrupt raw register (INTRAW), the EMIF interrupt masked register (INTMSK) is used to monitor and clear the status of the EMIF'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 is 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 EMIF interrupt mask set register (INTMSKSET). The EMIF on some devices does not have the EMIF_nWAIT pin, therefore, these registers and fields are reserved on those devices. The INTMSK is shown in Figure 17-23 and described in Table 17-33.

Figure 17-23. EMIF Interrupt Mask Register (INTMSK) [offset = 44h]

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 17-33. EMIF 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 EMIF_nWAIT pin, provided that the WR_MASK_SET bit is set to 1 in the EMIF 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 clears this bit and the WR bit in the EMIF 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 EMIF interrupt mask set register (INTMSKSET) is set to 1.
		0	Writing a 0 has no effect.
		1	Writing a 1 clears this bit as well as the LT bit in the EMIF 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 EMIF_nWAIT 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 EMIF 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 clears this bit as well as the AT bit in the EMIF interrupt raw register (INTRAW).

17.3.10 EMIF Interrupt Mask Set Register (INTMSKSET)

The EMIF interrupt mask set register (INTMSKSET) is used to enable the Asynchronous Timeout Interrupt. If read as 1, the AT_MASKED bit in the EMIF interrupt masked register (INTMSK) is set and an interrupt is generated when an Asynchronous Timeout occurs. If read as 0, the AT_MASKED bit is always read 0 and no interrupt is generated when an Asynchronous Timeout occurs. Writing a 1 to the AT_MASK_SET bit enables the Asynchronous Timeout Interrupt. The EMIF on some devices does not have the EMIF_nWAIT pin; therefore, these registers and fields are reserved on those devices. The INTMSKSET is shown in Figure 17-24 and described in Table 17-34.

Figure 17-24. EMIF Interrupt Mask Set Register (INTMSKSET) [offset = 48h]

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 17-34. EMIF 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 EMIF 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 EMIF interrupt mask clear register (INTMSKCLR).
1	LT_MASK_SET		Mask set for LT_MASKED bit in the EMIF 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 EMIF 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 EMIF 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 EMIF 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 EMIF interrupt mask clear register (INTMSKCLR).

17.3.11 EMIF Interrupt Mask Clear Register (INTMSKCLR)

The EMIF interrupt mask clear register (INTMSKCLR) is used to disable the Asynchronous Timeout Interrupt. If read as 1, the AT_MASKED bit in the EMIF interrupt masked register (INTMSK) is set and an interrupt is generated when an Asynchronous Timeout occurs. If read as 0, the AT_MASKED bit is always read 0 and no interrupt is generated when an Asynchronous Timeout occurs. Writing a 1 to the AT_MASK_CLR bit disables the Asynchronous Timeout Interrupt. The EMIF on some devices does not have the EMIF_nWAIT pin, therefore, these registers and fields are reserved on those devices. The INTMSKCLR is shown in Figure 17-25 and described in Table 17-35.

Figure 17-25. EMIF Interrupt Mask Clear Register (INTMSKCLR) [offset = 4Ch]

<table>
<tr><td>31</td><td>16</td></tr>
<tr><td colspan="2">Reserved</td></tr>
<tr><td colspan="2">R-0</td></tr>
</table>

<table>
<tr><td>15</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">Reserved</td><td>WR_MASK_CLR</td><td>Reserved</td><td>AT_MASK_CLR</td></tr>
<tr><td colspan="2">R-0</td><td>R/W-0</td><td>R-0</td><td>R/W-0</td></tr>
</table>

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

Table 17-35. EMIF 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 EMIF 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 EMIF 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 EMIF 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 EMIF 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 EMIF 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 EMIF 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 EMIF interrupt mask set register (INTMSKSET).

17.3.12 Page Mode Control Register (PMCR)

The page mode control register (PMCR) is shown in Figure 17-26 and described in Table 17-36. This register is configured when using NOR Flash page mode.

Figure 17-26. Page Mode Control Register (PMCR) [offset = 68h]

31 – 26	25	24
CS5_PG_DEL	CS5_PG_SIZE	CS5_PG_MD_EN
R/W-3Fh	R/W-0	R/W-0

23 – 18	17	16
CS4_PG_DEL	CS4_PG_SIZE	CS4_PG_MD_EN
R/W-3Fh	R/W-0	R/W-0

15 – 10	9	8
CS3_PG_DEL	CS3_PG_SIZE	CS3_PG_MD_EN
R/W-3Fh	R/W-0	R/W-0

7 – 2	1	0
CS2_PG_DEL	CS2_PG_SIZE	CS2_PG_MD_EN
R/W-3Fh	R/W-0	R/W-0

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

Table 17-36. Page Mode Control Register (PMCR) Field Descriptions

Bit	Field	Value	Description
31-26	CS5_PG_DEL	1-3Fh	Page access delay for NOR Flash connected on CS5. CS5 is not available on this device.
25	CS5_PG_SIZE		Page Size for NOR Flash connected on CS5. CS5 is not available on this device.
24	CS5_PG_MD_EN		Page Mode enable for NOR Flash connected on CS5. CS5 is not available on this device.
23-18	CS4_PG_DEL	1-3Fh	Page access delay for NOR Flash connected on CS4. Number of EMIF_CLK cycles required for the page read data to be valid, minus 1 cycle. This value must not be cleared to 0.
17	CS4_PG_SIZE		Page Size for NOR Flash connected on CS4.
		0	Page size is 4 words.
		1	Page size is 8 words.
16	CS4_PG_MD_EN		Page Mode enable for NOR Flash connected on CS4.
		0	Page mode is disabled for this chip select.
		1	Page mode is enabled for this chip select.
15-10	CS3_PG_DEL	1-3Fh	Page access delay for NOR Flash connected on CS3. Number of EMIF_CLK cycles required for the page read data to be valid, minus 1 cycle. This value must not be cleared to 0.
9	CS3_PG_SIZE		Page Size for NOR Flash connected on CS3.
		0	Page size is 4 words.
		1	Page size is 8 words.
8	CS3_PG_MD_EN		Page Mode enable for NOR Flash connected on CS3.
		0	Page mode is disabled for this chip select.
		1	Page mode is enabled for this chip select.
7-2	CS2_PG_DEL	1-3Fh	Page access delay for NOR Flash connected on CS2. Number of EMIF_CLK cycles required for the page read data to be valid, minus 1 cycle. This value must not be cleared to 0.
1	CS2_PG_SIZE		Page Size for NOR Flash connected on CS2.
		0	Page size is 4 words.
		1	Page size is 8 words.
0	CS2_PG_MD_EN		Page Mode enable for NOR Flash connected on CS2.
		0	Page mode is disabled for this chip select.
		1	Page mode is enabled for this chip select.

17.4 Example Configuration

This section presents an example of interfacing the EMIF to both an SDR SDRAM device and an asynchronous flash device.

17.4.1 Hardware Interface

Figure 17-27 shows the hardware interface between the EMIF, a Samsung K4S641632H-TC(L)70 64Mb SDRAM device, and two SHARP LH28F800BJE-PTTL90 8Mb Flash memory. The connection between the EMIF and the SDRAM is straightforward, but the connection between the EMIF and the flash deserves a detailed look.

The address inputs for the flash are provided by three sources. The A[18:0] address inputs are provided by a combination of the EMIF_A and EMIF_BA pins according to Section 17.2.6.1. RD/nBY signal from one flash is connected to EMIF_nWAIT pin of EMIF.

Finally, this example configuration connects the EMIF_nWE pin to the nWE input of the flash and operates the EMIF in Select Strobe Mode.

17.4.2 Software Configuration

The following sections describe how to configure the EMIF registers and bit fields to interface the EMIF with the Samsung K4S641632H-TC(L)70 SDRAM and the SHARP LH28F800BJE-PTTL90 8Mb Flash memory.

17.4.2.1 Configuring the SDRAM Interface

This section describes how to configure the EMIF to interface with the Samsung K4S641632H-TC(L)70 SDRAM with a clock frequency of f_{EMIF_CLK} = 100 MHz. Procedure A described in Section 17.2.5.5 is followed which assumes that the SDRAM power-up timing constraint were met during the SDRAM Auto-Initialization sequence after Reset.

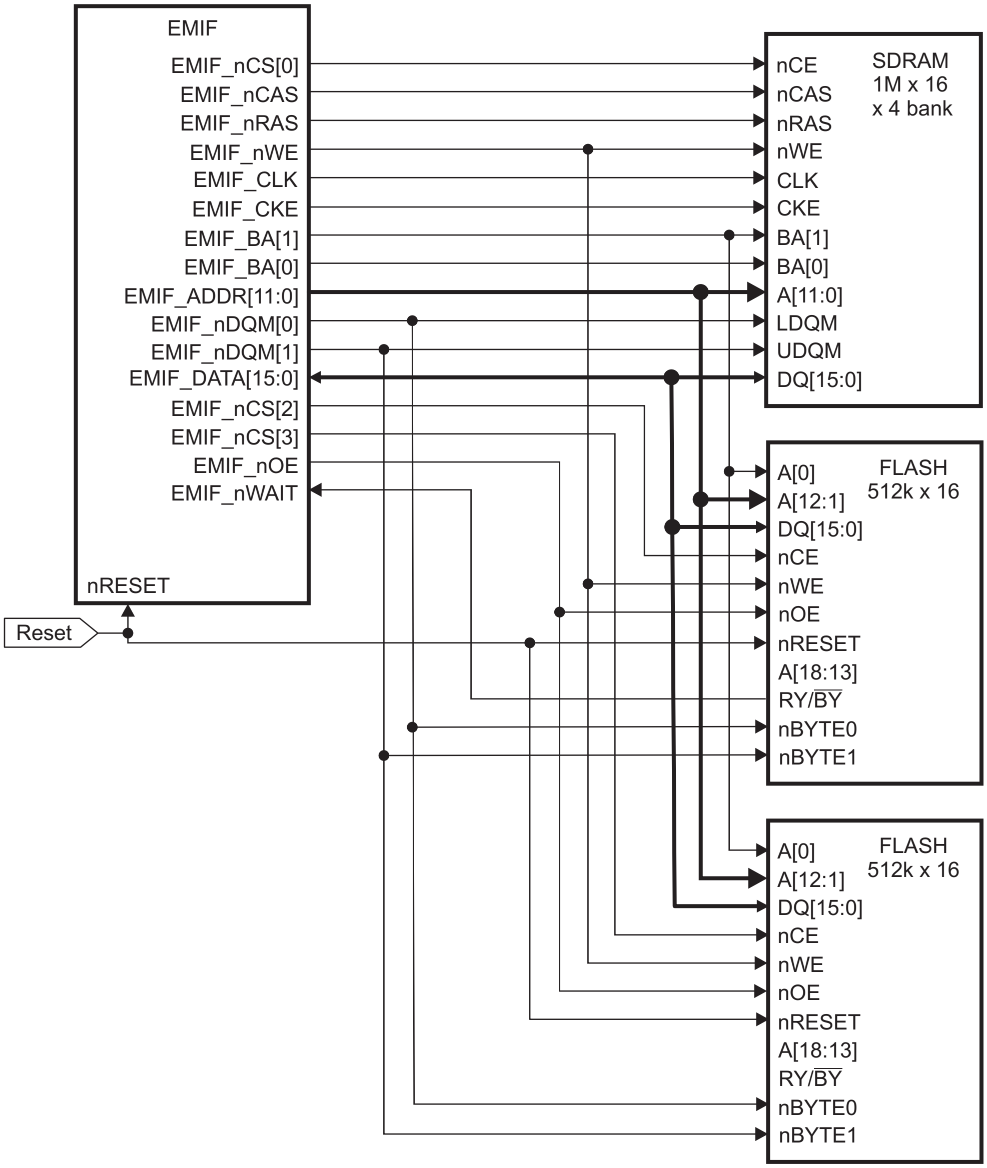
17.4.2.1.1 PLL Programming for the EMIF to K4S641632H-TC(L)70 Interface

The device global clock module (GCM) should first be programmed to select the desired EMIF_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 EMIF_CLK frequency can now be configured to the desired value by selecting the appropriate clock source for the VCLK3 domain. Once the VCLK3 domain frequency has been configured, remove the SDRAM from Self-Refresh by clearing the SR bit in SDCR, again with a byte-write.

Table 17-37. SR Field Value For the EMIF to K4S641632H-TC(L)70 Interface

Field	Value	Purpose
SR	1 then 0	To place the EMIF into the self refresh state

Figure 17-27. Example Configuration Interface



17.4.2.1.2 SDRAM Timing Register (SDTIMR) Settings for the EMIF to K4S641632H-TC(L)70 Interface

The fields of the SDRAM timing register (SDTIMR) should be programmed first as described in Table 17-38 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 17-28 shows a graphical description of how SDTIMR should be programmed.

Table 17-38. SDTIMR Field Calculations for the EMIF to K4S641632H-TC(L)70 Interface

Field Name	Formula	Value from K4S641632H-TC(L)70 Datasheet	Value Calculated for Field
T_RFC	T_RFC >= ($t_{RFC} \times f_{EMIF\_CLK}$) - 1	t_{RC} = 68 ns (min)[1]	6
T_RP	T_RP >= ($t_{RP} \times f_{EMIF\_CLK}$) - 1	t_{RP} = 20 ns (min)	1
T_RCD	T_RCD >= ($t_{RCD} \times f_{EMIF\_CLK}$) - 1	t_{RCD} = 20 ns (min)	1
T_WR	T_WR >= ($t_{WR} \times f_{EMIF\_CLK}$) - 1	t_{RDL} = 2 CLK = 20 ns (min)[2]	1
T_RAS	T_RAS >= ($t_{RAS} \times f_{EMIF\_CLK}$) - 1	t_{RAS} = 49 ns (min)	4
T_RC	T_RC >= ($t_{RC} \times f_{EMIF\_CLK}$) - 1	t_{RC} = 68 ns (min)	6
T_RRD	T_RRD >= ($t_{RRD} \times f_{EMIF\_CLK}$) - 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 17-28. 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

17.4.2.1.3 SDRAM Self Refresh Exit Timing Register (SDSRETR) Settings for the EMIF 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 17-39 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 17-29 shows how SDSRETR should be programmed.

Table 17-39. RR Calculation for the EMIF 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_{EMIF_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 17-29. 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

17.4.2.1.4 SDRAM Refresh Control Register (SDRCR) Settings for the EMIF 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 17-40 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 17-30 shows how SDRCR should be programmed.

Table 17-40. RR Calculation for the EMIF to K4S641632H-TC(L)70 Interface

Field Name	Formula	Values	Value Calculated for Field
RR	RR ≤ f_{EMIF_CLK} × $t_{Refresh\ Period}$ / n_{cycles}	From SDRAM datasheet: $t_{Refresh\ Period}$ = 64 ms; n_{cycles} = 4096 EMIF clock rate: f_{EMIF_CLK} = 100 MHz	RR = 1562 cycles = 61Ah cycles

Figure 17-30. 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

17.4.2.1.5 SDRAM Configuration Register (SDCR) Settings for the EMIF to K4S641632H-TC(L)70 Interface

Finally, the fields of the SDRAM configuration register (SDCR) should be programmed as described in Table 17-37 to properly interface with the K4S641632H-TC(L)70 device. Based on these settings, a value of 4720h should be written to SDCR. Figure 17-31 shows how SDCR should be programmed. The EMIF is now ready to perform read and write accesses to the SDRAM.

Table 17-41. SDCR Field Values For the EMIF to K4S641632H-TC(L)70 Interface

Field	Value	Purpose
SR	0	To avoid placing the EMIF into the self refresh state
NM	1	To configure the EMIF 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 17-31. 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

17.4.2.2 Configuring the Flash Interface

This section describes how to configure the EMIF to interface with the two of SHARP LH28F800BJE-PTTL90 8Mb Flash memory with a clock frequency of f_{EMIF_CLK} = 100 MHz. The example assumes that one flash is connected to EMIF_nCS2 and the other to EMIF_nCS3.

17.4.2.2.1 Asynchronous 1 Configuration Register (CE2CFG) Settings for the EMIF to LH28F800BJE-PTTL90 Interface

The asynchronous 1 configuration register (CE2CFG) and asynchronous 2 configuration register (CE3CFG) are the only registers that is necessary to program for this asynchronous interface (assuming that one Flash is connected to EMIF_nCS[2] and the other to EMIF_nCS[3]. The SS bit (in both registers) should be set to 1 to enable Select Strobe Mode and the ASIZE field (in both registers) should be set to 1 to select a 16-bit interface. The other fields in this register control the shaping of the EMIF signals, and the proper values can be determined by referring to the AC Characteristics in the Flash datasheet and the device datasheet. Based on the following calculations, a value of 8862 25BDh should be written to CE2CFG. Table 17-42 and Table 17-43 show the pertinent AC Characteristics for reads and writes to the Flash device, and Figure 17-32 and Figure 17-33 show the associated timing waveforms. Finally, Figure 17-34 shows programming the CE*n*CFG (*n* = 2, 3) with the calculated values.

Table 17-42. AC Characteristics for a Read Access

AC Characteristic	Device	Definition	Min	Max	Unit
t_{SU}	EMIF	Setup time, read EMIF_D before EMIF_CLK high	6.5		ns
t_H	EMIF	Data hold time, read EMIF_D after EMIF_CLK high	1		ns
t_D	EMIF	Output delay time, EMIF_CLK high to output signal valid		7	ns
t_{ELQV}	Flash	nCE to Output Delay		90	ns
t_{EHQZ}	Flash	nCE High to Output in High Impedance		55	ns

Table 17-43. AC Characteristics for a Write Access

AC Characteristic	Device	Definition	Min	Max	Unit
t_{AVAV}	Flash	Write Cycle Time	90		ns
t_{ELEH}	Flash	nCE Pulse Width Low	50		ns
t_{EHEL}	Flash	nCE Pulse Width High (not shown in Figure 17-33)	30		ns

Figure 17-32. LH28F800BJE-PTTL90 to EMIF Read Timing Waveforms

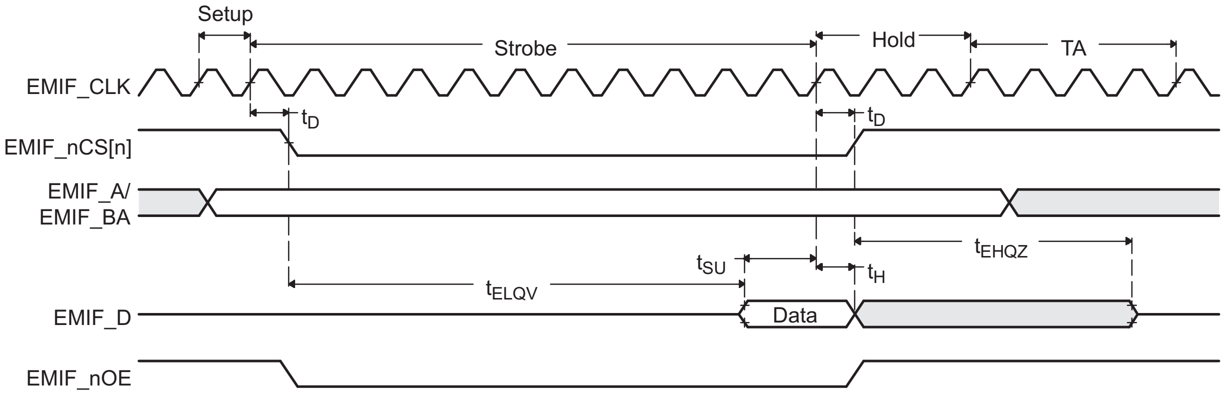
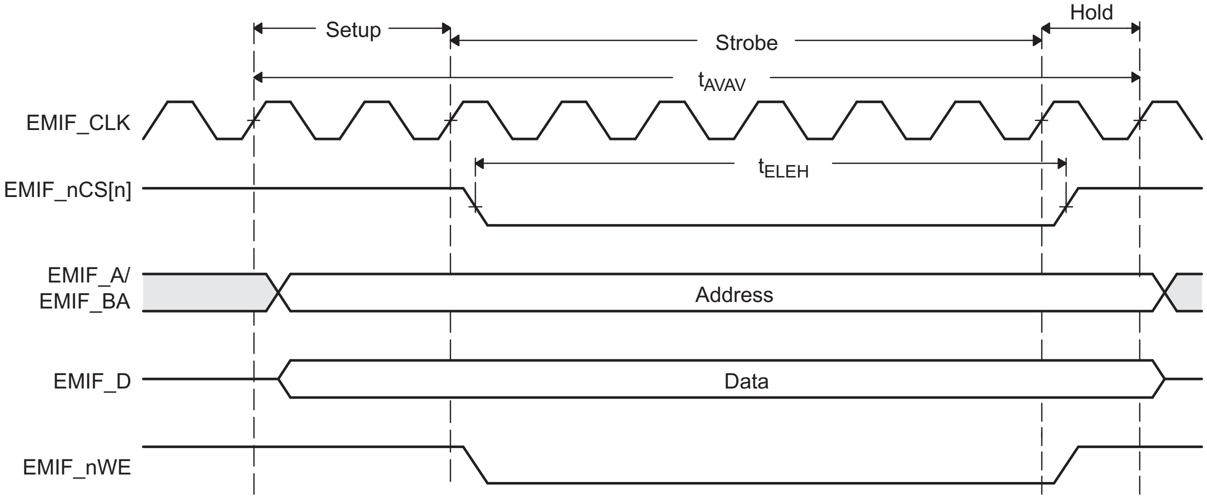


Figure 17-33. LH28F800BJE-PTTL90 to EMIF Write Timing Waveforms



The R_STROBE field should be set to meet the following equation:

R_STROBE >= (t_D + t_{ELQV} + t_{SU}) × f_{EMIF_CLK} - 1

R_STROBE >= (7 ns + 90 ns + 6.5 ns) × 100 MHz - 1

R_STROBE >= 9.35

R_STROBE = 10

The R_HOLD field must be large enough to satisfy the EMIF Data hold time, t_H:

R_HOLD > = t_H × f_{EMIF_CLK} - 1

R_HOLD >= 1 ns × 100 MHz - 1

R_HOLD >= -0.9

The R_HOLD field must also combine with the TA field to satisfy the Flash's nCE High to Output in High Impedance time, t_{EHQZ}:

R_HOLD + TA >= (t_D + t_{EHQZ}) × f_{EMIF_CLK} - 2

R_HOLD + TA >= (7 ns + 55 ns) × 100 MHz - 2

R_HOLD + TA >= 4.2

The largest value that can be programmed into the TA field is 3h, therefore the following values can be used:

R_HOLD = 2

TA = 3

For Writes, the W_STROBE field should be set to satisfy the Flash's nCE Pulse Width constraint, t_{ELEH}:

W_STROBE >= t_{ELEH} × f_{EMIF_CLK} - 1

W_STROBE >= 50 ns × 100 MHz - 1

W_STROBE >= 4

The W_SETUP and W_HOLD fields should combine to satisfy the Flash's nCE Pulse Width High constraint, t_{EHEL}, when performing back-to-back writes:

W_SETUP + W_HOLD > = $t_{EHEL} \times f_{EMIF\_CLK} - 2$

W_SETUP + W_HOLD > = 30 ns × 100 MHz - 2

W_SETUP + W_HOLD > = 1

In addition, the entire Write access length must satisfy the Flash's minimum Write Cycle Time, t_{AVAV}:

W_SETUP + W_STROBE + W_HOLD >= $t_{AVAV} \times f_{EMIF\_CLK} - 3$

W_SETUP + W_STROBE + W_HOLD >= 90 ns × 100 MHz - 3

W_SETUP + W_STROBE + W_HOLD >= 6

Solving the above equations for the Write fields results in the following possible solution:

W_SETUP = 1

W_STROBE = 5

W_HOLD = 0

Adding a 10 ns (1 cycle) margin to each of the periods (excluding TA which is already at its maximum) in this example produces the following recommended values:

W_SETUP = 2h

W_STROBE = 6h

W_HOLD = 1h

R_SETUP = 1h

R_STROBE = Bh

R_HOLD = 3h

TA = 3h

Figure 17-34. Asynchronous *m* Configuration Register (*m* = 1, 2) (CE*n*CFG (*n* = 2, 3))

31	30	29 - 26	25 - 24
1	0	0010	00
SS	EW	W_SETUP	W_STROBE

23 - 20	19 - 17	16
0110	001	0
W_STROBE	W_HOLD	R_SETUP

15 - 13	12 - 7	6 - 4	3 - 2	1 - 0
001	001011	011	11	01
R_SETUP	R_STROBE	R_HOLD	TA	ASIZE

Chapter 18

SPNU514C–March 2018

Parameter Overlay Module (POM)

This chapter describes the parameter overlay module (POM).

Topic **Page**

18.1 Introduction 705
18.2 Module Operation 706
18.3 POM Control Registers 708

18.1 Introduction

In many applications it is important to be able to change certain parameters in the program without having to re-flash the device and immediately test these changes either in a hardware-in-the-loop simulation or in a real environment.

The POM provides a mechanism to redirect accesses to non-volatile memory into a volatile memory internal or external to the device. The data requested by the CPU will be fetched from the overlay memory instead of the main non-volatile memory.

18.1.1 Main Features

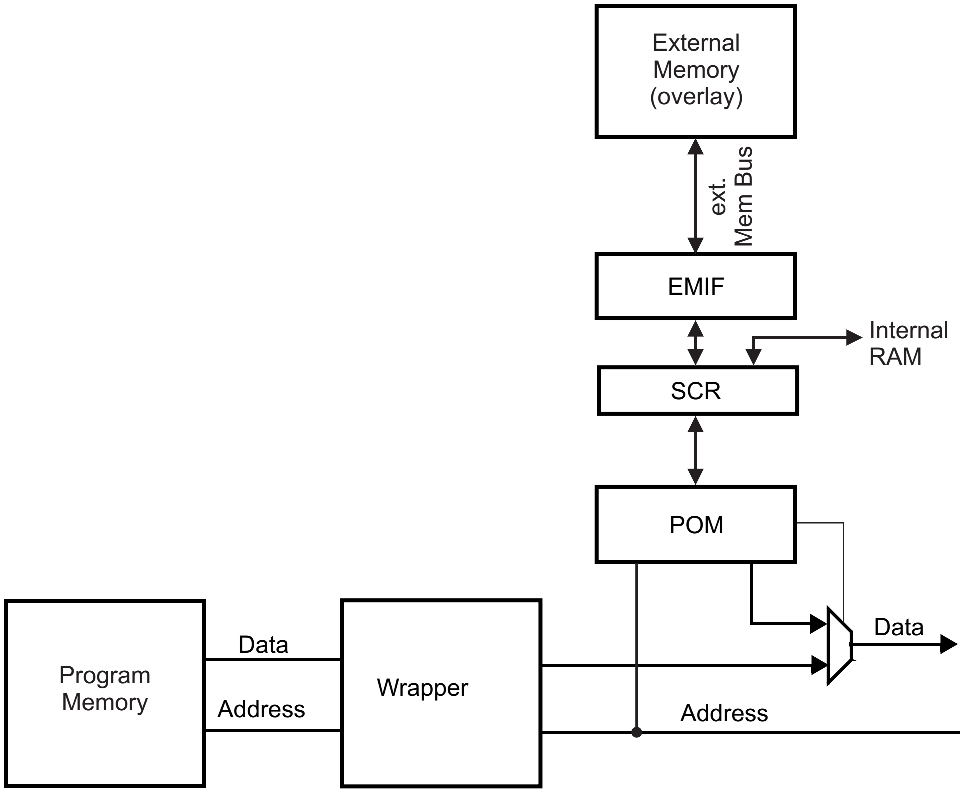
- Redirects program memory accesses to internal or external memory (overlay memory)
- Up to 8 MByte of external overlay memory
- Provides up to 32 programmable memory regions to replace non-volatile memory
  - Programmable region start address
  - Programmable region size (64 Bytes up to 256 kBytes in power of 2 steps)

18.1.2 Parameter Overlay Module (POM) Considerations

- The POM can map onto up to 8MB of the internal or external memory space. The starting address and the size of the memory overlay are configurable through the POM control registers. Care must be taken to ensure that the overlay is mapped on to available memory.
- ECC must be disabled by software through CP15 in case POM overlay is enabled; otherwise ECC errors will be generated.
- POM overlay must not be enabled when the flash and internal RAM memories are swapped through the MEM SWAP field of the Bus Matrix Module Control Register 1 (BMMCR1).
- When POM is used to overlay the flash on to internal or external RAM, there is a bus contention possibility when another master accesses the TCM flash. This results in a system hang.
  - The POM implements a timeout feature to detect this exact scenario. The timeout needs to be enabled whenever POM overlay is enabled.
  - The timeout can be enabled by writing Ah to the enable timeout (ETO) field of the POM global control register (POMGLBCTRL).
  - In case a read request by the POM cannot be completed within 32 HCLK cycles, the timeout (TO) flag is set in the POM status register (POMFLG). Also, an abort is generated to the CPU. This can be a prefetch abort for an instruction fetch or a data abort for a data fetch.
  - The prefetch-abort and data-abort handlers must be modified to check if the TO flag in the POM is set. If so, then the application can assume that the timeout is caused by a bus contention between the POM transaction and another master accessing the same memory region. The abort handlers need to clear the TO flag, so that any further aborts are not misinterpreted as having been caused due to a timeout from the POM.

18.1.3 Block Diagram

Figure 18-1. System Overlay Block Diagram



18.2 Module Operation

The POM has up to 32 programmable regions. Whenever the CPU requests data from the non-volatile memory which address falls into one of the programmed regions, the POM will request the data from the overlay memory. Wait states will be automatically inserted until the data is available from the overlay memory. This ensures that the overlay memory has the same (in the case that the latency from overlay memory is less than the program memory latency) or slower access time than the program memory.

The POM does not provide a feature to write into the overlay memory. The write has to be performed directly to the memory mapped address space of the overlay memory.

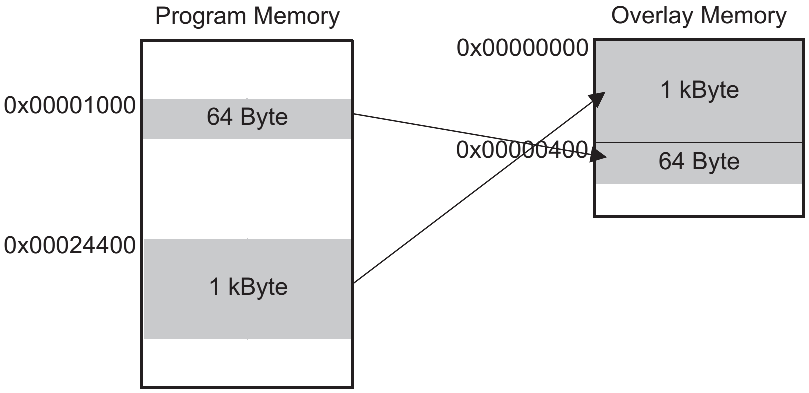
When the module is disabled, no redirection of access will be performed.

NOTE: The overlay feature is not available for any other bus masters other than the main CPU. Any attempt to access overlaid memory via the POM by any other bus master would result in a deadlock situation (system hang). Other bus masters need to access the target internal or external memory directly.

18.2.1 Decode Regions

There are 32 decode regions. Regions are defined by a start address and a region size. The start address is 23 bits wide to cover the maximum 8 MByte address space defined for program memory. The region size ranges from 64 Bytes to 256 kBytes with a step size of power of 2. If a region size of 0 is selected, the region is disabled. To support overlay memory that is not the same size as the program memory, an overlay start address that is 23-bits wide can be specified. The size of the overlay region is the same as the program memory region. The start address of both program memory and overlay memory region have to be a multiple of the programmed region size. When the address of the access falls into one of the programmed regions, the POM will start requesting the data from the overlay memory address, based on the overlay start address. If the address falls into multiple overlapping regions, the region with the lowest number has highest priority and only one read request from overlay memory will be initiated. This avoids that regions with the same program memory address, but different overlay memory addresses, request different data.

Figure 18-2. Region Definition Example



18.2.2 Bus Errors on Accesses via POM

In case a read by the POM cannot be completed within 32 HCLK cycles due to other transactions ongoing or due to other bus errors, a timeout mechanism is implemented. This timeout mechanism is disabled by default and must be enabled by writing Ah to the enable timeout (ETO) field of the POM global control register (POMGLBCTRL).

An abort is generated to the host CPU when this timeout occurs. The host CPU responds by either taking the prefetch abort or the data abort exceptions depending on the nature of the access that caused the timeout.

The abort handler can check if the TO flag is set in the POM module's flag register (POMFLG). If this flag is set, then the application can assume that the timeout is caused by the POM access not completing. This flag must be cleared so that any further aborts are not misinterpreted as having been caused by timeout on a POM access.

18.3 POM Control Registers

Table 18-1 lists the Parameter Overlay Module registers. The registers support only 32-bit writes. The offset is relative to the associated peripheral select. The base address for the control packets is FFA0 4000h.

Table 18-1. POM Registers

Offset	Acronym	Register Description	Section
00h	POMGLBCTRL	POM Global Control Register	Section 18.3.1
04h	POMREV	POM Revision ID	Section 18.3.2
08h	POMCLKCTRL	POM Clock Gate Control Register	Section 18.3.3
0Ch	POMFLG	POM Status Register	Section 18.3.4
200h, 210h, ..., 3F0h	POMPROGSTARTx	POM Program Region Start Address Register x	Section 18.3.5
204h, 214h, ..., 3F4h	POMOVLSTARTx	POM Overlay Region Start Address Register x	Section 18.3.6
208h, 218h, ..., 3F8h	POMREGSIZEx	POM Region Size Register x	Section 18.3.7
F00h	POMITCTRL	POM Integration Control Register	Section 18.3.8
FA0h	POMCLAIMSET	POM Claim Set Register	Section 18.3.9
FA4h	POMCLAIMCLR	POM Claim Clear Register	Section 18.3.10
FB0h	POMLOCKACCESS	POM Lock Access Register	Section 18.3.11
FB4h	POMLOCKSTATUS	POM Lock Status Register	Section 18.3.12
FB8h	POMAUTHSTATUS	POM Authentication Status Register	Section 18.3.13
FC8h	POMDEVID	POM Device ID Register	Section 18.3.14
FCCh	POMDEVTYPE	POM Device Type Register	Section 18.3.15
FD0h	POMPERIPHERALID4	POM Peripheral ID 4 Register	Section 18.3.16
FD4h	POMPERIPHERALID5	POM Peripheral ID 5 Register	Section 18.3.17
FD8h	POMPERIPHERALID6	POM Peripheral ID 6 Register	Section 18.3.18
FDCh	POMPERIPHERALID7	POM Peripheral ID 7 Register	Section 18.3.19
FE0h	POMPERIPHERALID0	POM Peripheral ID 0 Register	Section 18.3.20
FE4h	POMPERIPHERALID1	POM Peripheral ID 1 Register	Section 18.3.21
FE8h	POMPERIPHERALID2	POM Peripheral ID 2 Register	Section 18.3.22
FECh	POMPERIPHERALID3	POM Peripheral ID 3 Register	Section 18.3.23
FF0h	POMCOMPONENTID0	POM Component ID 0 Register	Section 18.3.24
FF4h	POMCOMPONENTID1	POM Component ID 1 Register	Section 18.3.25
FF8h	POMCOMPONENTID2	POM Component ID 2 Register	Section 18.3.26
FFCh	POMCOMPONENTID3	POM Component ID 3 Register	Section 18.3.27

18.3.1 POM Global Control Register (POMGLBCTRL)

This register contains a key to enable the POM module. Logic remains reset until this key is set.

Figure 18-3. POM Global Control Register (POMGLBCTRL) [address = FFA0 4000h]

31 23	22 16
OTADDR	Reserved
R/WP-60h	R-0

15 12	11 8	7 4	3 0
Reserved	ETO	Reserved	ON/OFF
R-0	R/WP-5h	R-0	R/WP-5h

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

Table 18-2. POM Global Control Register (POMGLBCTRL) Field Descriptions

Bit	Field	Value	Description
31-23	OTADDR	60h	**Overlay target Address**. These bits determine the upper address bits of the target overlay address. Writing a different value to this field will steer the POM access to a different location in the 4GB address space. The application must ensure that the overlay memory address points to a valid internal or external memory location.
22-12	Reserved	0	Reads return 0, writes have no effect.
11-8	ETO		**Enable Timeout**. Refer to Section 18.2.2 for more details on the timeout error.
		Ah	Timeout for bus transactions is enabled.
		All other values	Timeout for bus transactions is disabled. The timeout is disabled by default.
7-4	Reserved	0	Reads return zeros, writes have no effect.
3-0	ON/OFF		**Turn functionality of POM on or off.**
		Ah	POM is functional.
		All other values	POM is held in reset.
			NOTE: The key should be written to 5h, to avoid single bit flips inadvertently turning on the module.

18.3.2 POM Revision ID (POMREV)

This register contains the revision ID of the POM module.

Figure 18-4. POM Revision ID (POMREV) [address = FFA0 4004h]

31 30	29 28	27 16
SCHEME	Reserved	FUNC
R-1h	R-0	R-A03h

15 11	10 8	7 6	5 0
RTL	MAJOR	CUSTOM	MINOR
R-0	R-1h	R-0	R-8h

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

Table 18-3. POM Revision ID (POMREV) Field Descriptions

Bit	Field	Value	Description
31-30	SCHEME	1h	Used to distinguish between different ID schemes.
29-28	Reserved	0	Reads return 0, writes have no effect.
27-16	FUNC	A03h	Indicates the SW compatible module family.
15-11	RTL	0	RTL version number.
10-8	MAJOR	1h	Major revision number.
7-6	CUSTOM	0	Indicates a device specific implementation.
5-0	MINOR	8h	Minor revision number.

18.3.3 POM Clock Gate Control Register (POMCLKCTRL)

This register is for TI internal use only.

Figure 18-5. POM Clock Gate Control Register [address = FFA0 4008h]

31 16
Reserved
R-0

15	0
Reserved	CLK GATE OFF
R-0	R/W-0

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

Table 18-4. POM Clock Gate Control Register (POMCLKCTRL) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0, writes have no effect.
0	CLK GATE OFF	0	Do not modify this bit. Leave it in its reset state. Modifying the bit while the POM module is switched on can result in unexpected behavior.

18.3.4 POM Status Register (POMFLG)

This register provides POM status information.

Figure 18-6. POM Status Register [address = FFA0 400Ch]

31	16
Reserved	
R-0	

15	0
Reserved	TO
R-0	R/W1CP-0

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 to clear in privilege mode only; -*n* = value after reset

Table 18-5. POM Status Register (POMFLG) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0, writes have no effect.
0	TO		**Timeout**. This flag signals a timeout condition on the last CPU read access through the POM. This flag is only updated when the POM module is enabled (ON/OFF = Ah).
		0	Read: No timeout occurred. Write: No effect.
		1	Read: Timeout occurred. Write: Bit is cleared.

18.3.5 POM Program Region Start Address Register x (POMPROGSTARTx)

This register contains the start address of the region in program memory. x goes from 0 to 31, since up to 32 regions can be defined.

Figure 18-7. POM Program Region Start Register x (POMPROGSTARTx) [address = FFA0 4200h, FFA0 4210h, ..., FFA0 43F0h]

31 23	22 16
Reserved	STARTADDRESS
R-0	R/WP-0

15 0
STARTADDRESS
R/WP-0

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

Table 18-6. POM Program Region Start Address Register x (POMPROGSTARTx) Field Descriptions

Bit	Field	Value	Description
31-23	Reserved	0	Reads return 0, writes have no effect.
22-0	STARTADDRESS		Defines the start address of the program memory region. The start address has to be a multiple of the region size.

NOTE: If the region start address is programmed to a non-region size boundary, the region will begin at the next lower region size boundary.

18.3.6 POM Overlay Region Start Address Register x (POMOVLSTARTx)

This register contains the start address of the region in overlay memory. x goes from 0 to 31, since up to 32 regions can be defined.

Figure 18-8. POM Overlay Region Start Register x (POMOVLSTARTx) [address = FFA0 4204h, FFA0 4214h, ..., FFA0 43F4h]

31 23	22 16
Reserved	STARTADDRESS
R-0	R/W-0

15 0
STARTADDRESS
R/W-0

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

Table 18-7. POM Overlay Region Start Address Register x (POMOVLSTARTx) Field Descriptions

Bit	Field	Value	Description
31-23	Reserved	0	Reads return 0, writes have no effect.
22-0	STARTADDRESS		Defines the start address of the overlay memory region. The start address has to be a multiple of the region size.

NOTE: If the region start address is programmed to a non-region size boundary, the region will begin at the next lower region size boundary.

18.3.7 POM Region Size Register x (POMREGSIZEx)

This register contains the region size for both program memory and overlay memory. x goes from 0 to 31, since up to 32 regions can be defined.

Figure 18-9. POM Region Size Register x (POMREGSIZEx) [address = FFA0 4208h, FFA0 4218h, ..., FFA0 43F8h]

31	16
Reserved	
R-0	

15	4	3	0
Reserved		SIZE	
R-0		R/W-0	

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

Table 18-8. POM Region Size Register x (POMREGSIZEx) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0, writes have no effect.
3-0	SIZE		**Region size**
		0	Region is disabled.
		1h	64 Bytes
		2h	128 Bytes
		:	:
		Dh	256 kBytes
		Eh-Fh	Reserved

NOTE: If the region is enabled by writing a non-zero value to the SIZE bitfield, it will take some number of VCLK cycles until the write takes effect. If during this time an access to the programmed region is taking place and the POM is already enabled in the POMGLBCTRL register, it either decodes the previous setting of the SIZE field or the access will be directed to the program memory when the region was disabled.

18.3.8 POM Integration Control Register (POMITCTRL)

This is a CoreSight register and is for debug purpose only. The integration functionality is not implemented. The register reads 00000000.

Figure 18-10. POM Integration Control Register (POMITCTRL) [address = FFA0 4F00h]

31	0
Reserved	
R-0	

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

Table 18-9. POM Integration Control Register (POMITCTRL) Field Descriptions

Bit	Field	Value	Description
31-0	Reserved	0	Reads return 0, writes have no effect.

18.3.9 POM Claim Set Register (POMCLAIMSET)

This is a CoreSight register and is for debug purpose only. This register allows different masters to claim the control for the POM module. There is no control functionality for POM register accesses implemented. The only functionality of these bits is to indicate that another master is controlling the module.

Figure 18-11. POM Claim Set Register (POMCLAIMSET) [address = FFA0 4FA0h]

31	16
Reserved	
R-0	

15	2	1	0
Reserved		SET1	SET0
R-0		R/W-1	R/W-1

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

Table 18-10. POM Claim Set Register (POMCLAIMSET) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reads return 0, writes have no effect.
1	SET1	0	The module is claimed. Read: This claim tag bit is not implemented. Write: No effect.
		1	Read: This claim tag is implemented. Write: Set claim tag.
0	SET0	0	The module is claimed. Read: This claim tag bit is not implemented. Write: No effect.
		1	Read: This claim tag is implemented. Write: Set claim tag.

18.3.10 POM Claim Clear Register (POMCLAIMCLR)

This is a CoreSight register and is for debug purpose only. This register allows different masters to claim the control for the POM module. There is no control functionality for POM register accesses implemented. The only functionality of these bits is to indicate that another master is controlling the module.

Figure 18-12. POM Claim Clear Register (POMCLAIMCLR) [address = FFA0 4FA4h]

31	16
Reserved	
R-0	

15	2	1	0
Reserved		CLR1	CLR0
R-0		R/W-1	R/W-1

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

Table 18-11. POM Claim Clear Register (POMCLAIMCLR) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reads return 0, writes have no effect.
1	CLR1		The module is claimed.
			Read: Current claim tag value.
		0	Write: No effect.
		1	Write: Clear claim tag.
0	CLR0		The module is claimed.
			Read: Current claim tag value.
		0	Write: No effect.
		1	Write: Clear claim tag.

18.3.11 POM Lock Access Register (POMLOCKACCESS)

This is a CoreSight register and is for debug purpose only. The register reads 00000000.

Figure 18-13. POM Lock Access Register (POMLOCKACCESS) [address = FFA0 4FB0h]

31 ... 0
Reserved
R-0

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

Table 18-12. POM Lock Access Register (POMLOCKACCESS) Field Descriptions

Bit	Field	Value	Description
31-0	Reserved	0	Reads return 0, writes have no effect.

18.3.12 POM Lock Status Register (POMLOCKSTATUS)

This is a CoreSight register and is for debug purpose only. The register reads 00000000.

Figure 18-14. POM Lock Status Register (POMLOCKSTATUS) [address = FFA0 4FB4h]

31 ... 0
Reserved
R-0

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

Table 18-13. POM Lock Status Register (POMLOCKSTATUS) Field Descriptions

Bit	Field	Value	Description
31-0	Reserved	0	Reads return 0, writes have no effect.

18.3.13 POM Authentication Status Register (POMAUTHSTATUS)

This is a CoreSight register and is for debug purpose only. The register reads 00000000.

Figure 18-15. POM Authentication Status Register (POMAUTHSTATUS) [address = FFA0 4FB8h]

31 ... 0
Reserved
R-0

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

Table 18-14. POM Authentication Status Register (POMAUTHSTATUS) Field Descriptions

Bit	Field	Value	Description
31-0	Reserved	0	Reads return 0, writes have no effect.

18.3.14 POM Device ID Register (POMDEVID)

This is a CoreSight register and is for debug purpose only. The register reads 00000000.

Figure 18-16. POM Device ID Register (POMDEVID) [address = FFA0 4FC8h]

<table>
<tr><td>31</td><td>0</td></tr>
<tr><td colspan="2">Reserved</td></tr>
<tr><td colspan="2">R-0</td></tr>
</table>

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

Table 18-15. POM Device ID Register (POMDEVID) Field Descriptions

Bit	Field	Value	Description
31-0	Reserved	0	Reads return 0, writes have no effect.

18.3.15 POM Device Type Register (POMDEVTYPE)

This is a CoreSight register and is for debug purpose only. The device type register defines the CoreSight module class.

Figure 18-17. POM Device Type Register (POMDEVTYPE) [address = FFA0 4FCCh]

<table>
<tr><td>31</td><td>16</td></tr>
<tr><td colspan="2">Reserved</td></tr>
<tr><td colspan="2">R-0</td></tr>
</table>

<table>
<tr><td>15</td><td>8</td><td>7</td><td>4</td><td>3</td><td>0</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="2">Sub Type</td><td colspan="2">Major Type</td></tr>
<tr><td colspan="2">R-0</td><td colspan="2">R-0</td><td colspan="2">R-4h</td></tr>
</table>

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

Table 18-16. POM Device Type Register (POMDEVTYPE) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0, writes have no effect.
7-4	Sub Type	0	Other
3-0	Major Type	4h	Debug Control

18.3.16 POM Peripheral ID 4 Register (POMPERIPHERALID4)

This is a CoreSight register and is for debug purpose only. This register shows the peripheral identification of the CoreSight component.

Figure 18-18. POM Peripheral ID 4 Register (POMPERIPHERALID4) [address = FFA0 4FD0h]

31 – 16
Reserved
R-0

15 – 8	7 – 4	3 – 0
Reserved	4KB Count	JEP106 Continuation Code
R-0	R-0	R-0

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

Table 18-17. POM Peripheral ID 4 Register (POMPERIPHERALID4) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0, writes have no effect.
7-4	4KB Count	0	Only 4KB is implemented.
3-0	JEP Continuation Code	0	JEP106 Code

18.3.17 POM Peripheral ID 5 Register (POMPERIPHERALID5)

This is a CoreSight register and is for debug purpose only. This register reads 00000000.

Figure 18-19. POM Peripheral ID 5 Register (POMPERIPHERALID5) [address = FFA0 4FD4h]

31 – 0
Reserved
R-0

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

Table 18-18. POM Peripheral ID 5 Register (POMPERIPHERALID5) Field Descriptions

Bit	Field	Value	Description
31-0	Reserved	0	Reads return 0, writes have no effect.

18.3.18 POM Peripheral ID 6 Register (POMPERIPHERALID6)

This is a CoreSight register and is for debug purpose only. This register reads 00000000.

Figure 18-20. POM Peripheral ID 6 Register (POMPERIPHERALID6) [address = FFA0 4FD8h]

31 … 0
Reserved
R-0

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

Table 18-19. POM Peripheral ID 6 Register (POMPERIPHERALID6) Field Descriptions

Bit	Field	Value	Description
31-0	Reserved	0	Reads return 0, writes have no effect.

18.3.19 POM Peripheral ID 7 Register (POMPERIPHERALID7)

This is a CoreSight register and is for debug purpose only. This register reads 00000000.

Figure 18-21. POM Peripheral ID 7 Register (POMPERIPHERALID7) [address = FFA0 4FDCh]

31 … 0
Reserved
R-0

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

Table 18-20. POM Peripheral ID 7 Register (POMPERIPHERALID7) Field Descriptions

Bit	Field	Value	Description
31-0	Reserved	0	Reads return 0, writes have no effect.

18.3.20 POM Peripheral ID 0 Register (POMPERIPHERALID0)

This is a CoreSight register and is for debug purpose only. This register shows the peripheral identification of the CoreSight component.

Figure 18-22. POM Peripheral ID 0 Register (POMPERIPHERALID0) [address = FFA0 4FE0h]

31 … 16
Reserved
R-0

15 … 8	7 … 0
Reserved	Part Number
R-0	R-0

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

Table 18-21. POM Peripheral ID 0 Register (POMPERIPHERALID0) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0, writes have no effect.
7-0	Part Number	0	Reads 0, since POMREV defines the module.

18.3.21 POM Peripheral ID 1 Register (POMPERIPHERALID1)

This is a CoreSight register and is for debug purpose only. This register shows the peripheral identification of the CoreSight component.

Figure 18-23. POM Peripheral ID 1 Register (POMPERIPHERALID1) [address = FFA0 4FE4]

31 … 16
Reserved
R-0

15 … 8	7 … 4	3 … 0
Reserved	JEP106 Identity Code	Part Number
R-0	R-7h	R-0

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

Table 18-22. POM Peripheral ID 1 Register (POMPERIPHERALID1) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0, writes have no effect.
7-4	JEP106 Identity Code	7h	Part of TI JEDEC number.
3-0	Part Number	0	Reads 0, since POMREV defines the module.

18.3.22 POM Peripheral ID 2 Register (POMPERIPHERALID2)

This is a CoreSight register and is for debug purpose only. This register shows the peripheral identification of the CoreSight component.

Figure 18-24. POM Peripheral ID 2 Register (POMPERIPHERALID2) [address = FFA0 4FE8h]

31 - 16
Reserved
R-0

15 - 8	7 - 4	3	2 - 0
Reserved	Revision	JEDEC	JEP106 Identity Code
R-0	R-0	R-1h	R-1h

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

Table 18-23. POM Peripheral ID 2 Register (POMPERIPHERALID2) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0, writes have no effect.
7-4	Revision	0	Reads 0, since POMREV defines the module.
3	JEDEC	1h	Indicates JEDEC assigned value.
2-0	JEP106 Identity Code	1h	JEDEC+JEP106 Identity Code (POMPERIPHERALID2)+JEP106 Identity Code (POMPERIPHERALID1) form TI JEDEC ID of 97h.

18.3.23 POM Peripheral ID 3 Register (POMPERIPHERALID3)

This is a CoreSight register and is for debug purpose only. This register reads 00000000.

Figure 18-25. POM Peripheral ID 3 Register (POMPERIPHERALID3) [address = FFA0 4FECh]

31 - 0
Reserved
R-0

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

Table 18-24. POM Peripheral ID 3 Register (POMPERIPHERALID3) Field Descriptions

Bit	Field	Value	Description
31-0	Reserved	0	Reads return 0, writes have no effect.

18.3.24 POM Component ID 0 Register (POMCOMPONENTID0)

This is a CoreSight register and is for debug purpose only.

Figure 18-26. POM Component ID 0 Register (POMCOMPONENTID0) [address = FFA0 4FF0h]

31-16
Reserved
R-0

15-8	7-0
Reserved	Preamble
R-0	R-Dh

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

Table 18-25. POM Component ID 0 Register (POMCOMPONENTID0) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0, writes have no effect.
7-0	Preamble	Dh	Preamble

18.3.25 POM Component ID 1 Register (POMCOMPONENTID1)

This is a CoreSight register and is for debug purpose only.

Figure 18-27. POM Component ID 1 Register (POMCPOMPONENTID1) [address = FFA0 4FF4h]

31-16
Reserved
R-0

15-8	7-4	3-0
Reserved	Component Class	Preamble
R-0	R-9h	R-0

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

Table 18-26. POM Component ID 1 Register (POMCOMPONENTID1) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0, writes have no effect.
7-4	Component Class	9h	CoreSight Component
3-0	Preamble	0	Preamble

18.3.26 POM Component ID 2 Register (POMCOMPONENTID2)

This is a CoreSight register and is for debug purpose only.

Figure 18-28. POM Component ID 2 Register (POMCPOMPONENTID2) [address = FFA0 4FF8h]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		Preamble	
R-0		R-5h	

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

Table 18-27. POM Component ID 2 Register (POMCOMPONENTID2) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0, writes have no effect.
7-0	Preamble	5h	Preamble

18.3.27 POM Component ID 3 Register (POMCOMPONENTID3)

This is a CoreSight register and is for debug purpose only.

Figure 18-29. POM Component ID 3 Register (POMCPOMPONENTID3) [address = FFA0 4FFCh]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		Preamble	
R-0		R-B1h	

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

Table 18-28. POM Component ID 3 Register (POMCOMPONENTID3) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0, writes have no effect.
7-0	Preamble	B1h	Preamble

Chapter 19

SPNU514C–March 2018

Enhanced Pulse Width Modulator (ePWM) Module

The enhanced pulse width modulator (ePWM) peripheral is a key element in controlling many of the power electronic systems found in both commercial and industrial equipments. The features supported by the ePWM make it especially suitable for digital motor control.

Topic **Page**

19.1 Introduction **725**
19.2 ePWM Submodules **729**
19.3 Application Examples **788**
19.4 EPWM_MODULE_CONTROL_AND_STATUS_REGISTER Registers **803**

19.1 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 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 document the letter x within a signal or module name is used to indicate a generic ePWM instance on a device. For example, output signals EPWMxA and EPWMxB refer to the output signals from the ePWMx instance. Thus, EPWM1A and EPWM1B belong to ePWM1 and likewise EPWM4A and EPWM4B belong to ePWM4.

19.1.1 Submodule Overview

The ePWM module represents one complete PWM channel composed of two PWM outputs: EPWMxA and EPWMxB. Multiple ePWM modules are instanced within a device as shown in Figure 19-1. Each ePWM instance is identical and is indicated by a numerical value starting with 1. For example, ePWM1 is the first instance and ePWM3 is the third 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). Modules can also operate stand-alone.

Each ePWM module supports the following features:

- Dedicated 16-bit time-base counter with period and frequency control
- Two PWM outputs (EPWMxA and EPWMxB) that can be used in the following configurations:
  - Two independent PWM outputs with single-edge operation
  - Two independent PWM outputs with dual-edge symmetric operation
  - One independent PWM output with dual-edge asymmetric operation
- Asynchronous override control of PWM signals through software.
- Programmable phase-control support for lag or lead operation relative to other ePWM modules.
- Hardware-locked (synchronized) phase relationship on a cycle-by-cycle basis.
- Dead-band generation with independent rising and falling edge delay control.
- Programmable trip zone allocation of both cycle-by-cycle trip and one-shot trip on fault conditions.
- A trip condition can force either high, low, or high-impedance state logic levels at PWM outputs.
- All events can trigger both CPU interrupts and ADC start of conversion (SOC)
- Programmable event prescaling minimizes CPU overhead on interrupts.
- PWM chopping by high-frequency carrier signal, useful for pulse transformer gate drives.

Each ePWM module is connected to the input/output signals shown in Figure 19-1. The signals are described in detail in subsequent sections.

Each ePWM module consists of eight submodules and is connected within a system via the signals shown in Figure 19-2.

Figure 19-1. Multiple ePWM Modules

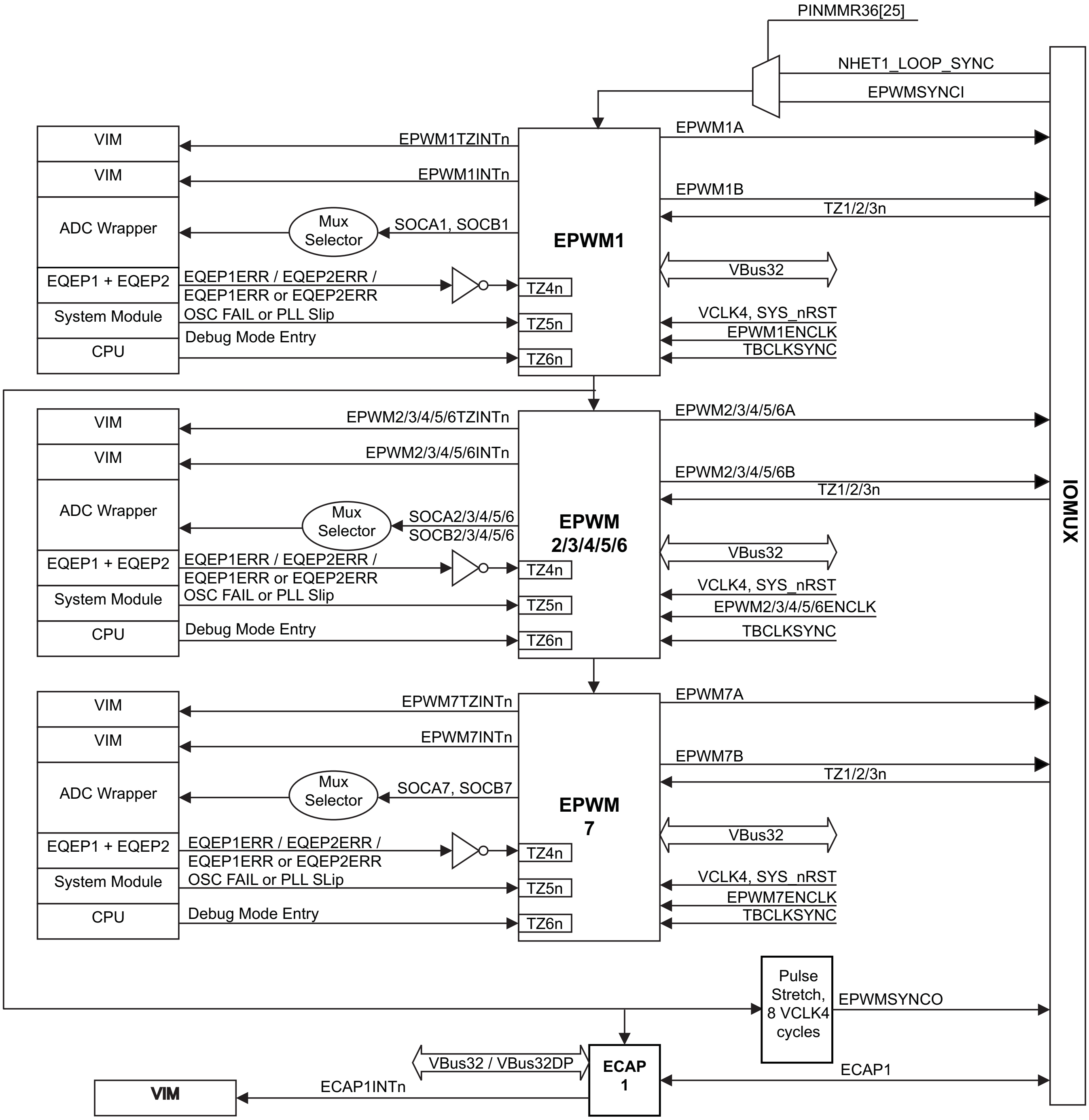
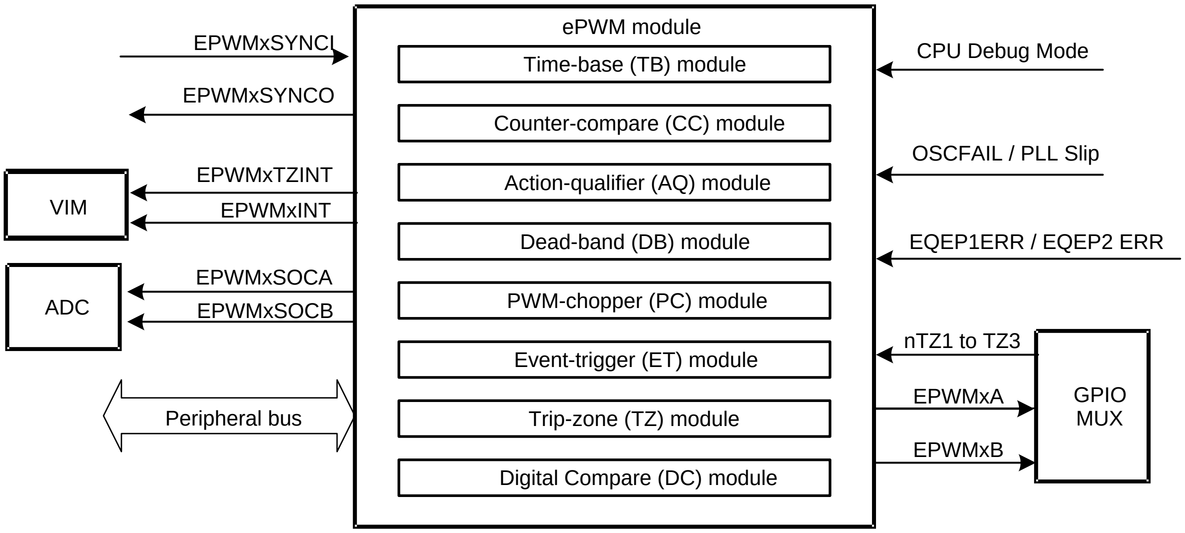


Figure 19-2. Submodules and Signal Connections for an ePWM Module



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 I/O Multiplexing Module (IOMM) as described in the IOMM chapter of the device's technical reference manual.

- **Trip-zone signals ($\overline{\text{TZ1}}$ to $\overline{\text{TZ6}}$).**

  These input signals alert the ePWM module of fault conditions external to the ePWM module. Each ePWM module can be configured to either use or ignore any of the trip-zone signals. The $\overline{\text{TZ1}}$ to $\overline{\text{TZ3}}$ trip-zone signals can be configured as asynchronous inputs, or double-synchronized using VCLK4, or double-synchronized and filtered through a 6-VCLK4-cycle counter before connecting to the ePWM modules. This selection is done by configuring registers in the IOMM. $\overline{\text{TZ4}}$ is connected to an inverted eQEP1 error signal (EQEP1ERR), or to an inverted eQEP2 error signal (EQEP2ERR), or an OR-combination of EQEP1ERR and EQEP2ERR. This selection is also done via the IOMM registers . $\overline{\text{TZ5}}$ is connected to the system clock fail status. This is asserted whenever an oscillator failure is detected, or a PLL slip is detected. $\overline{\text{TZ6}}$ is connected to the debug mode entry indicator output from the CPU. This allows you to configure a trip action when the CPU halts.

- **Time-base synchronization input (EPWMxSYNCI) and output (EPWMxSYNCO) signals**.

  The synchronization signals daisy chain the ePWM modules together. Each module can be configured 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).

- **ADC start-of-conversion signals (EPWMxSOCA and EPWMxSOCB).**

  Each ePWM module has two ADC start of conversion signals. Any ePWM module can trigger a start of conversion. Which event triggers the start of conversion is configured in the Event-Trigger submodule of the ePWM.

- **Peripheral Bus**

  The peripheral bus is 32-bits wide and allows both 16-bit and 32-bit writes to the ePWM register file.

19.1.2 Register Mapping

The complete ePWM module control and status register set is grouped by submodule as shown in Table 19-1. Each register set is duplicated for each instance of the ePWM module. The start address for each ePWM register file instance on a device is specified in the specific part's datasheet.

Table 19-1. ePWM Module Control and Status Register Set Grouped by Submodule

Name	Address Offset[1]	Size (×16)	Shadow	Privileged Mode Write Only?	Description
Time-Base Submodule Registers					
TBCTL	0x0000	1	No	No	Time-Base Control Register
TBSTS	0x0002	1	No	No	Time-Base Status Register
Reserved	0x0004	1	–	–	Reserved
TBPHS	0x0006	1	No	No	Time-Base Phase Register
TBCTR	0x0008	1	No	No	Time-Base Counter Register
TBPRD	0x000A	1	Yes	No	Time-Base Period Register
Reserved	0x000C	1	–	–	Reserved
Counter-Compare Submodule Registers					
CMPCTL	0x000E	1	No	No	Counter-Compare Control Register
Reserved	0x0010	1	–	–	Reserved
CMPA	0x0012	1	Yes	No	Counter-Compare A Register
CMPB	0x0014	1	Yes	No	Counter-Compare B Register
Action-Qualifier Submodule Registers					
AQCTLA	0x0016	1	No	No	Action-Qualifier Control Register for Output A (EPWMxA)
AQCTLB	0x0018	1	No	No	Action-Qualifier Control Register for Output B (EPWMxB)
AQSFRC	0x001A	1	No	No	Action-Qualifier Software Force Register
AQCSFRC	0x001C	1	Yes	No	Action-Qualifier Continuous S/W Force Register Set
Dead-Band Generator Submodule Registers					
DBCTL	0x001E	1	No	No	Dead-Band Generator Control Register
DBRED	0x0020	1	No	No	Dead-Band Generator Rising Edge Delay Count Register
DBFED	0x0022	1	No	No	Dead-Band Generator Falling Edge Delay Count Register
Trip-Zone Submodule Registers					
TZSEL	0x0024	1	No	Yes	Trip-Zone Select Register
TZDCSEL	0x0026	1	No	Yes	Trip Zone Digital Compare Select Register
TZCTL	0x0028	1	No	Yes	Trip-Zone Control Register
TZEINT	0x002A	1	No	Yes	Trip-Zone Enable Interrupt Register
TZFLG	0x002C	1	No	No	Trip-Zone Flag Register
TZCLR	0x002E	1	No	Yes	Trip-Zone Clear Register
TZFRC	0x0030	1	No	Yes	Trip-Zone Force Register
Event-Trigger Submodule Registers					
ETSEL	0x0032	1	No	No	Event-Trigger Selection Register
ETPS	0x0034	1	No	No	Event-Trigger Pre-Scale Register
ETFLG	0x0036	1	No	No	Event-Trigger Flag Register
ETCLR	0x0038	1	No	No	Event-Trigger Clear Register
ETFRC	0x003A	1	No	No	Event-Trigger Force Register
PWM-Chopper Submodule Registers					
PCCTL	0x003C	1	No	No	PWM-Chopper Control Register

[1] Locations not shown are reserved.

Table 19-1. ePWM Module Control and Status Register Set Grouped by Submodule (continued)

Name	Address Offset[1]	Size (×16)	Shadow	Privileged Mode Write Only?	Description
			Digital Compare Event Registers		
DCTRIPSEL	0x0060	1	No	Yes	Digital Compare Trip Select Register
DCACTL	0x0062	1	No	Yes	Digital Compare A Control Register
DCBCTL	0x0064	1	No	Yes	Digital Compare B Control Register
DCFCTL	0x0066	1	No	Yes	Digital Compare Filter Control Register
DCCAPCTL	0x0068	1	No	Yes	Digital Compare Capture Control Register
DCFOFFSET	0x006A	1	Writes	No	Digital Compare Filter Offset Register
DCFOFFSETCNT	0x006C	1	No	No	Digital Compare Filter Offset Counter Register
DCFWINDOW	0x006E	1	No	No	Digital Compare Filter Window Register
DCFWINDOWCNT	0x0070	1	No	No	Digital Compare Filter Window Counter Register
DCCAP	0x0072	1	Yes	No	Digital Compare Counter Capture Register

19.2 ePWM Submodules

Eight submodules are included in every ePWM peripheral. Each of these submodules performs specific tasks that can be configured by software.

19.2.1 Overview

Table 19-2 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 19.2.3 for relevant details.

Table 19-2. Submodule Configuration Parameters

Submodule	Configuration Parameter or Option
Time-base (TB)	• Scale the time-base clock (TBCLK) relative to the system clock (VCLK4). • Configure the PWM time-base counter (TBCTR) frequency or period. • Set the mode for the time-base counter: – count-up mode: used for asymmetric PWM – count-down mode: used for asymmetric PWM – count-up-and-down mode: used for symmetric PWM • Configure the time-base phase relative to another ePWM module. • Synchronize the time-base counter between modules through hardware or software. • Configure the direction (up or down) of the time-base counter after a synchronization event. • Configure how the time-base counter will behave when the device is halted by an emulator. • Specify the source for the synchronization output of the ePWM module: – Synchronization input signal – Time-base counter equal to zero – Time-base counter equal to counter-compare B (CMPB) – No output synchronization signal generated.
Counter-compare (CC)	• Specify the PWM duty cycle for output EPWMxA and/or output EPWMxB • Specify the time at which switching events occur on the EPWMxA or EPWMxB output

Table 19-2. Submodule Configuration Parameters (continued)

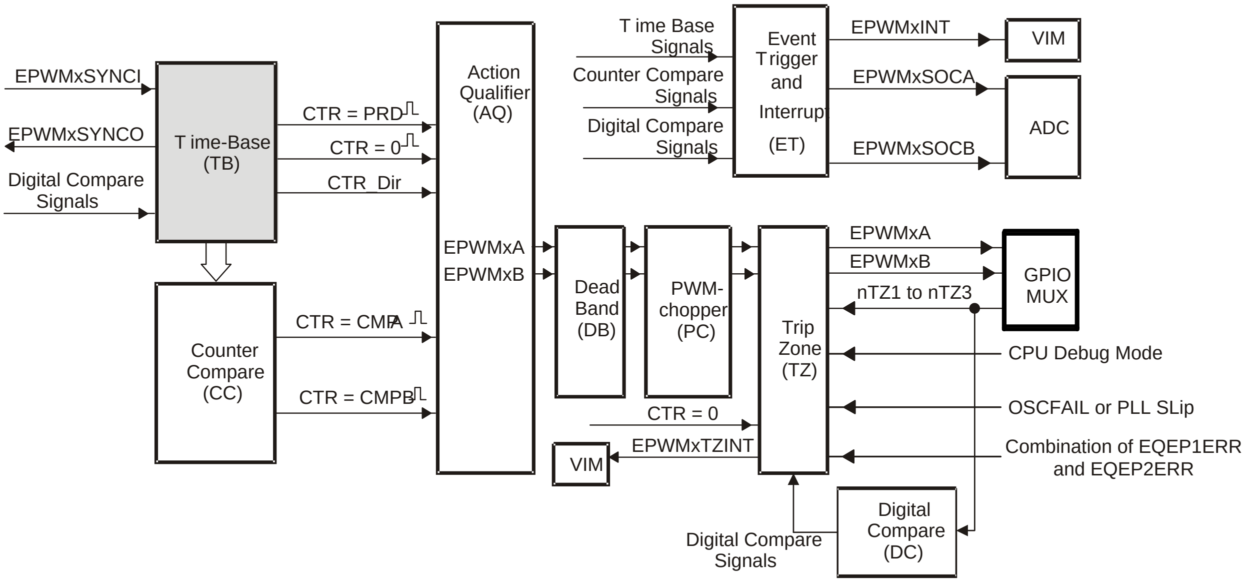
Submodule	Configuration Parameter or Option
Action-qualifier (AQ)	• Specify the type of action taken when a time-base or counter-compare submodule event occurs: – No action taken – Output EPWMxA and/or EPWMxB switched high – Output EPWMxA and/or EPWMxB switched low – Output EPWMxA and/or EPWMxB toggled • Force the PWM output state through software control • Configure and control the PWM dead-band through software
Dead-band (DB)	• Control of traditional complementary dead-band relationship between upper and lower switches • Specify the output rising-edge-delay value • Specify the output falling-edge delay value • Bypass the dead-band module entirely. In this case the PWM waveform is passed through without modification. • Option to enable half-cycle clocking for double resolution.
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.
Trip-zone (TZ)	• Configure the ePWM module to react to one, all, or none of the trip-zone signals or digital compare events. • Specify the tripping action taken when a fault occurs: – Force EPWMxA and/or EPWMxB high – Force EPWMxA and/or EPWMxB low – Force EPWMxA and/or EPWMxB to a high-impedance state – Configure EPWMxA and/or EPWMxB to ignore any trip condition. • Configure how often the ePWM will react to each trip-zone signal: – One-shot – Cycle-by-cycle • Enable the trip-zone to initiate an interrupt. • Bypass the trip-zone module entirely.
Event-trigger (ET)	• Enable the ePWM events that will trigger an interrupt. • Enable ePWM events that will trigger an ADC start-of-conversion event. • Specify the rate at which events cause triggers (every occurrence or every second or third occurrence) • Poll, set, or clear event flags
Digital-compare (DC)	• Enables trip zone signals to create events and filtered events • Specify event-filtering options to capture TBCTR counter or generate blanking window

Code examples are provided in the remainder of this document that show how to implement various ePWM module configurations. These examples use the constant definitions in the device *EPwm_defines.h* file in the device-specific header file and peripheral examples software package.

19.2.2 Time-Base (TB) Submodule

Each ePWM module has its own time-base submodule that determines all of the event timing for the ePWM module. Built-in synchronization logic allows the time-base of multiple ePWM modules to work together as a single system. Figure 19-3 illustrates the time-base module's place within the ePWM.

Figure 19-3. Time-Base Submodule Block Diagram



19.2.2.1 Purpose of the Time-Base Submodule

You can configure the time-base submodule for the following:

- Specify the ePWM time-base counter (TBCTR) frequency or period to control how often events occur.
- Manage time-base synchronization with other ePWM modules.
- Maintain a phase relationship with other ePWM modules.
- Set the time-base counter to count-up, count-down, or count-up-and-down mode.
- Generate the following events:
  - CTR = PRD: Time-base counter equal to the specified period (TBCTR = TBPRD) .
  - CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000).
- Configure the rate of the time-base clock; a prescaled version of the device peripheral clock domain (VCLK4). This allows the time-base counter to increment/decrement at a slower rate.

19.2.2.2 Controlling and Monitoring the Time-base Submodule

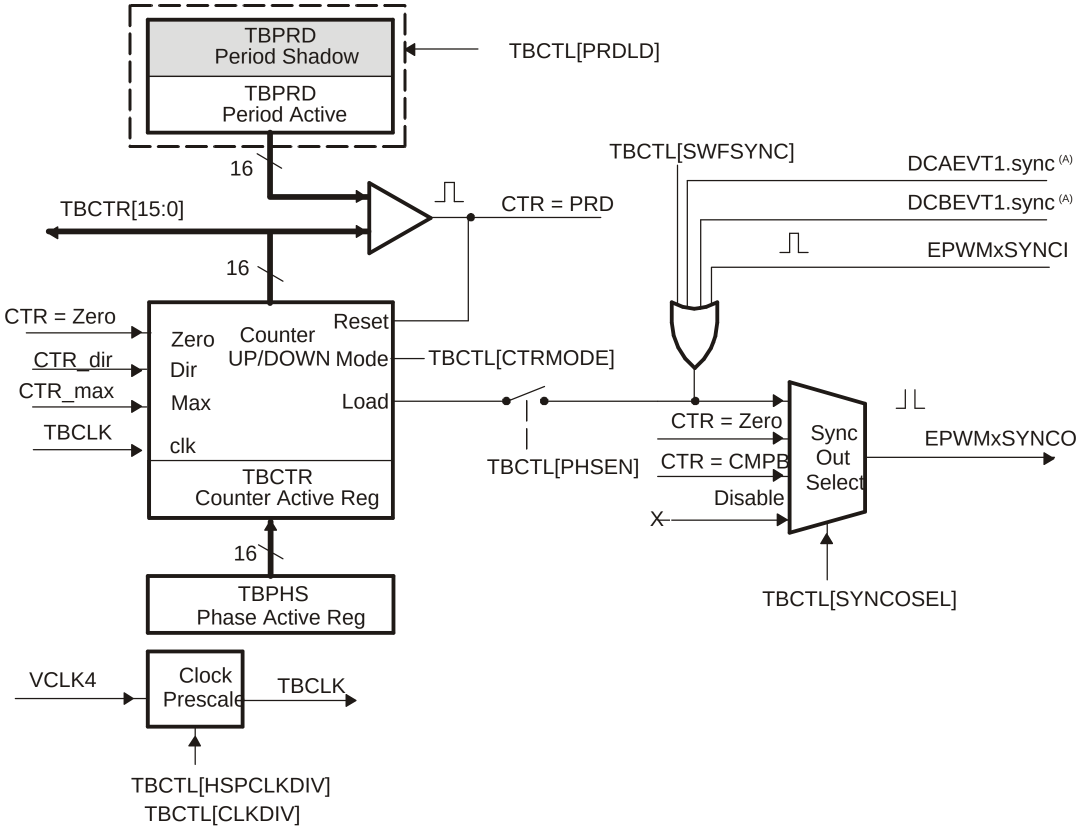
Table 19-3 shows the registers used to control and monitor the time-base submodule.

Table 19-3. Time-Base Submodule Registers

Register Name	Address Offset	Shadowed	Description
TBCTL	0x0000	No	Time-Base Control Register
TBSTS	0x0002	No	Time-Base Status Register
TBPHS	0x0006	No	Time-Base Phase Register
TBCTR	0x0008	No	Time-Base Counter Register
TBPRD	0x000A	Yes	Time-Base Period Register

The block diagram in Figure 19-4 shows the critical signals and registers of the time-base submodule. Table 19-4 provides descriptions of the key signals associated with the time-base submodule.

Figure 19-4. Time-Base Submodule Signals and Registers



A. These signals are generated by the digital compare (DC) submodule.

Table 19-4. 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 or from the N2HET1 module. 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 19.2.2.3.3 for information on the synchronization order of a particular device.
EPWMxSYNCO	Time-base synchronization output. This output pulse is used to synchronize the counter of an ePWM module later in the synchronization chain. The ePWM module generates this signal from one of three event sources: 1. EPWMxSYNCI (Synchronization input pulse) 2. CTR = Zero: The time-base counter equal to zero (TBCTR = 0x0000). 3. CTR = CMPB: The time-base counter equal to the counter-compare B (TBCTR = CMPB) register.
CTR = PRD	Time-base counter equal to the specified period. This signal is generated whenever the counter value is equal to the active period register value. That is when TBCTR = TBPRD.
CTR = Zero	Time-base counter equal to zero This signal is generated whenever the counter value is zero. That is when TBCTR equals 0x0000.
CTR = CMPB	Time-base counter equal to active counter-compare B register (TBCTR = CMPB). This event is generated by the counter-compare submodule and used by the synchronization out logic
CTR_dir	Time-base counter direction. Indicates the current direction of the ePWM's time-base counter. This signal is high when the counter is increasing and low when it is decreasing.
CTR_max	Time-base counter equal max value. (TBCTR = 0xFFFF) Generated event when the TBCTR value reaches its maximum value. This signal is only used only as a status bit
TBCLK	Time-base clock. This is a prescaled version of the system clock (VCLK4) and is used by all submodules within the ePWM. This clock determines the rate at which time-base counter increments or decrements.

19.2.2.3 Calculating PWM Period and Frequency

The frequency of PWM events is controlled by the time-base period (TBPRD) register and the mode of the time-base counter. Figure 19-5 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 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 (VCLK4).

The time-base counter has three modes of operation selected by the time-base control register (TBCTL):

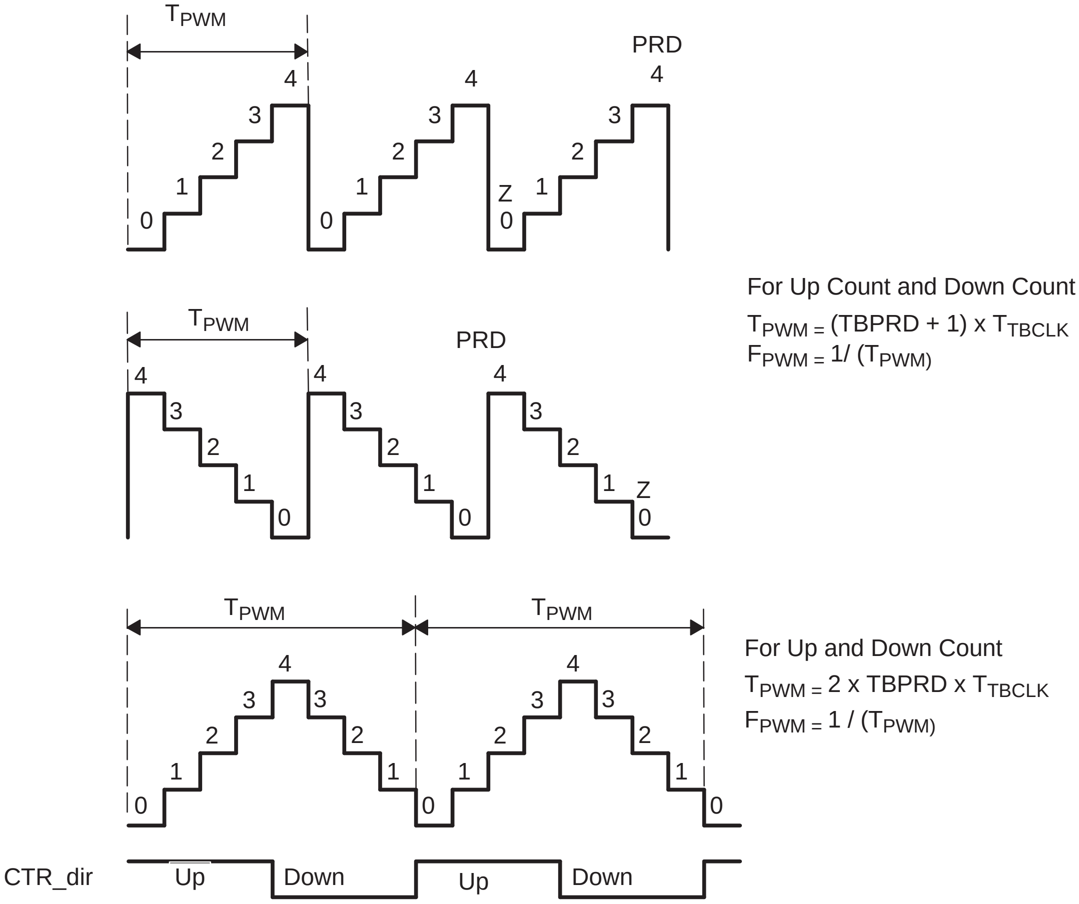
- **Up-Down-Count Mode:**

  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.

Figure 19-5. Time-Base Frequency and Period



19.2.2.3.1 Time-Base Period Shadow Register

The time-base period register (TBPRD) has a shadow register. Shadowing allows the register update to be synchronized with the hardware. The following definitions are used to describe all shadow registers in the ePWM module:

- **Active Register**

  The active register controls the hardware and is responsible for actions that the hardware causes or invokes.

- **Shadow Register**

  The shadow register buffers or provides a temporary holding location for the active register. It has no direct effect on any control hardware. At a strategic point in time the shadow register's content is transferred to the active register. This prevents corruption or spurious operation due to the register being asynchronously modified by software.

The memory address of the shadow period register is the same as the active register. Which register is written to or read from is determined by the TBCTL[PRDLD] bit. This bit enables and disables the TBPRD shadow register as follows:

- **Time-Base Period Shadow Mode:**

  The TBPRD shadow register is enabled when TBCTL[PRDLD] = 0. Reads from and writes to the TBPRD memory address go to the shadow register. The shadow register contents are transferred to the active register (TBPRD (Active) ← TBPRD (shadow)) when the time-base counter equals zero (TBCTR = 0x0000). 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.

19.2.2.3.2 Time-Base Clock Synchronization

Bit 1 of the device-level multiplexing control module (IOMM) register PINMMR37 is defined as the TBCLKSYNC bit. The TBCLKSYNC bit allows users to globally synchronize all enabled ePWM modules to the time-base clock (TBCLK). When set, all enabled ePWM module clocks are started with the first rising edge of TBCLK aligned. For perfectly synchronized TBCLKs, the prescalers for each ePWM module must be set identically.

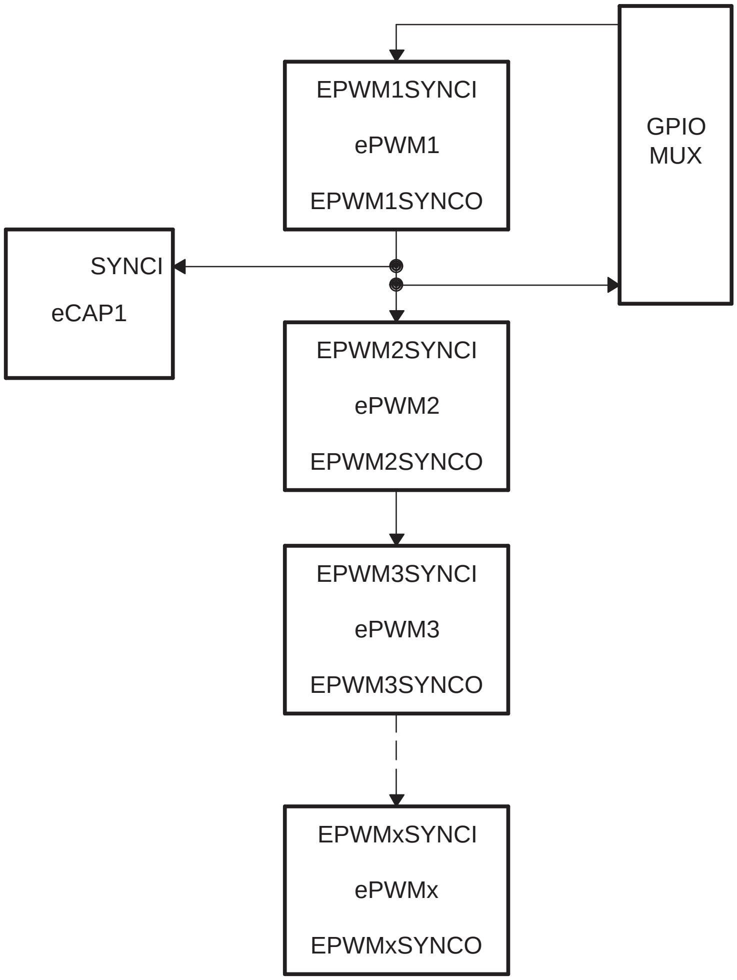
The proper procedure for enabling ePWM clocks is as follows:

1. Enable ePWM module clocks using the IOMM control registers for each ePWM module instance
2. Set TBCLKSYNC = 0. This will stop the time-base clock within any enabled ePWM module.
3. Configure ePWM modules: prescaler values and ePWM modes.
4. Set TBCLKSYNC = 1.

19.2.2.3.3 Time-Base Counter Synchronization

A time-base synchronization scheme connects all of the ePWM modules on a device. Each ePWM module has a synchronization input (EPWMxSYNCI) and a synchronization output (EPWMxSYNCO). The input synchronization for the first instance (ePWM1) comes from an external pin. The synchronization connections for the remaining ePWM modules are shown in Figure 19-6.

Figure 19-6. Time-Base Counter Synchronization Scheme



Each ePWM module can be configured to use or ignore the synchronization input. If the TBCTL[PHSEN] bit is set, then the time-base counter (TBCTR) of the ePWM module will be automatically loaded with the phase register (TBPHS) contents when one of the following conditions occur:

- **EPWMxSYNCI: Synchronization Input Pulse:**

  The value of the phase register is loaded into the counter register when an input synchronization pulse is detected (TBPHS → TBCTR). This operation occurs on the next valid time-base clock (TBCLK) edge.

  The delay from internal master module to slave modules is given by:

  – if (TBCLK = VCLK4): 2 × VCLK4

  – if (TBCLK != VCLK4): 1 TBCLK
- **Software Forced Synchronization Pulse:**

  Writing a 1 to the TBCTL[SWFSYNC] control bit invokes a software forced synchronization. This pulse is ORed with the synchronization input signal, and therefore has the same effect as a pulse on EPWMxSYNCI.
- **Digital Compare Event Synchronization Pulse:**

  DCAEVT1 and DCBEVT1 digital compare events can be configured to generate synchronization pulses which have the same affect as EPWMxSYNCI.

This feature enables the ePWM module to be automatically synchronized to the time base of another ePWM module. Lead or lag phase control can be added to the waveforms generated by different ePWM modules to synchronize them. In up-down-count mode, the TBCTL[PSHDIR] bit configures the direction of the time-base counter immediately after a synchronization event. The new direction is independent of the direction prior to the synchronization event. The PHSDIR bit is ignored in count-up or count-down modes. See Figure 19-7 through Figure 19-10 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.

19.2.2.4 Phase Locking the Time-Base Clocks of Multiple ePWM Modules

The TBCLKSYNC bit can be used to globally synchronize the time-base clocks of all enabled ePWM modules on a device. 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 ePWM module clocks using the IOMM control registers for each ePWM module instance
2. Set TBCLKSYNC = 0. This will stop the time-base clock within any enabled ePWM module.
3. Configure ePWM modules: prescaler values and ePWM modes.
4. Set TBCLKSYNC = 1.

19.2.2.5 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, the following timing diagrams show when events are generated and how the time-base responds to an EPWMxSYNCI signal.

Figure 19-7. Time-Base Up-Count Mode Waveforms

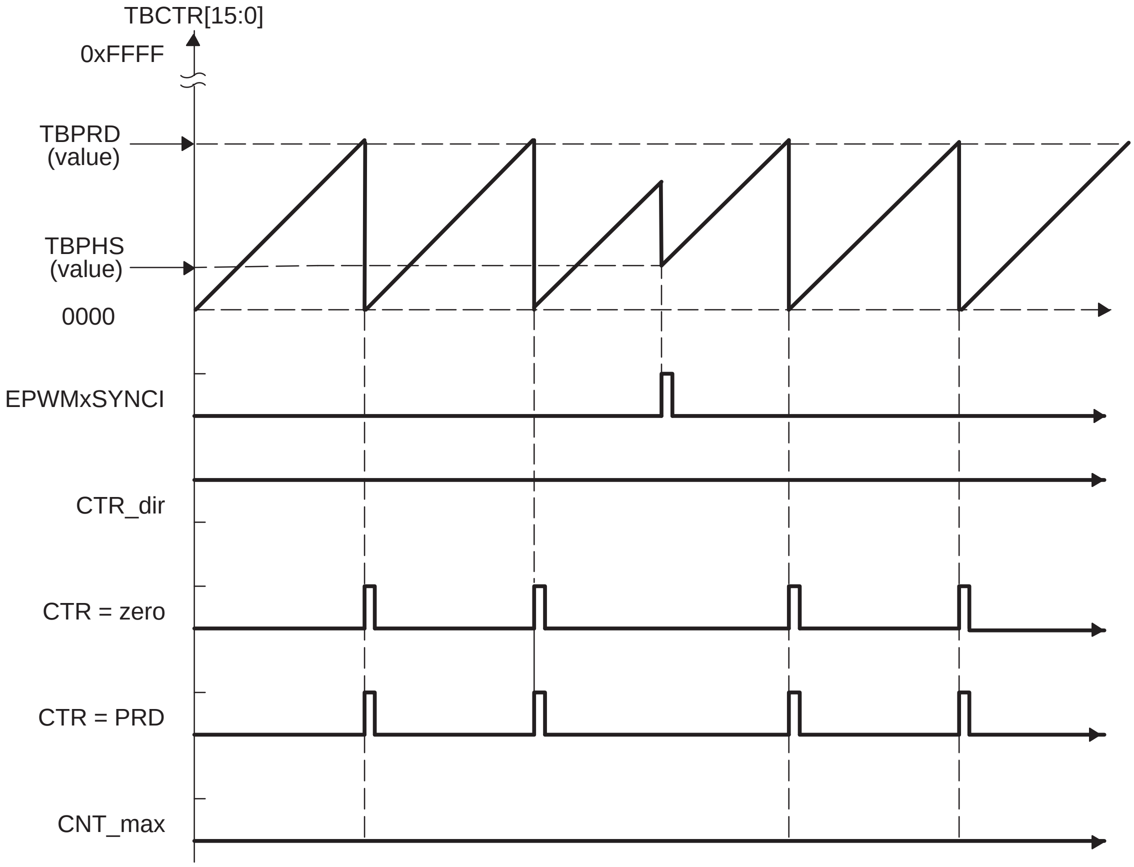


Figure 19-8. Time-Base Down-Count Mode Waveforms

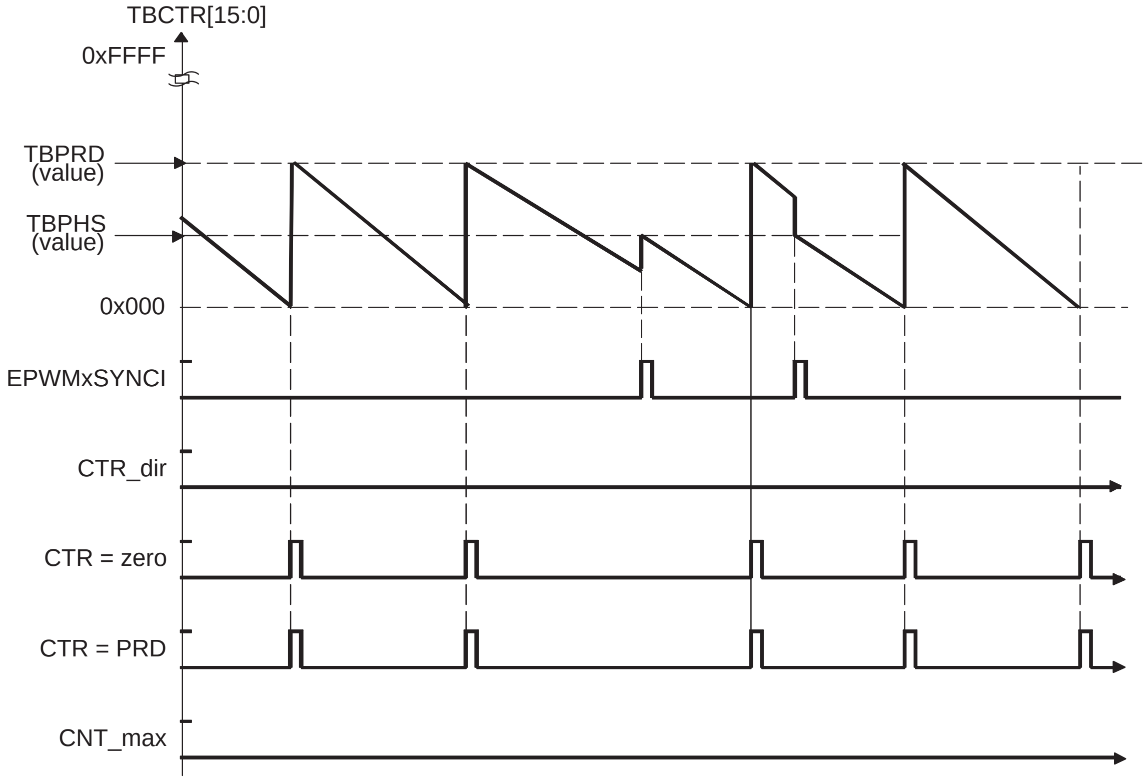


Figure 19-9. Time-Base Up-Down-Count Waveforms, TBCTL[PHSDIR = 0] Count Down On Synchronization Event

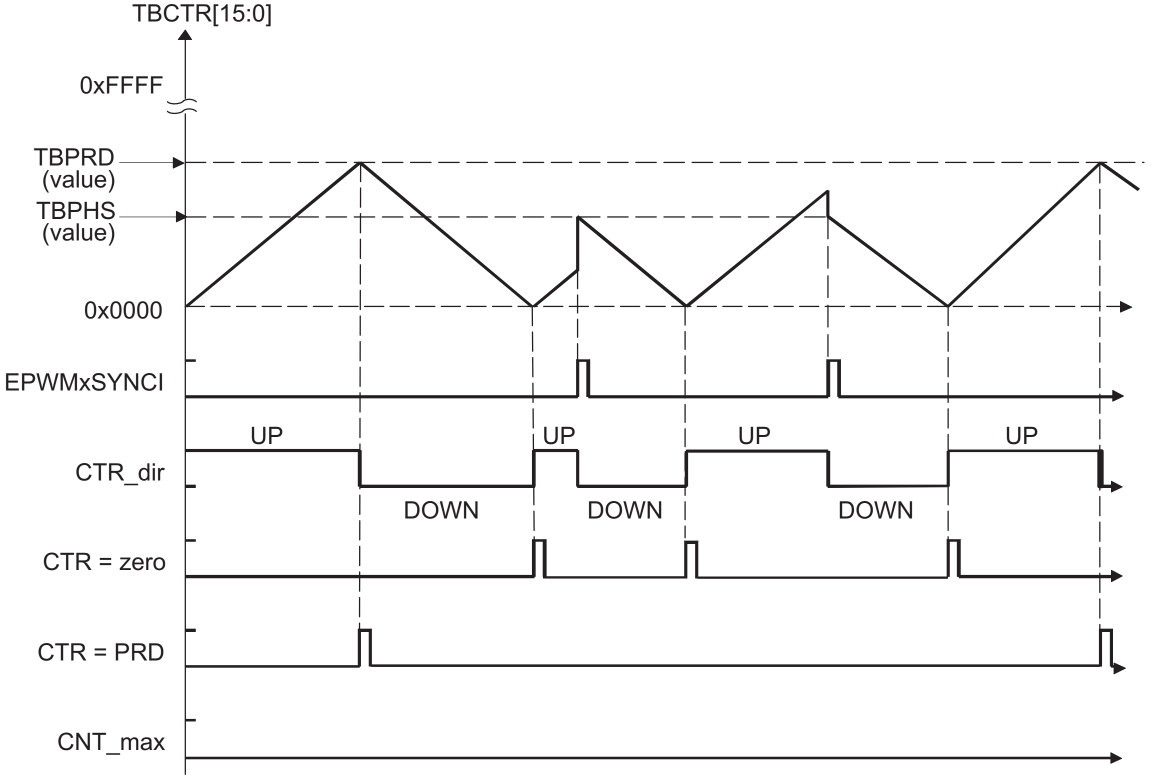
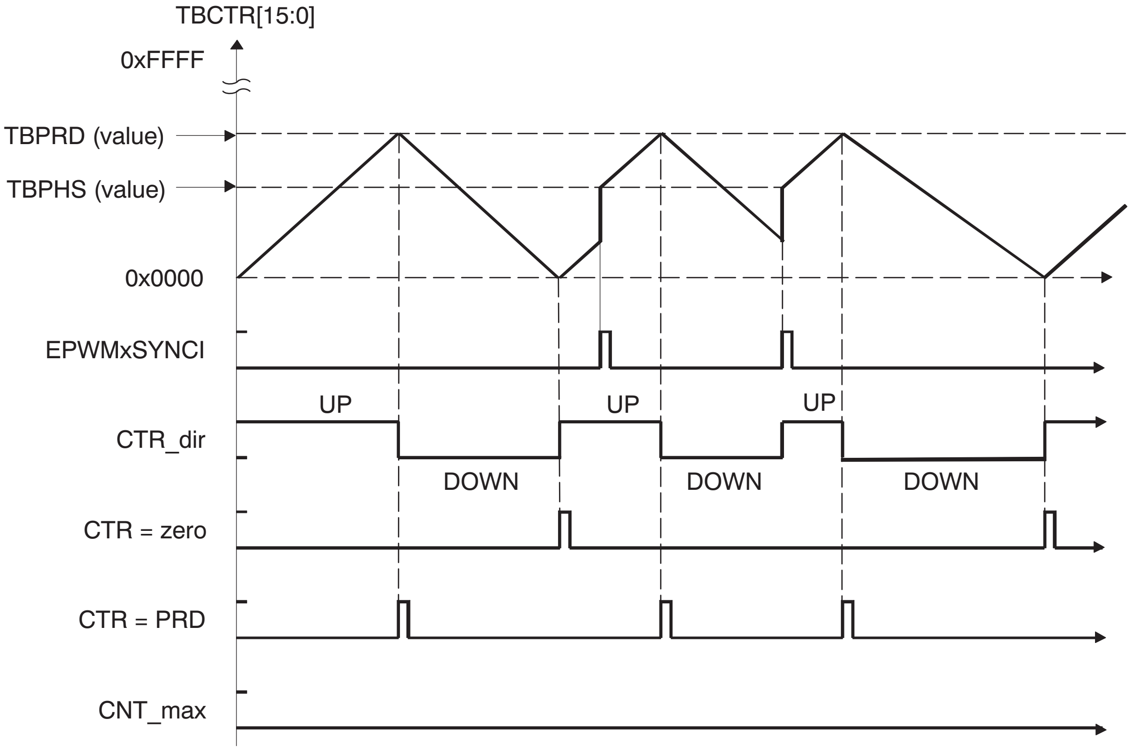


Figure 19-10. Time-Base Up-Down Count Waveforms, TBCTL[PHSDIR = 1] Count Up On Synchronization Event



19.2.3 Counter-Compare (CC) Submodule

Figure 19-11 illustrates the counter-compare submodule within the ePWM.

Figure 19-12 shows the basic structure of the counter-compare submodule.

Figure 19-11. Counter-Compare Submodule

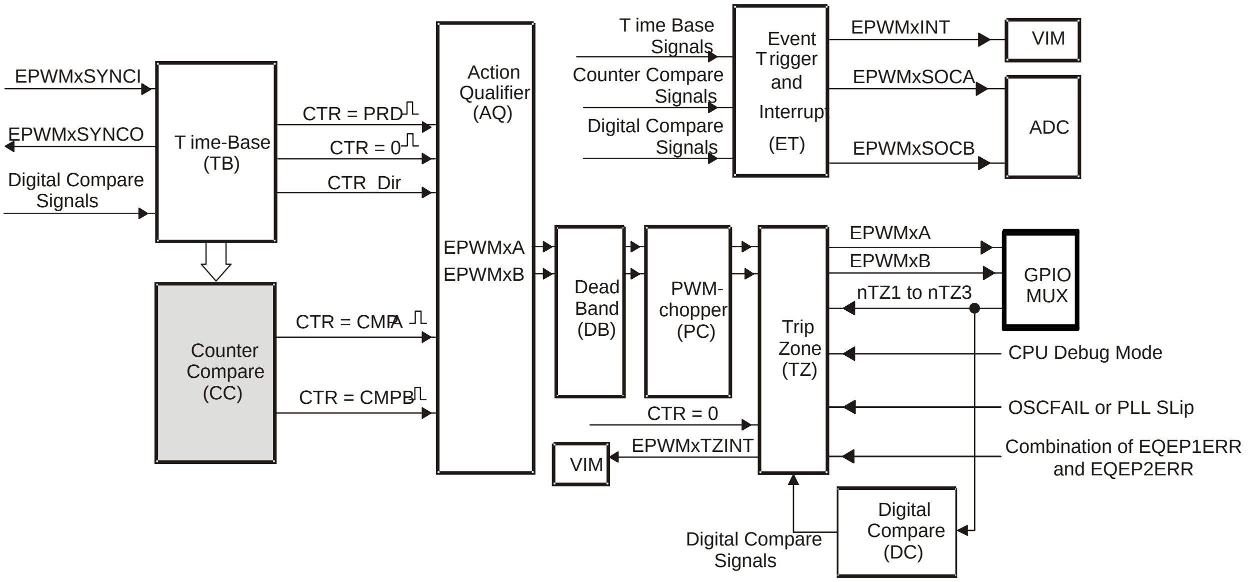
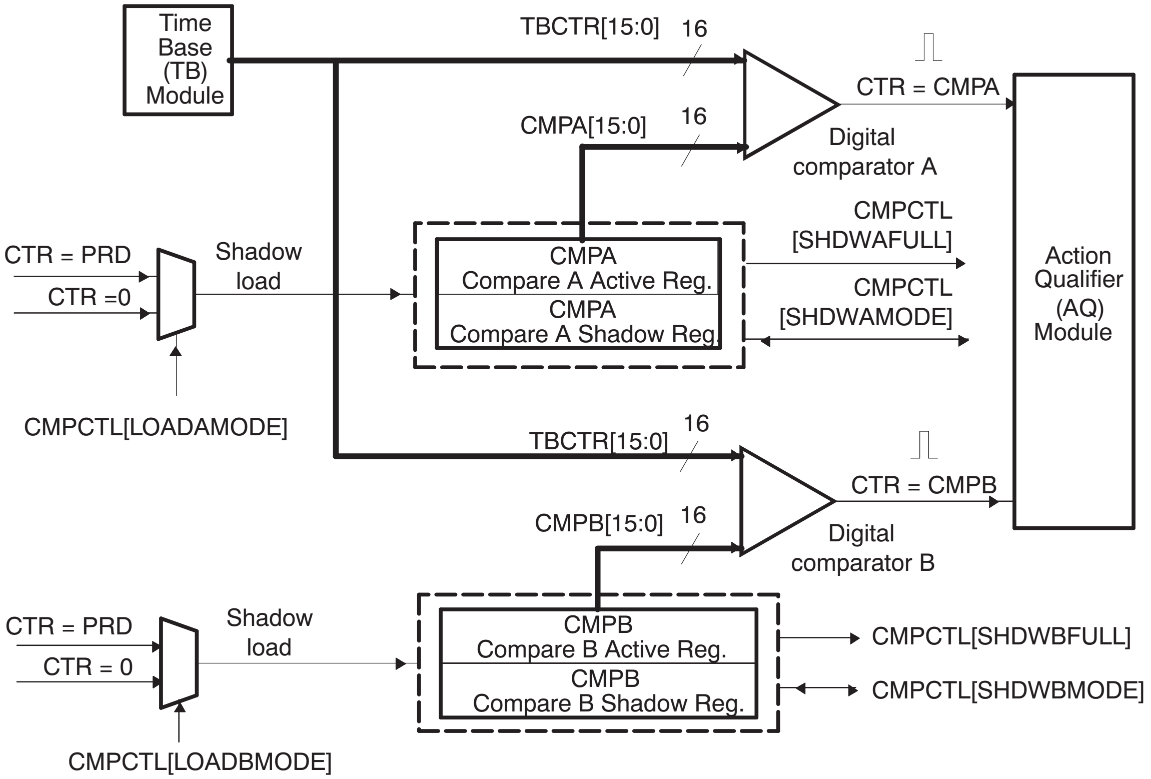


Figure 19-12. Detailed View of the Counter-Compare Submodule



19.2.3.1 Purpose of the Counter-Compare Submodule

The counter-compare submodule takes as input the time-base counter value. This value is continuously compared to the counter-compare A (CMPA) 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:

- Generates events based on programmable time stamps using the CMPA and CMPB registers
  - CTR = CMPA: Time-base counter equals counter-compare A register (TBCTR = CMPA).
  - CTR = CMPB: Time-base counter equals counter-compare B register (TBCTR = CMPB)
- 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

19.2.3.2 Controlling and Monitoring the Counter-Compare Submodule

The counter-compare submodule operation is controlled and monitored by the registers shown in Table 19-5:

Table 19-5. Counter-Compare Submodule Registers

Register Name	Address Offset	Shadowed	Description
CMPCTL	0x000E	No	Counter-Compare Control Register.
CMPA	0x0012	Yes	Counter-Compare A Register
CMPB	0x0014	Yes	Counter-Compare B Register

The key signals associated with the counter-compare submodule are described in Table 19-6.

Table 19-6. Counter-Compare Submodule Key Signals

Signal	Description of Event	Registers Compared
CTR = CMPA	Time-base counter equal to the active counter-compare A value	TBCTR = CMPA
CTR = CMPB	Time-base counter equal to the active counter-compare B value	TBCTR = 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	TBCTR = TBPRD
CTR = ZERO	Time-base counter equal to zero. Used to load active counter-compare A and B registers from the shadow register	TBCTR = 0x0000

19.2.3.3 Operational Highlights for the Counter-Compare Submodule

The counter-compare submodule is responsible for generating two independent compare events based on two compare registers:

1. CTR = CMPA: Time-base counter equal to counter-compare A register (TBCTR = CMPA).
2. CTR = CMPB: Time-base counter equal to counter-compare B register (TBCTR = CMPB).

For up-count or down-count mode, each event occurs only once per cycle. For up-down-count mode each event occurs twice per cycle if the compare value is between 0x0000-TBPRD and once per cycle if the compare value is equal to 0x0000 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 19.2.4.1 for more details.

The counter-compare registers CMPA and CMPB each have an associated shadow register. Shadowing provides a way to keep updates to the registers synchronized with the hardware. When shadowing is used, updates to the active registers only occur at strategic points. This prevents corruption or spurious operation due to the register being asynchronously modified by software. The memory address of the active register and the shadow register is identical. Which register is written to or read from is determined by the CMPCTL[SHDWAMODE] and CMPCTL[SHDWBMODE] bits. These bits enable and disable the CMPA shadow register and CMPB shadow register respectively. The behavior of the two load modes is as described:

Shadow Mode:

The shadow mode for the CMPA is enabled by clearing the CMPCTL[SHDWAMODE] bit and the shadow register for CMPB is enabled by clearing the CMPCTL[SHDWBMODE] bit. Shadow mode is enabled by default for both CMPA and CMPB.

If the shadow register is enabled then the content of the shadow register is transferred to the active register on one of the following events as specified by the CMPCTL[LOADAMODE] and CMPCTL[LOADBMODE] register bits:

- CTR = PRD: Time-base counter equal to the period (TBCTR = TBPRD).
- CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000)
- Both CTR = PRD and CTR = Zero

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 (that is, TBCTL[SHADWAMODE] = 1 or TBCTL[SHADWBMODE] = 1), then a read from or a write to the register will go directly to the active register.

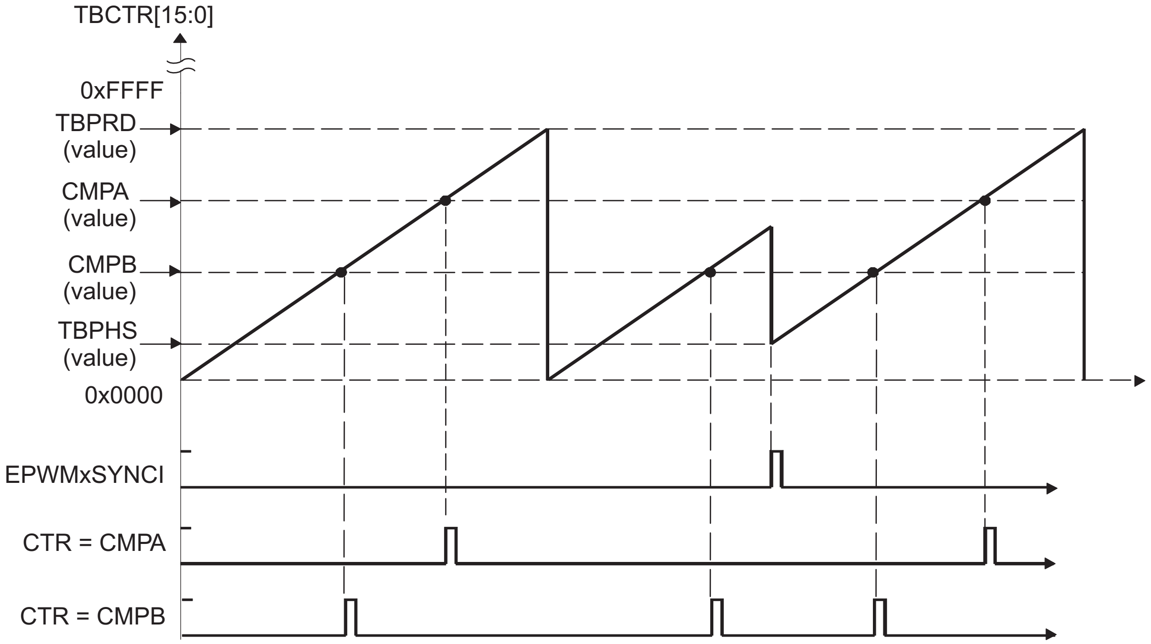
19.2.3.4 Count Mode Timing Waveforms

The counter-compare module can generate compare events in all three count modes:

- Up-count mode: used to generate an asymmetrical PWM waveform.
- Down-count mode: used to generate an asymmetrical PWM waveform.
- Up-down-count mode: used to generate a symmetrical PWM waveform.

To best illustrate the operation of the first three modes, the timing diagrams in Figure 19-13 through Figure 19-16 show when events are generated and how the EPWMxSYNCI signal interacts.

Figure 19-13. Counter-Compare Event Waveforms in Up-Count Mode



NOTE: An EPWMxSYNCI external synchronization event can cause a discontinuity in the TBCTR count sequence. This can lead to a compare event being skipped. This skipping is considered normal operation and must be taken into account.

Figure 19-14. Counter-Compare Events in Down-Count Mode

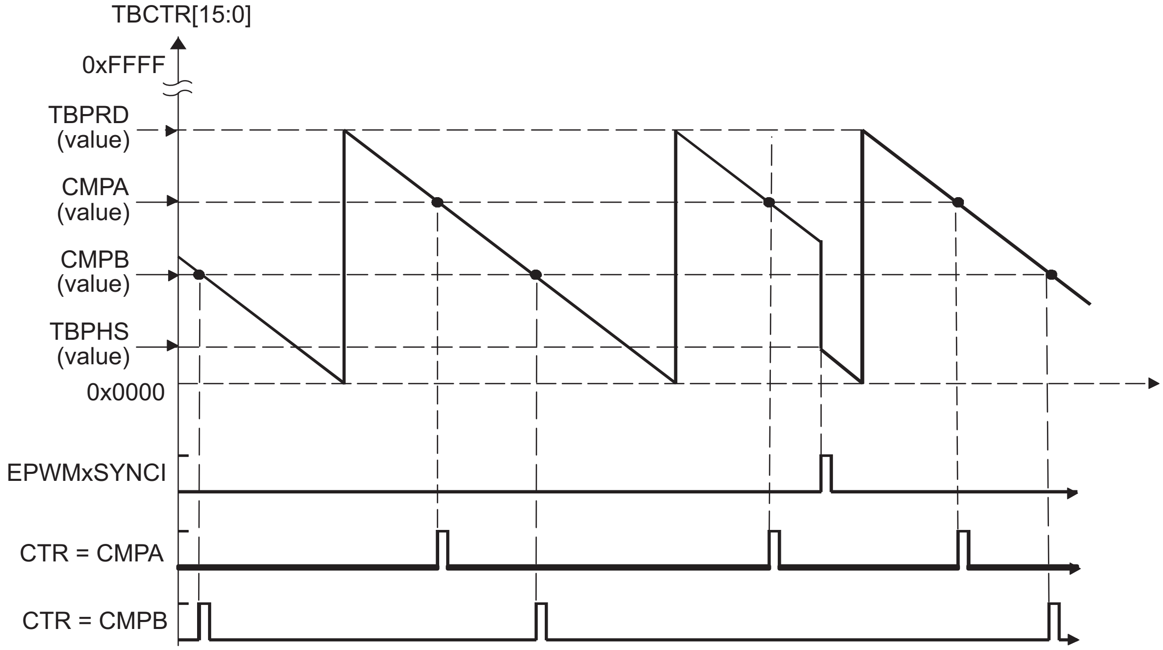


Figure 19-15. Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 0] Count Down On Synchronization Event

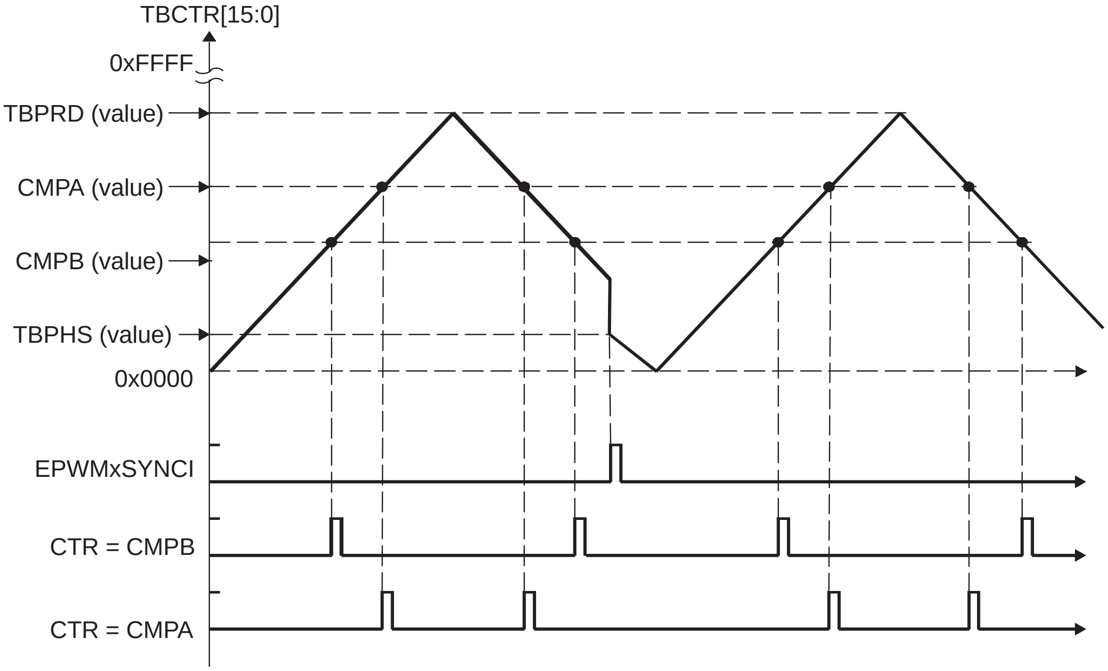
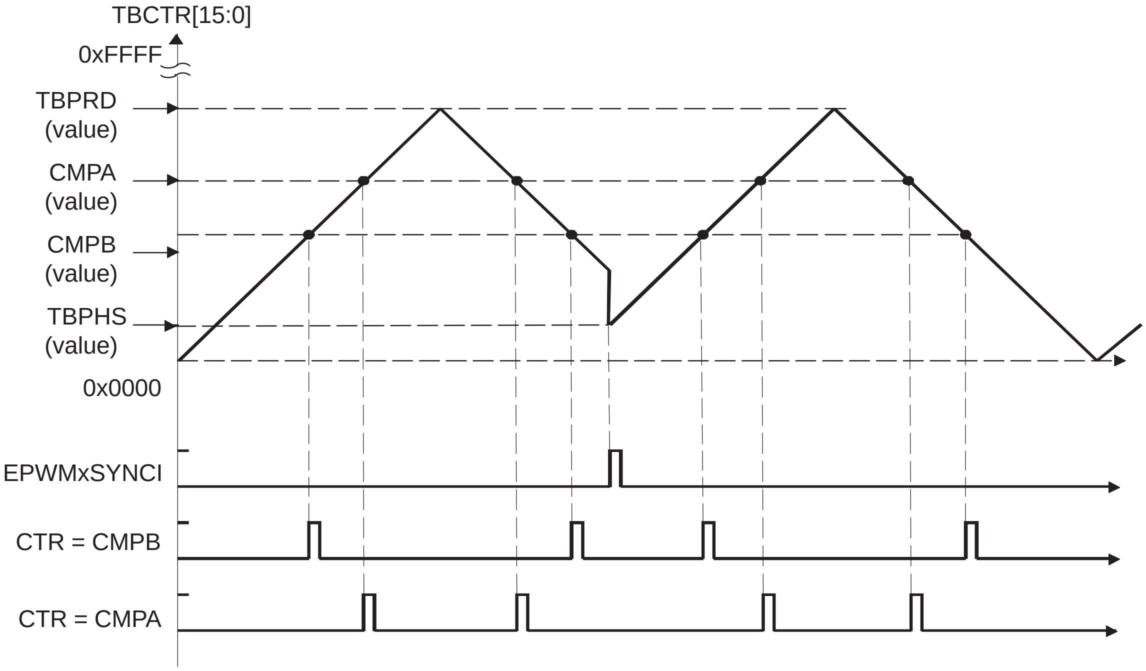


Figure 19-16. Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 1] Count Up On Synchronization Event

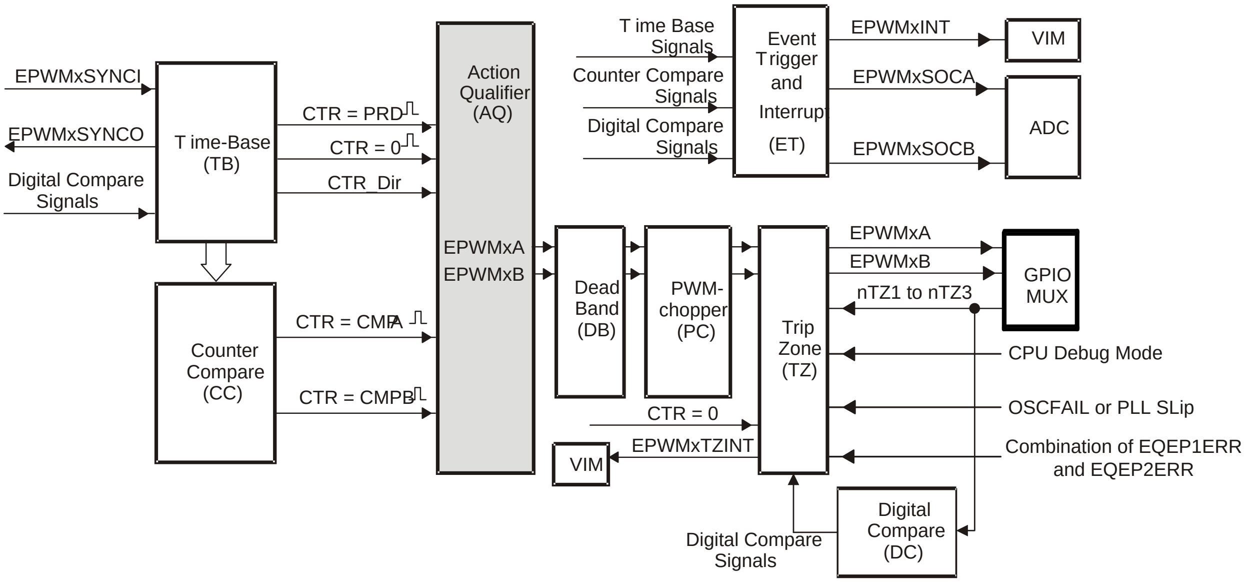


19.2.4 Action-Qualifier (AQ) Submodule

Figure 19-17 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 19-17. Action-Qualifier Submodule



19.2.4.1 Purpose of the Action-Qualifier Submodule

The action-qualifier submodule is responsible for the following:

- Qualifying and generating actions (set, clear, toggle) based on the following events:
  - CTR = PRD: Time-base counter equal to the period (TBCTR = TBPRD)
  - CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000)
  - CTR = CMPA: Time-base counter equal to the counter-compare A register (TBCTR = CMPA)
  - CTR = CMPB: Time-base counter equal to the counter-compare B register (TBCTR = CMPB)
- Managing priority when these events occur concurrently
- Providing independent control of events when the time-base counter is increasing and when it is decreasing

19.2.4.2 Action-Qualifier Submodule Control and Status Register Definitions

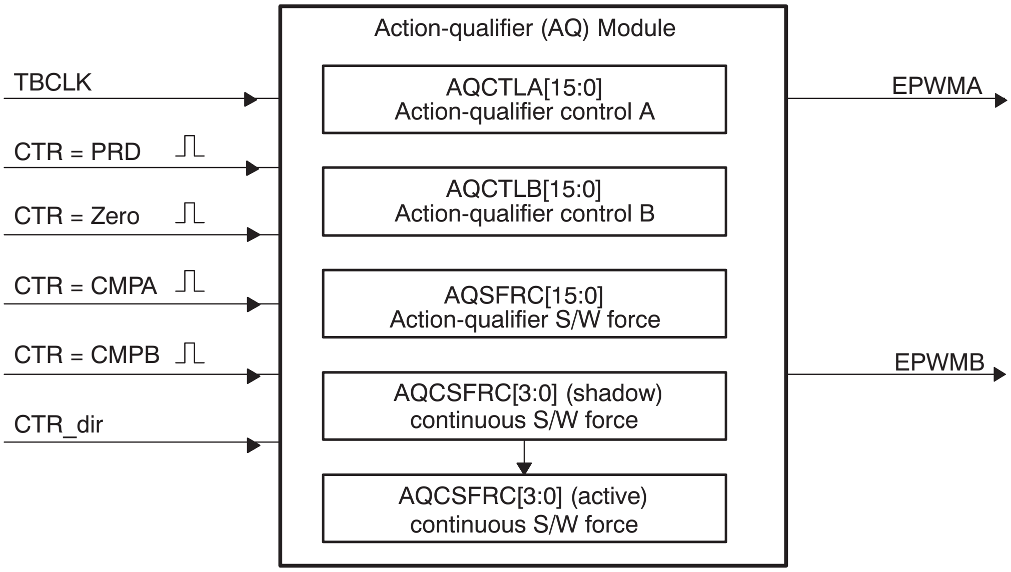
The action-qualifier submodule operation is controlled and monitored via the registers in Table 19-7.

Table 19-7. Action-Qualifier Submodule Registers

Register Name	Address Offset	Shadowed	Description
AQCTLA	0x0016	No	Action-Qualifier Control Register For Output A (EPWMxA)
AQCTLB	0x0018	No	Action-Qualifier Control Register For Output B (EPWMxB)
AQSFRC	0x001A	No	Action-Qualifier Software Force Register
AQCSFRC	0x001C	Yes	Action-Qualifier Continuous Software Force

The action-qualifier submodule is based on event-driven logic. It can be thought of as a programmable cross switch with events at the input and actions at the output, all of which are software controlled via the set of registers shown in Table 19-7.

Figure 19-18. Action-Qualifier Submodule Inputs and Outputs



For convenience, the possible input events are summarized again in Table 19-8.

Table 19-8. Action-Qualifier Submodule Possible Input Events

Signal	Description	Registers Compared
CTR = PRD	Time-base counter equal to the period value	TBCTR = TBPRD
CTR = Zero	Time-base counter equal to zero	TBCTR = 0x0000
CTR = CMPA	Time-base counter equal to the counter-compare A	TBCTR = CMPA
CTR = CMPB	Time-base counter equal to the counter-compare B	TBCTR = CMPB
Software forced event	Asynchronous event initiated by software	

The software forced action is a useful asynchronous event. This control is handled by registers AQSFRC and AQCSFRC.

The action-qualifier submodule controls how the two outputs EPWMxA and EPWMxB behave when a particular event occurs. The event inputs to the action-qualifier submodule are further qualified by the counter direction (up or down). This allows for independent action on outputs on both the count-up and count-down phases.

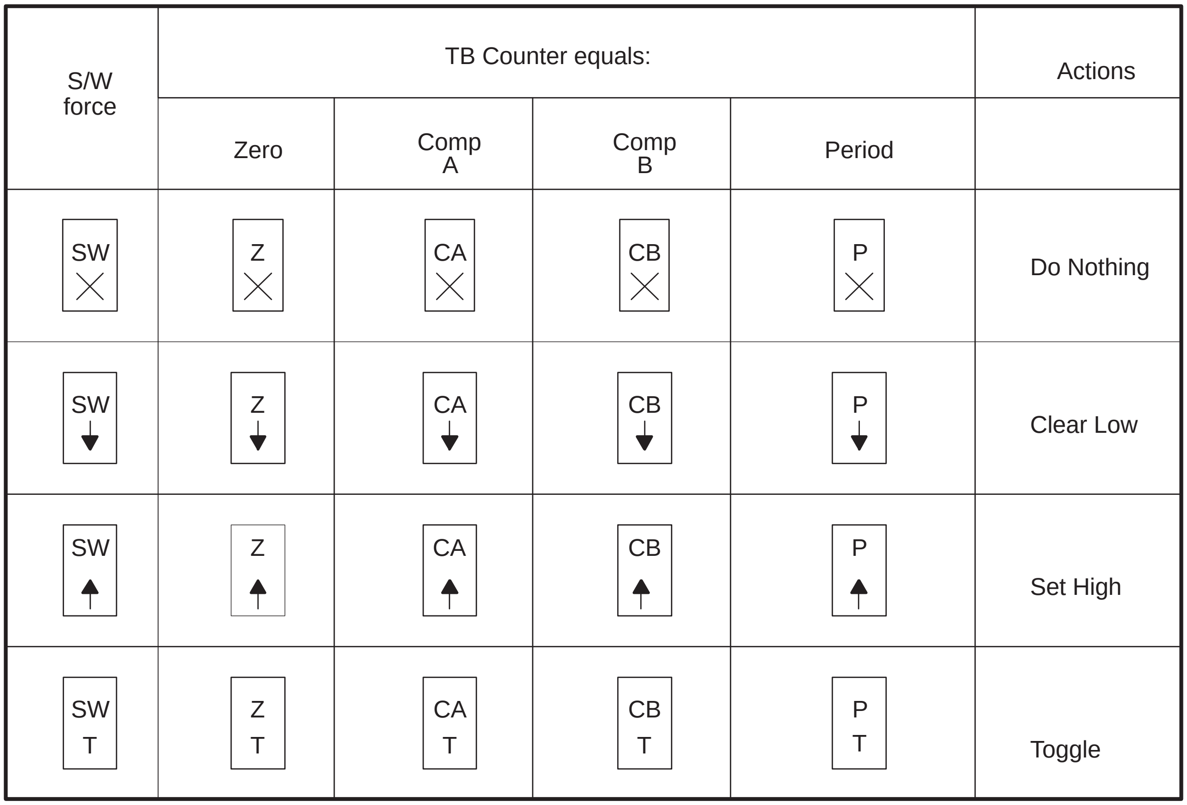
The possible actions imposed on outputs EPWMxA and EPWMxB are:

- **Set High:**
  Set output EPWMxA or EPWMxB to a high level.
- **Clear Low:**
  Set output EPWMxA or EPWMxB to a low level.
- **Toggle:**
  If EPWMxA or EPWMxB is currently pulled high, then pull the output low. If EPWMxA or EPWMxB is currently pulled low, then pull the output high.
- **Do Nothing:**
  Keep outputs EPWMxA and EPWMxB at same level as currently set. Although the "Do Nothing" option prevents an event from causing an action on the EPWMxA and EPWMxB outputs, this event can still trigger interrupts and ADC start of conversion. See the Event-trigger Submodule description in Section 19.2.8 for details.

Actions are specified independently for either output (EPWMxA or EPWMxB). Any or all events can be configured to generate actions on a given output. For example, both CTR = CMPA and CTR = CMPB can operate on output EPWMxA. All qualifier actions are configured via the control registers found at the end of this section.

For clarity, the drawings in this document use a set of symbolic actions. These symbols are summarized in Figure 19-19. 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 19-19. Possible Action-Qualifier Actions for EPWMxA and EPWMxB Outputs



S/W force	TB Counter equals:				Actions
	Zero	Comp A	Comp B	Period	
SW ✕	Z ✕	CA ✕	CB ✕	P ✕	Do Nothing
SW ↓	Z ↓	CA ↓	CB ↓	P ↓	Clear Low
SW ↑	Z ↑	CA ↑	CB ↑	P ↑	Set High
SW T	Z T	CA T	CB T	P T	Toggle

19.2.4.3 Action-Qualifier Event Priority

It is possible for the ePWM action qualifier to receive more than one event at the same time. In this case events are assigned a priority by the hardware. The general rule is events occurring later in time have a higher priority and software forced events always have the highest priority. The event priority levels for up-down-count mode are shown in Table 19-9. A priority level of 1 is the highest priority and level 7 is the lowest. The priority changes slightly depending on the direction of TBCTR.

Table 19-9. Action-Qualifier Event Priority for Up-Down-Count Mode

Priority Level	Event If TBCTR is Incrementing TBCTR = Zero up to TBCTR = TBPRD	Event If TBCTR is Decrementing TBCTR = TBPRD down to TBCTR = 1
1 (Highest)	Software forced event	Software forced event
2	Counter equals CMPB on up-count (CBU)	Counter equals CMPB on down-count (CBD)
3	Counter equals CMPA on up-count (CAU)	Counter equals CMPA on down-count (CAD)
4	Counter equals zero	Counter equals period (TBPRD)
5	Counter equals CMPB on down-count (CBD)	Counter equals CMPB on up-count (CBU)
6 (Lowest)	Counter equals CMPA on down-count (CAD)	Counter equals CMPA on up-count (CBU)

Table 19-10 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 19-10. 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 19-11 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 19-11. 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)

It is possible to set the compare value greater than the period. In this case the action will take place as shown in Table 19-12.

Table 19-12. Behavior if CMPA/CMPB is Greater than the Period

Counter Mode	Compare on Up-Count Event CAD/CBD	Compare on Down-Count Event CAD/CBD
Up-Count Mode	If CMPA/CMPB ≤ TBPRD period, then the event occurs on a compare match (TBCTR=CMPA or CMPB). If CMPA/CMPB > TBPRD, then the event will not occur.	Never occurs.
Down-Count Mode	Never occurs.	If CMPA/CMPB < TBPRD, the event will occur on a compare match (TBCTR=CMPA or CMPB). If CMPA/CMPB ≥ TBPRD, the event will occur on a period match (TBCTR=TBPRD).
Up-Down-Count Mode	If CMPA/CMPB < TBPRD and the counter is incrementing, the event occurs on a compare match (TBCTR=CMPA or CMPB). If CMPA/CMPB is ≥ TBPRD, the event will occur on a period match (TBCTR = TBPRD).	If CMPA/CMPB < TBPRD and the counter is decrementing, the event occurs on a compare match (TBCTR=CMPA or CMPB). If CMPA/CMPB ≥ TBPRD, the event occurs on a period match (TBCTR=TBPRD).

19.2.4.4 Waveforms for Common Configurations

NOTE: The waveforms in this document show the ePWMs behavior for a static compare register value. In a running system, the active compare registers (CMPA and CMPB) are typically updated from their respective shadow registers once every period. The user specifies when the update will take place; either when the time-base counter reaches zero or when the time-base counter reaches period. There are some cases when the action based on the new value can be delayed by one period or the action based on the old value can take effect for an extra period. Some PWM configurations avoid this situation. These include, but are not limited to, the following:

Use up-down-count mode to generate a symmetric PWM:

- If you load CMPA/CMPB on zero, then use CMPA/CMPB values greater than or equal to 1.
- If you load CMPA/CMPB on period, then use CMPA/CMPB values less than or equal to TBPRD - 1.

  This means there will always be a pulse of at least one TBCLK cycle in a PWM period which, when very short, tend to be ignored by the system.

Use up-down-count mode to generate an asymmetric PWM:

- To achieve 50%-0% asymmetric PWM use the following configuration: Load CMPA/CMPB on period and use the period action to clear the PWM and a compare-up action to set the PWM. Modulate the compare value from 0 to TBPRD to achieve 50%-0% PWM duty.

When using up-count mode to generate an asymmetric PWM:

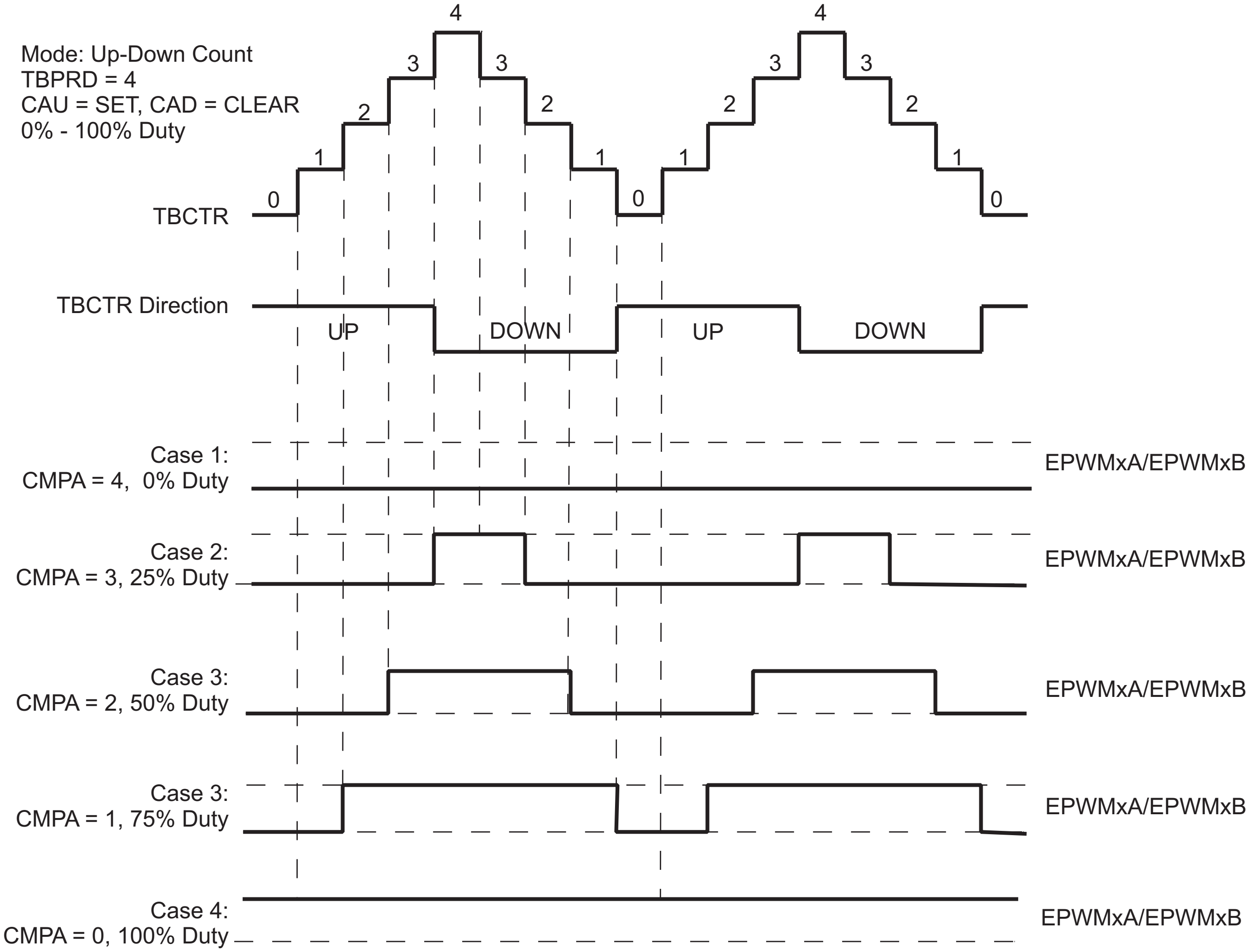
- To achieve 0-100% asymmetric PWM use the following configuration: Load CMPA/CMPB on TBPRD. Use the Zero action to set the PWM and a compare-up action to clear the PWM. Modulate the compare value from 0 to TBPRD + 1 to achieve 0-100% PWM duty.

See the *Using Enhanced Pulse Width Modulator (ePWM) Module for 0-100% Duty Cycle Control* Application Report (SPRAAI1)

Figure 19-20 shows how a symmetric PWM waveform can be generated using the up-down-count mode of the TBCTR. In this mode 0%-100% DC modulation is achieved by using equal compare matches on the up count and down count portions of the waveform. In the example shown, CMPA is used to make the comparison. When the counter is incrementing the CMPA match will pull the PWM output high. Likewise, when the counter is decrementing the compare match will pull the PWM signal low. When CMPA = 0, the PWM signal is low for the entire period giving the 0% duty waveform. When CMPA = TBPRD, the PWM signal is high achieving 100% duty.

When using this configuration in practice, if you load CMPA/CMPB on zero, then use CMPA/CMPB values greater than or equal to 1. If you load CMPA/CMPB on period, then use CMPA/CMPB values less than or equal to TBPRD - 1. This means there will always be a pulse of at least one TBCLK cycle in a PWM period which, when very short, tend to be ignored by the system.

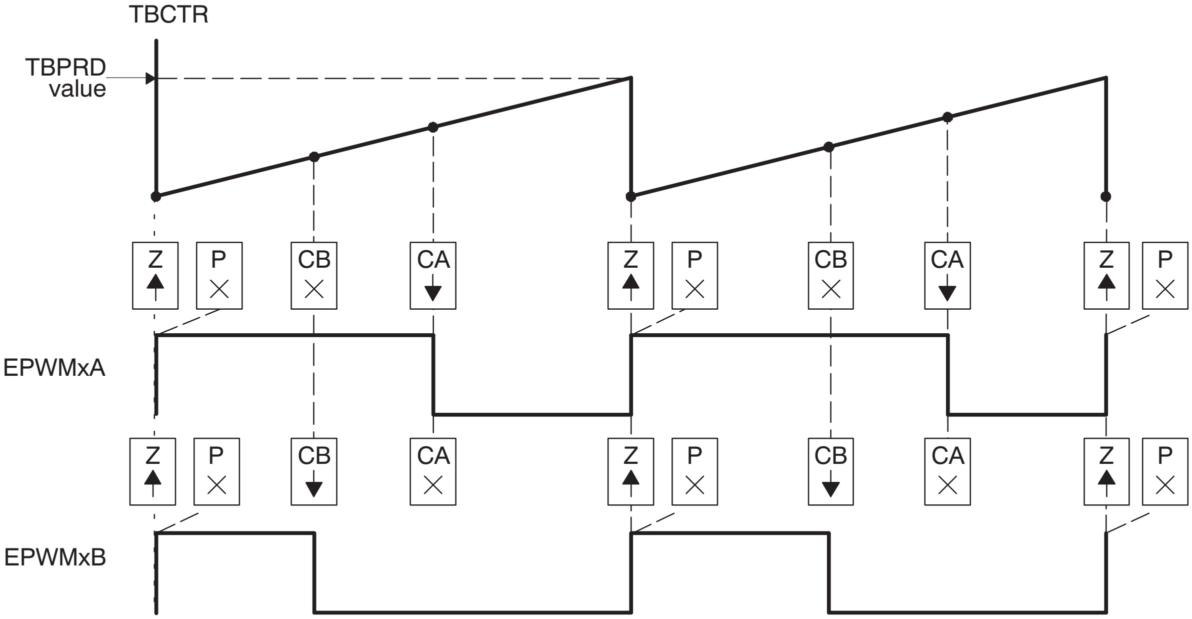
Figure 19-20. Up-Down-Count Mode Symmetrical Waveform



The PWM waveforms in Figure 19-21 through Figure 19-26 show some common action-qualifier configurations. The C-code samples in Example 19-1 through Example 19-6 shows how to configure an ePWM module for each case. Some conventions used in the figures and examples are as follows:

- TBPRD, CMPA, and CMPB refer to the value written in their respective registers. The active register, not the shadow register, is used by the hardware.
- CMPx, refers to either CMPA or CMPB
- EPWMxA and EPWMxB refer to the output signals from ePWMx
- Up-Down means Count-up-and-down mode, Up means up-count mode and Dwn means down-count mode
- Sym = Symmetric, Asym = Asymmetric

Figure 19-21. Up, Single Edge Asymmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB—Active High



A PWM period = (TBPRD + 1) × T_{TBCLK}

B Duty modulation for EPWMxA is set by CMPA, and is active high (that is, high time duty proportional to CMPA).

C Duty modulation for EPWMxB is set by CMPB and is active high (that is, high time duty proportional to CMPB).

D The "Do Nothing" actions (X) are shown for completeness, but will not be shown on subsequent diagrams.

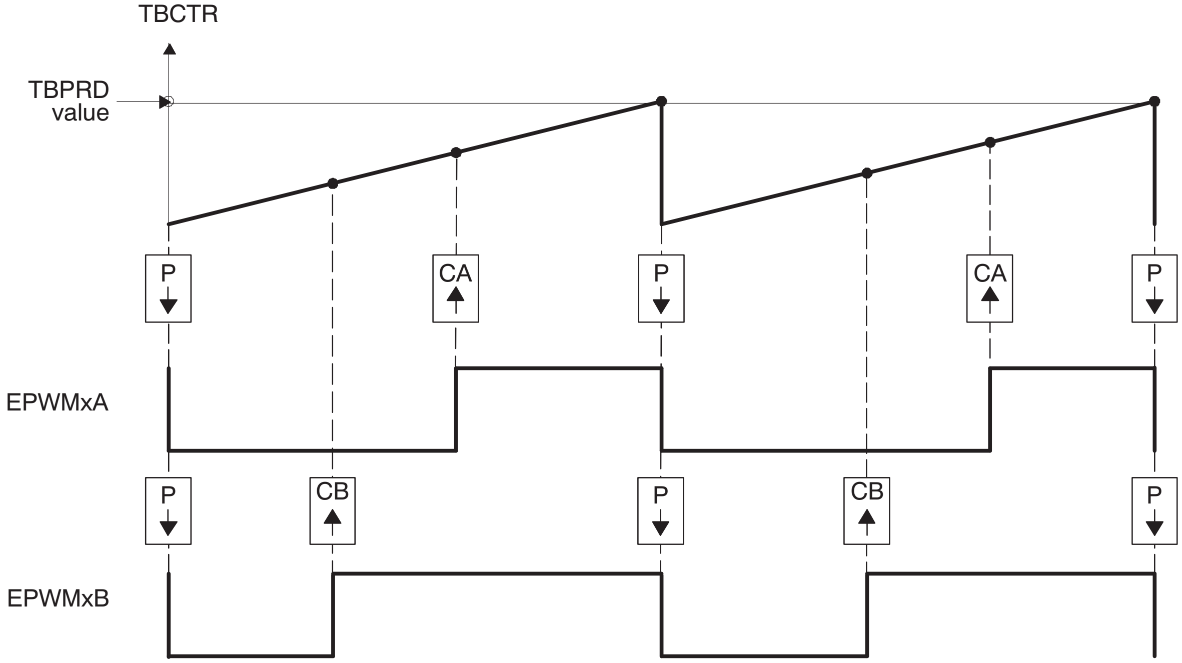
E Actions at zero and period, although appearing to occur concurrently, are actually separated by one TBCLK period. TBCTR wraps from period to 0000.

Example 19-1 contains a code sample showing initialization and run time for the waveforms in Figure 19-21.

Example 19-1. Code Sample for Figure 19-21

```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.TBPRD = 600;                        // Period = 601 TBCLK counts
EPwm1Regs.CMPA.half.CMPA = 350;               // Compare A = 350 TBCLK counts
EPwm1Regs.CMPB = 200;                         // Compare B = 200 TBCLK counts
EPwm1Regs.TBPHS = 0;                          // Set Phase register to zero
EPwm1Regs.TBCTR = 0;                          // clear TB counter
EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;
EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;       // Phase loading disabled
EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;      // TBCLK = SYSCLK
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // load on CTR = Zero
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // load on CTR = Zero
EPwm1Regs.AQCTLA.bit.ZRO = AQ_SET;
EPwm1Regs.AQCTLA.bit.CAU = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.ZRO = AQ_SET;
EPwm1Regs.AQCTLB.bit.CBU = AQ_CLEAR;
//
// Run Time
// = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.CMPA.half.CMPA = Duty1A;            // adjust duty for output EPWM1A
EPwm1Regs.CMPB = Duty1B;                      // adjust duty for output EPWM1B
```

Figure 19-22. Up, Single Edge Asymmetric Waveform With Independent Modulation on EPWMxA and EPWMxB—Active Low



A PWM period = (TBPRD + 1) × T_{TBCLK}

B Duty modulation for EPWMxA is set by CMPA, and is active low (that is, the low time duty is proportional to CMPA).

C Duty modulation for EPWMxB is set by CMPB and is active low (that is, the low time duty is proportional to CMPB).

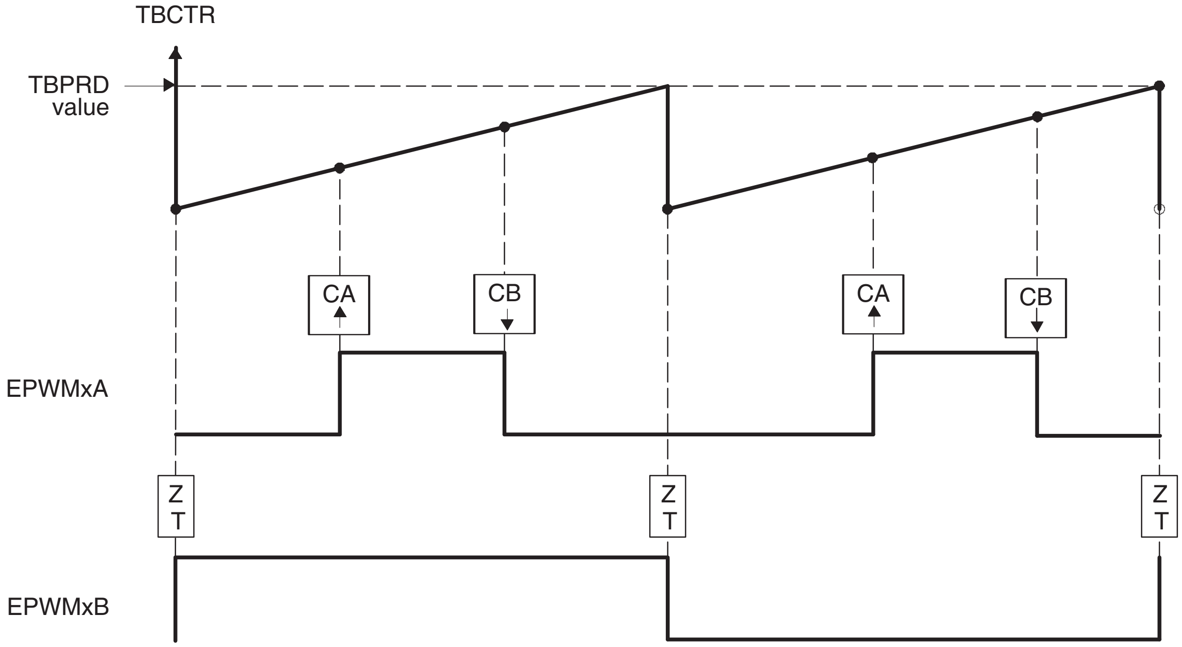
D Actions at zero and period, although appearing to occur concurrently, are actually separated by one TBCLK period. TBCTR wraps from period to 0000.

Example 19-2 contains a code sample showing initialization and run time for the waveforms in Figure 19-22.

Example 19-2. Code Sample for Figure 19-22

```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.TBPRD = 600;                        // Period = 601 TBCLK counts
EPwm1Regs.CMPA.half.CMPA = 350;               // Compare A = 350 TBCLK counts
EPwm1Regs.CMPB = 200;                         // Compare B = 200 TBCLK counts
EPwm1Regs.TBPHS = 0;                          // Set Phase register to zero
EPwm1Regs.TBCTR = 0;                          // clear TB counter
EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;
EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;       // Phase loading disabled
EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;      // TBCLK = VCLK4
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // load on TBCTR = Zero
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // load on TBCTR = Zero
EPwm1Regs.AQCTLA.bit.PRD = AQ_CLEAR;
EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;
EPwm1Regs.AQCTLB.bit.PRD = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.CBU = AQ_SET;
//
// Run Time
// = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.CMPA.half.CMPA = Duty1A;            // adjust duty for output EPWM1A
EPwm1Regs.CMPB = Duty1B;                      // adjust duty for output EPWM1B
```

Figure 19-23. Up-Count, Pulse Placement Asymmetric Waveform With Independent Modulation on EPWMxA



A PWM frequency = 1/((TBPRD + 1) × T_{TBCLK})

B Pulse can be placed anywhere within the PWM cycle (0000 - TBPRD)

C High time duty proportional to (CMPB - CMPA)

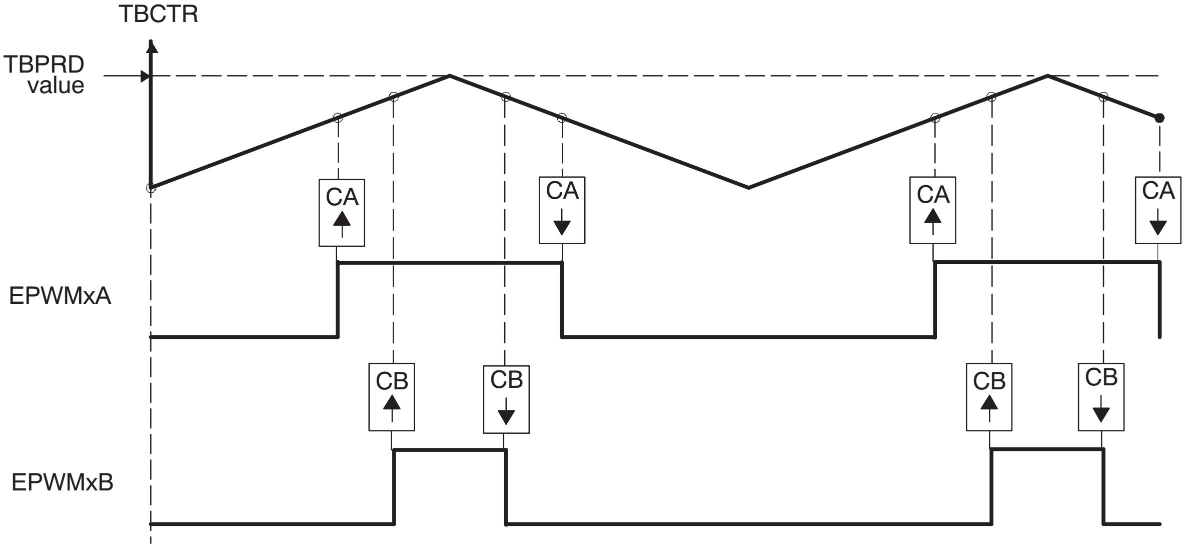
D EPWMxB can be used to generate a 50% duty square wave with frequency = 1/2 × ((TBPRD + 1) × TBCLK)

Example 19-3 contains a code sample showing initialization and run time for the waveforms in Figure 19-23.

Example 19-3. Code Sample for Figure 19-23

```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.TBPRD = 600;                        // Period = 601 TBCLK counts
EPwm1Regs.CMPA.half.CMPA = 200;               // Compare A = 200 TBCLK counts
EPwm1Regs.CMPB = 400;                         // Compare B = 400 TBCLK counts
EPwm1Regs.TBPHS = 0;                          // Set Phase register to zero
EPwm1Regs.TBCTR = 0;                          // clear TB counter
EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;
EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;       // Phase loading disabled
EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;      // TBCLK = VCLK4
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // load on TBCTR = Zero
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // load on TBCTR = Zero
EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;
EPwm1Regs.AQCTLA.bit.CBU = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.ZRO = AQ_TOGGLE;
//
// Run Time
// = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.CMPA.half.CMPA = EdgePosA;         // adjust duty for output EPWM1A only
EPwm1Regs.CMPB = EdgePosB;
```

Figure 19-24. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Active Low



A PWM period = 2 x TBPRD × T_{TBCLK}

B Duty modulation for EPWMxA is set by CMPA, and is active low (that is, the low time duty is proportional to CMPA).

C Duty modulation for EPWMxB is set by CMPB and is active low (that is, the low time duty is proportional to CMPB).

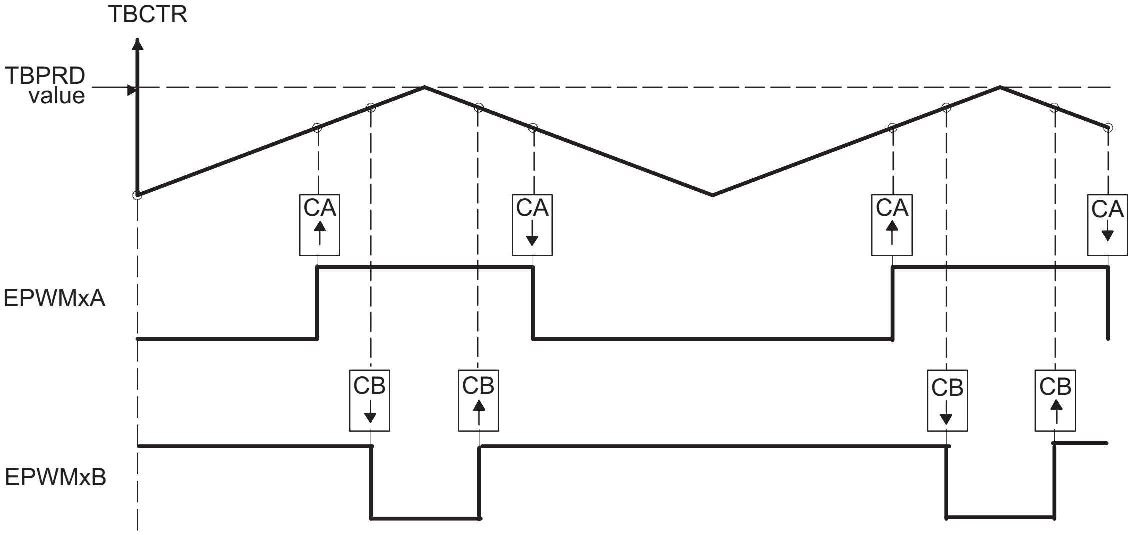
D Outputs EPWMxA and EPWMxB can drive independent power switches

Example 19-4 contains a code sample showing initialization and run time for the waveforms in Figure 19-24.

Example 19-4. Code Sample for Figure 19-24

```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.TBPRD = 600;                           // Period = 2´600 TBCLK counts
EPwm1Regs.CMPA.half.CMPA = 400;                  // Compare A = 400 TBCLK counts
EPwm1Regs.CMPB = 500;                            // Compare B = 500 TBCLK counts
EPwm1Regs.TBPHS = 0;                             // Set Phase register to zero
EPwm1Regs.TBCTR = 0;                             // clear TB counter
EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN; // Symmetric
xEPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;       // Phase loading disabled
xEPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;       // TBCLK = VCLK4
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;  // load on CTR = Zero
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;  // load on CTR = Zero
EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;
EPwm1Regs.AQCTLA.bit.CAD = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.CBU = AQ_SET;
EPwm1Regs.AQCTLB.bit.CBD = AQ_CLEAR;
//
// Run Time
// = = = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.CMPA.half.CMPA = Duty1A;            // adjust duty for output EPWM1A
EPwm1Regs.CMPB = Duty1B;                      // adjust duty for output EPWM1B
```

Figure 19-25. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Complementary



A PWM period = 2 × TBPRD × T_{TBCLK}

B Duty modulation for EPWMxA is set by CMPA, and is active low, that is, low time duty proportional to CMPA

C Duty modulation for EPWMxB is set by CMPB and is active high, that is, high time duty proportional to CMPB

D Outputs EPWMx can drive upper/lower (complementary) power switches

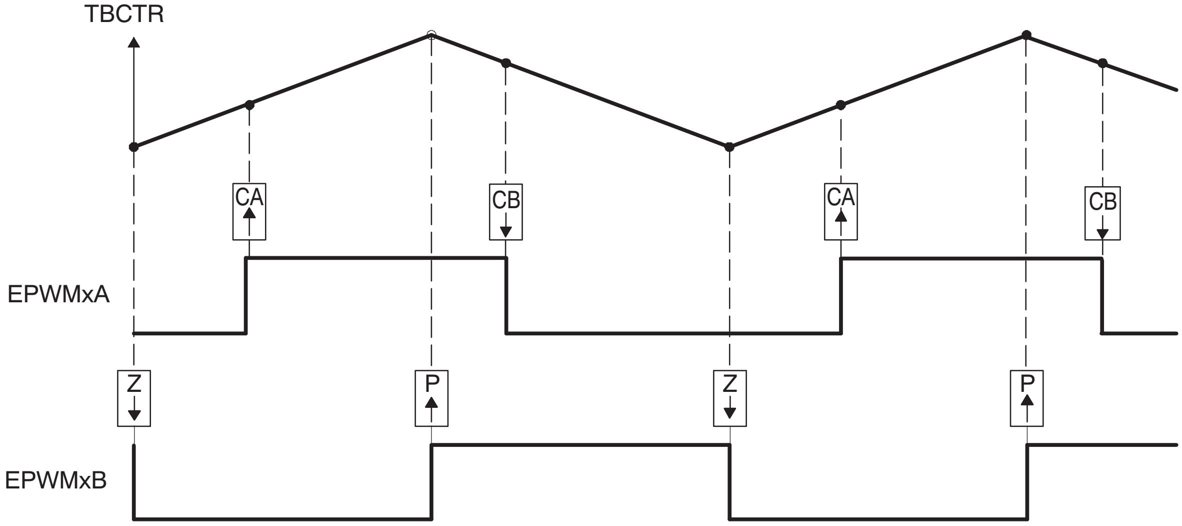
E Dead-band = CMPB - CMPA (fully programmable edge placement by software). Note the dead-band module is also available if the more classical edge delay method is required.

Example 19-5 contains a code sample showing initialization and run time for the waveforms in Figure 19-25.

Example 19-5. Code Sample for Figure 19-25

```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.TBPRD = 600;                          // Period = 2´600 TBCLK counts
EPwm1Regs.CMPA.half.CMPA = 350;                 // Compare A = 350 TBCLK counts
EPwm1Regs.CMPB = 400;                           // Compare B = 400 TBCLK counts
EPwm1Regs.TBPHS = 0;                            // Set Phase register to zero
EPwm1Regs.TBCTR = 0;                            // clear TB counter
EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN; // Symmetric
EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;       // Phase loading disabled
EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;      // TBCLK = VCLK4
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // load on CTR = Zero
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // load on CTR = Zero
EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;
EPwm1Regs.AQCTLA.bit.CAD = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.CBU = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.CBD = AQ_SET;
// Run Time
// = = = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.CMPA.half.CMPA = Duty1A;          // adjust duty for output EPWM1A
EPwm1Regs.CMPB = Duty1B;                    // adjust duty for output EPWM1B
```

Figure 19-26. Up-Down-Count, Dual Edge Asymmetric Waveform, With Independent Modulation on EPWMxA—Active Low



A PWM period = 2 × TBPRD × TBCLK

B Rising edge and falling edge can be asymmetrically positioned within a PWM cycle. This allows for pulse placement techniques.

C Duty modulation for EPWMxA is set by CMPA and CMPB.

D Low time duty for EPWMxA is proportional to (CMPA + CMPB).

E To change this example to active high, CMPA and CMPB actions need to be inverted (that is, Set ! Clear and Clear Set).

F Duty modulation for EPWMxB is fixed at 50% (utilizes spare action resources for EPWMxB)

Example 19-6 contains a code sample showing initialization and run time for the waveforms in Figure 19-26.

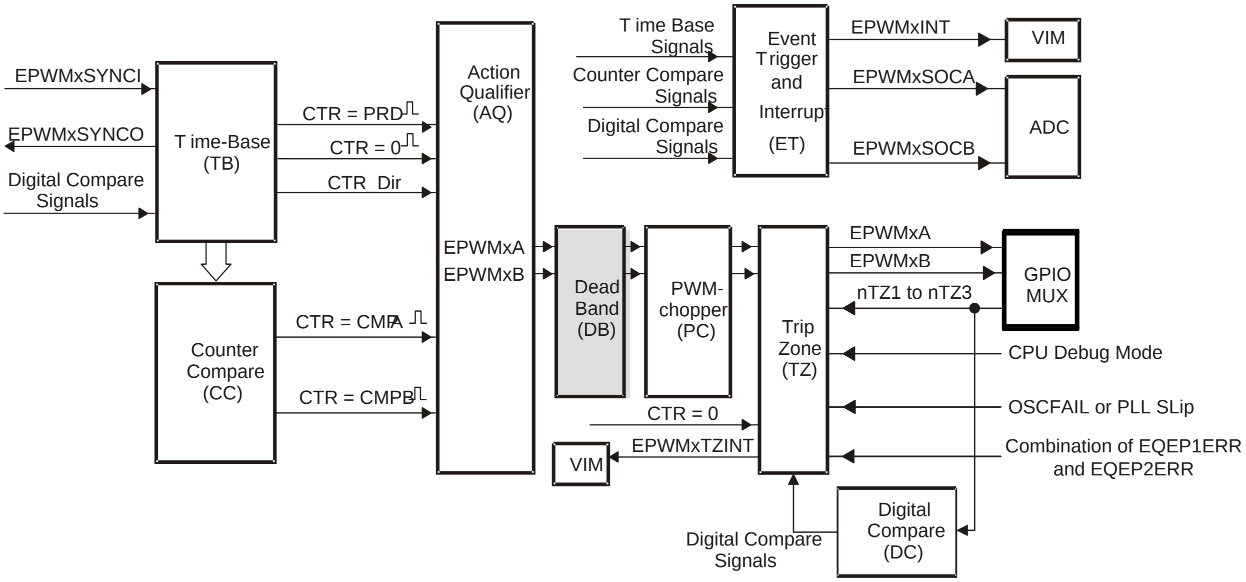
Example 19-6. Code Sample for Figure 19-26

```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.TBPRD = 600;                          //  Period = 2 ´ 600 TBCLK counts
EPwm1Regs.CMPA.half.CMPA = 250;                 //  Compare A = 250 TBCLK counts
EPwm1Regs.CMPB = 450;                           //  Compare B = 450 TBCLK counts
EPwm1Regs.TBPHS = 0;                            //  Set Phase register to zero
EPwm1Regs.TBCTR = 0;                            //  clear TB counter
EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN; //  Symmetric
EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;         //  Phase loading disabled
EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;        //  TBCLK = VCLK4
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;  //  load on CTR = Zero
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;  //  load on CTR = Zero
EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;
EPwm1Regs.AQCTLA.bit.CBD = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.ZRO = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.PRD = AQ_SET;
// Run Time
// = = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.CMPA.half.CMPA = EdgePosA;             // adjust duty for output EPWM1A only
EPwm1Regs.CMPB = EdgePosB;
```

19.2.5 Dead-Band Generator (DB) Submodule

Figure 19-27 illustrates the dead-band submodule within the ePWM module.

Figure 19-27. Dead_Band Submodule



19.2.5.1 Purpose of the Dead-Band Submodule

Section 19.2.4 discussed how it is possible to generate the required dead-band by having full control over edge placement using both the CMPA and CMPB resources of the ePWM module. However, if the more classical edge delay-based dead-band with polarity control is required, then the dead-band submodule described here should be used.

The key functions of the dead-band module are:

- Generating appropriate signal pairs (EPWMxA and EPWMxB) with dead-band relationship from a single EPWMxA input
- Programming signal pairs for:
  - Active high (AH)
  - Active low (AL)
  - Active high complementary (AHC)
  - Active low complementary (ALC)
- Adding programmable delay to rising edges (RED)
- Adding programmable delay to falling edges (FED)
- Can be totally bypassed from the signal path (note dotted lines in diagram)

19.2.5.2 Controlling and Monitoring the Dead-Band Submodule

The dead-band submodule operation is controlled and monitored via the following registers:

Table 19-13. Dead-Band Generator Submodule Registers

Register Name	Address Offset	Shadowed	Description
DBCTL	0x001E	No	Dead-Band Control Register
DBRED	0x0020	No	Dead-Band Rising Edge Delay Count Register
DBFED	0x0022	No	Dead-Band Falling Edge Delay Count Register

19.2.5.3 Operational Highlights for the Dead-Band Submodule

The following sections provide the operational highlights.

The dead-band submodule has two groups of independent selection options as shown in Figure 19-28.

- **Input Source Selection:**

  The input signals to the dead-band module are the EPWMxA and EPWMxB output signals from the action-qualifier. In this section they will be referred to as EPWMxA In and EPWMxB In. Using the DBCTL[IN_MODE) control bits, the signal source for each delay, falling-edge or rising-edge, can be selected:

  – EPWMxA In is the source for both falling-edge and rising-edge delay. This is the default mode.

  – EPWMxA In is the source for falling-edge delay, EPWMxB In is the source for rising-edge delay.

  – EPWMxA In is the source for rising edge delay, EPWMxB In is the source for falling-edge delay.

  – EPWMxB In is the source for both falling-edge and rising-edge delay.

- **Half Cycle Clocking:**

  The dead-band submodule can be clocked using half cycle clocking to double the resolution (that is, counter clocked at 2× TBCLK)

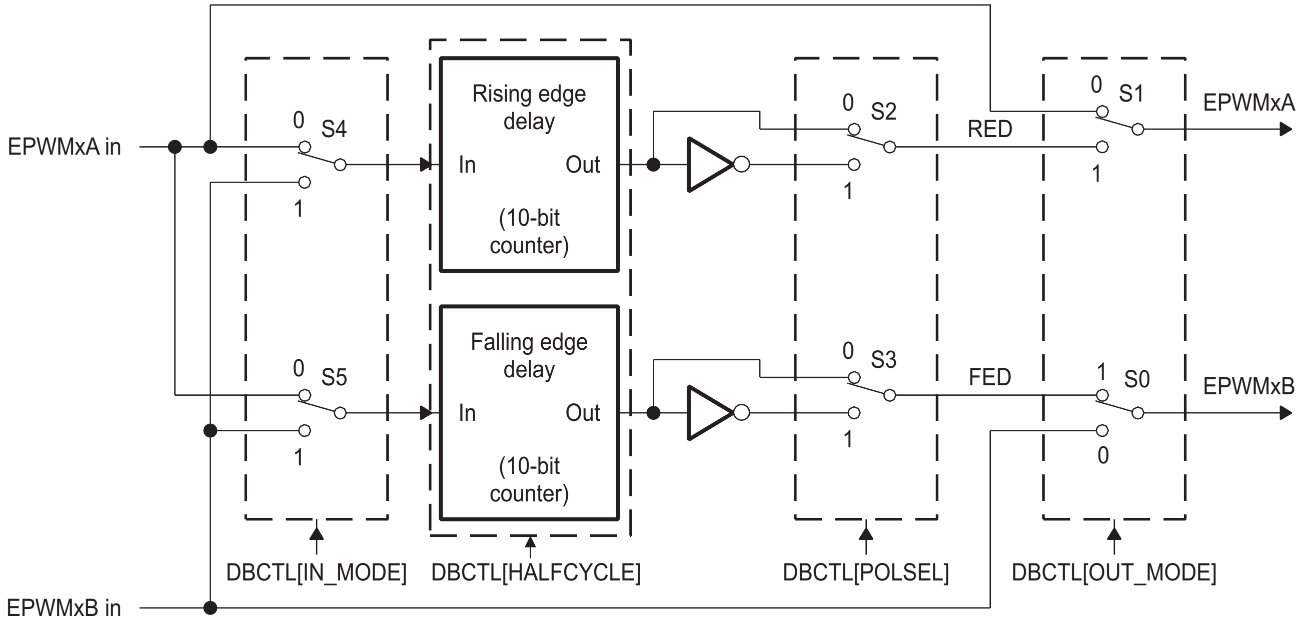
- **Output Mode Control:**

  The output mode is configured by way of the DBCTL[OUT_MODE] bits. These bits determine if the falling-edge delay, rising-edge delay, neither, or both are applied to the input signals.

- **Polarity Control:**

  The polarity control (DBCTL[POLSEL]) allows you to specify whether the rising-edge delayed signal and/or the falling-edge delayed signal is to be inverted before being sent out of the dead-band submodule.

Figure 19-28. Configuration Options for the Dead-Band Submodule



Although all combinations are supported, not all are typical usage modes. Table 19-14 documents some classical dead-band configurations. These modes assume that the DBCTL[IN_MODE] is configured such that EPWMxA In is the source for both falling-edge and rising-edge delay. Enhanced, or non-traditional modes can be achieved by changing the input signal source. The modes shown in Table 19-14 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 19-29. Note that to generate equivalent waveforms to Figure 19-29, 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**

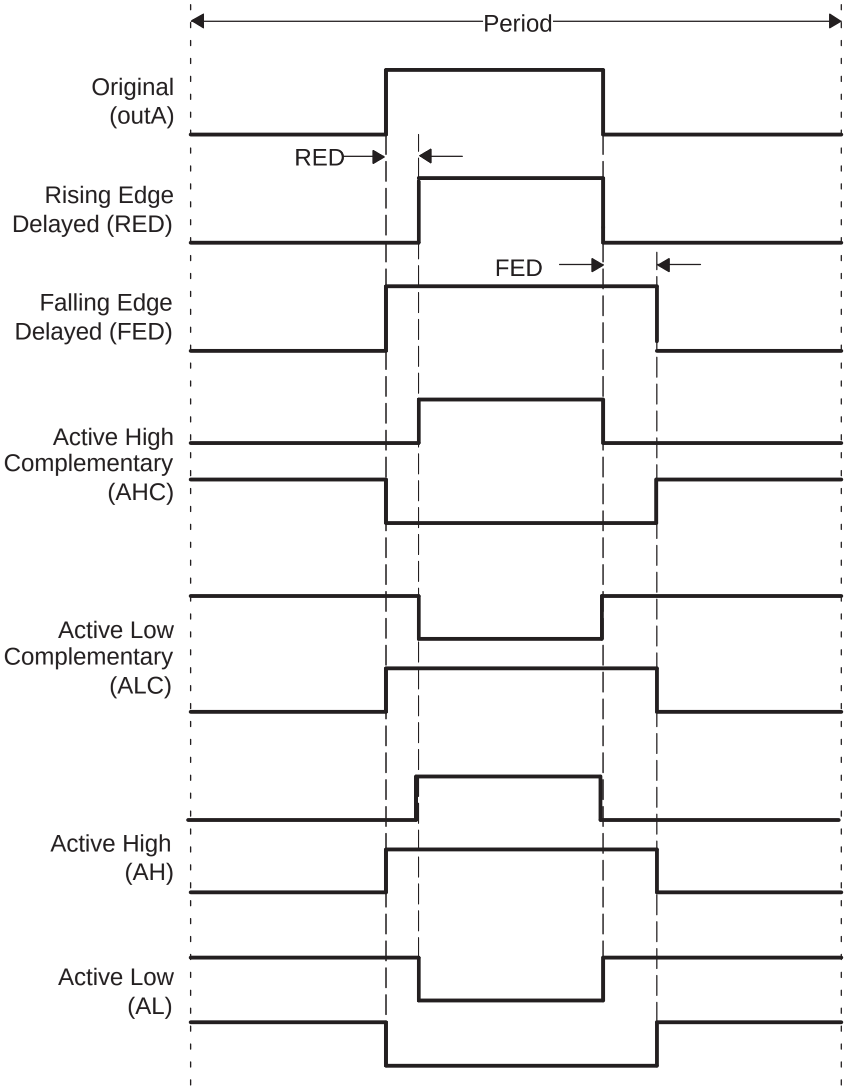
  Finally the last two entries in Table 19-14 show combinations where either the falling-edge-delay (FED) or rising-edge-delay (RED) blocks are bypassed.

Table 19-14. Classical Dead-Band Operating Modes

		DBCTL[POLSEL]		DBCTL[OUT_MODE]	
Mode	**Mode Description**	**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

Figure 19-29 shows waveforms for typical cases where 0% < duty < 100%.

Figure 19-29. 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 VCLK4.

For convenience, delay values for various TBCLK options are shown in Table 19-15.

Table 19-15. Dead-Band Delay Values in μS as a Function of DBFED and DBRED

Dead-Band Value	Dead-Band Delay in μS		
DBFED, DBRED	TBCLK = VCLK4/1	TBCLK = VCLK4 /2	TBCLK = VCLK4/4
1	0.02 μS	0.03 μS	0.07 μS
5	0.08 μS	0.17 μS	0.33 μS
10	0.17 μS	0.33 μS	0.67 μS
100	1.67 μS	3.33 μS	6.67 μS
200	3.33 μS	6.67 μS	13.33 μS
400	6.67 μS	13.33 μS	26.67 μS
500	8.33 μS	16.67 μS	33.33 μS
600	10.00 μS	20.00 μS	40.00 μS
700	11.67 μS	23.33 μS	46.67 μS
800	13.33 μS	26.67 μS	53.33 μS
900	15.00 μS	30.00 μS	60.00 μS
1000	16.67 μS	33.33μS	66.67 μS

When half-cycle clocking is enabled, the formula to calculate the falling-edge-delay and rising-edge-delay becomes:

$FED = DBFED \times T_{TBCLK}/2$

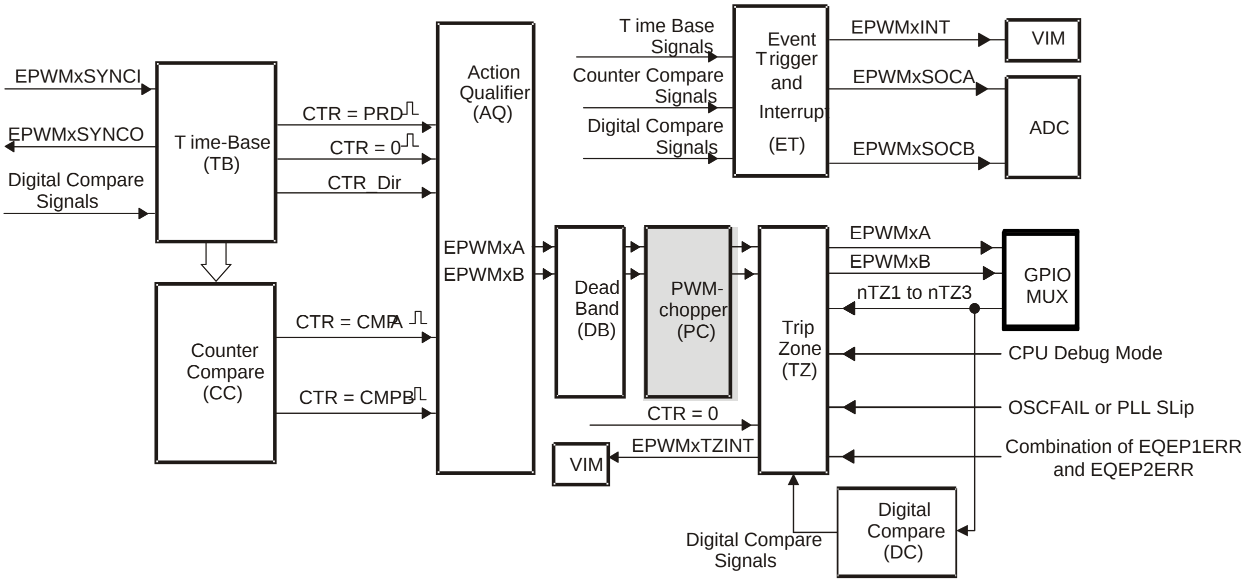
$RED = DBRED \times T_{TBCLK}/2$

19.2.6 PWM-Chopper (PC) Submodule

Figure 19-30 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 19-30. PWM-Chopper Submodule



19.2.6.1 Purpose of the PWM-Chopper Submodule

The key functions of the PWM-chopper submodule are:

- Programmable chopping (carrier) frequency
- Programmable pulse width of first pulse
- Programmable duty cycle of second and subsequent pulses
- Can be fully bypassed if not required

19.2.6.2 Controlling the PWM-Chopper Submodule

The PWM-chopper submodule operation is controlled via the registers in Table 19-16.

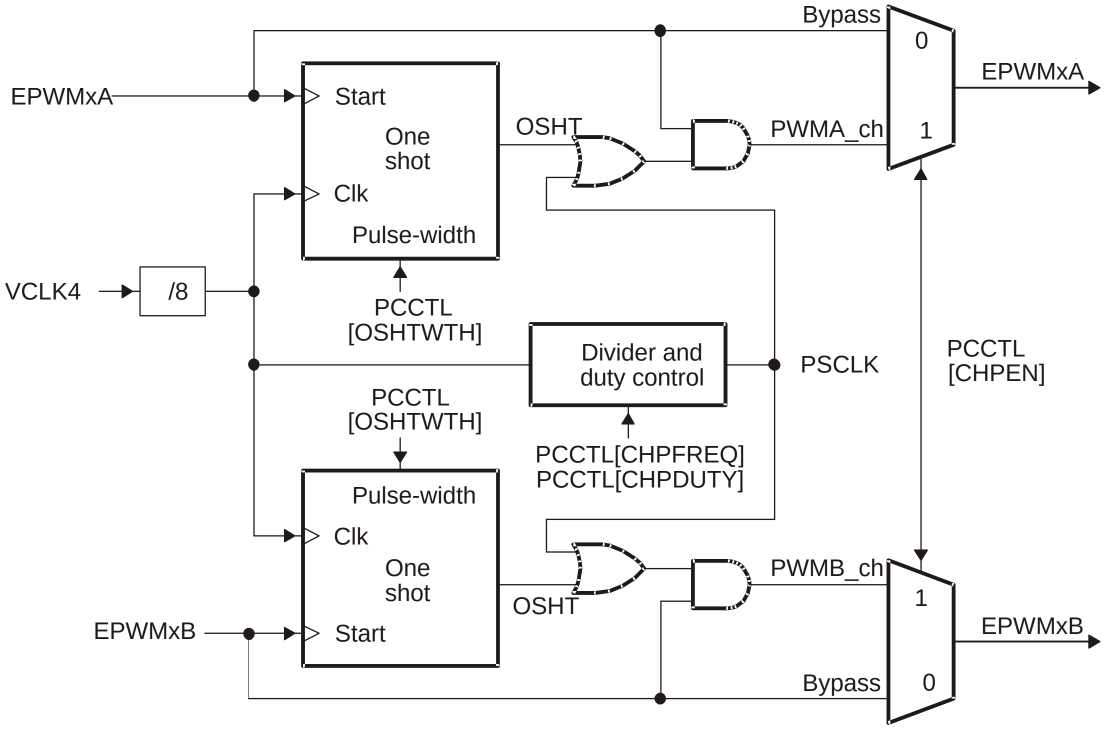
Table 19-16. PWM-Chopper Submodule Registers

Register Name	Address Offset	Shadowed	Description
PCCTL	0x003C	No	PWM-chopper Control Register

19.2.6.3 Operational Highlights for the PWM-Chopper Submodule

Figure 19-31 shows the operational details of the PWM-chopper submodule. The carrier clock is derived from VCLK4. 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.

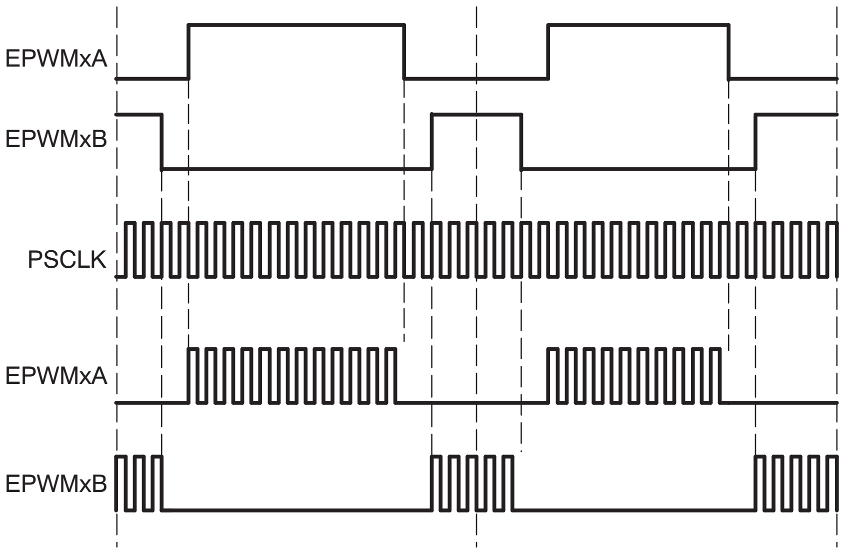
Figure 19-31. PWM-Chopper Submodule Operational Details



19.2.6.4 Waveforms

Figure 19-32 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 19-32. Simple PWM-Chopper Submodule Waveforms Showing Chopping Action Only



19.2.6.4.1 One-Shot Pulse

The width of the first pulse can be programmed to any of 16 possible pulse width values. The width or period of the first pulse is given by:

$T_{1stpulse} = T_{VCLK4} \times 8 \times OSHTWTH$

Where T_{VCLK4} is the period of the system clock (VCLK4) and OSHTWTH is the four control bits (value from 1 to 16)

Figure 19-33 shows the first and subsequent sustaining pulses and Table 19-17 gives the possible pulse width values for a VCLK4 = 100 MHz.

Figure 19-33. PWM-Chopper Submodule Waveforms Showing the First Pulse and Subsequent Sustaining Pulses

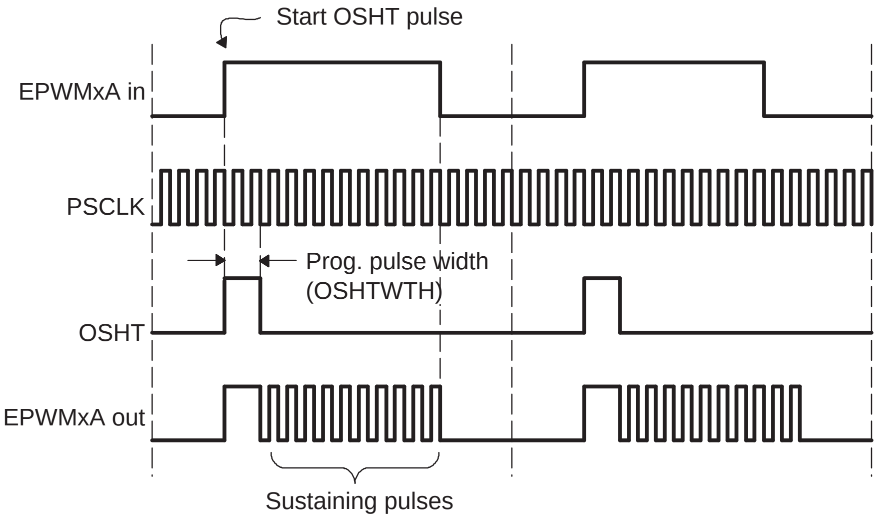


Table 19-17. Possible Pulse Width Values for VCLK4 = 100 MHz

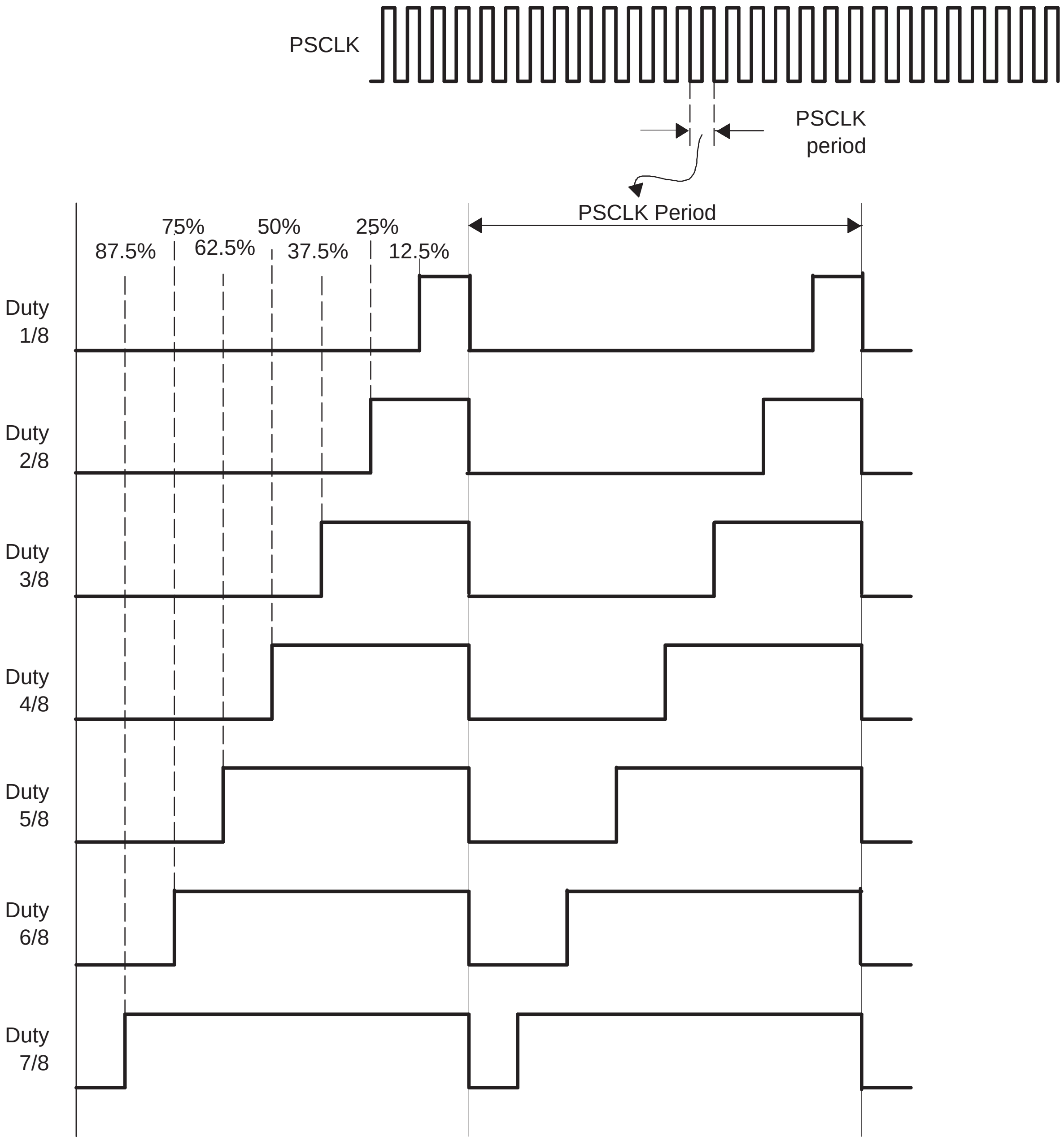
OSHTWTHz (hex)	Pulse Width (nS)
0	100
1	200
2	300
3	400
4	500
5	600
6	700
7	800
8	900
9	1000
A	1100
B	1200
C	1300
D	1400
E	1500
F	1600

19.2.6.4.2 Duty Cycle Control

Pulse transformer-based gate drive designs need to comprehend the magnetic properties or characteristics of the transformer and associated circuitry. Saturation is one such consideration. To assist the gate drive designer, the duty cycles of the second and subsequent pulses have been made programmable. These sustaining pulses ensure the correct drive strength and polarity is maintained on the power switch gate during the on period, and hence a programmable duty cycle allows a design to be tuned or optimized via software control.

Figure 19-34 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 19-34. PWM-Chopper Submodule Waveforms Showing the Pulse Width (Duty Cycle) Control of Sustaining Pulses

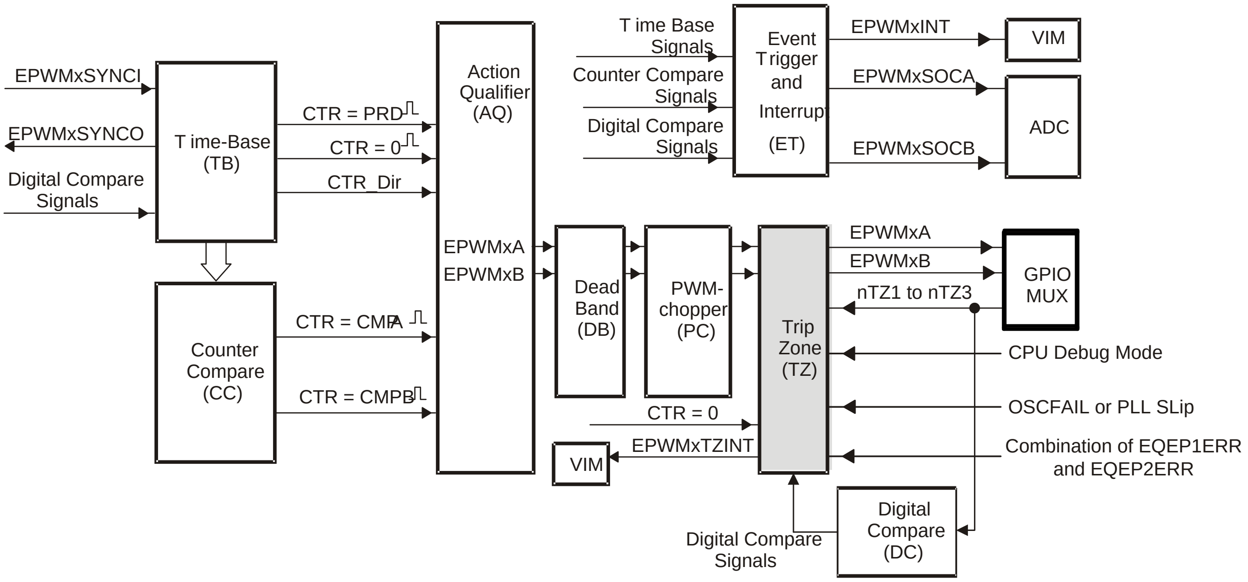


19.2.7 Trip-Zone (TZ) Submodule

Figure 19-35 shows how the trip-zone (TZ) submodule fits within the ePWM module.

Each ePWM module is connected to six $\overline{TZn}$ signals ($\overline{TZ1}$ to $\overline{TZ6}$). $\overline{TZ1}$ to $\overline{TZ3}$ are sourced from the GPIO mux. $\overline{TZ4}$ is sourced from a combination of EQEP1ERR and EQEP2ERR signals. $\overline{TZ5}$ is connected to the system oscillator or PLL clock fail logic, and $\overline{TZ6}$ is sourced from the debug mode halt indication output from the CPU. These signals indicate fault or trip conditions, and the ePWM outputs can be programmed to respond accordingly when faults occur.

Figure 19-35. Trip-Zone Submodule



19.2.7.1 Purpose of the Trip-Zone Submodule

The key functions of the Trip-Zone submodule are:

- Trip inputs $\overline{TZ1}$ to $\overline{TZ6}$ are mapped to all ePWM modules.
- Upon a fault indication, either no action is taken or the ePWM outputs EPWMxA and EPWMxB can be forced to one of the following:
  - High
  - Low
  - High-impedance
- Support for one-shot trip (OSHT) for major short circuits or over-current conditions.
- Support for cycle-by-cycle tripping (CBC) for current limiting operation.
- Support for digital compare tripping (DC) based on state of on-chip analog comparator module outputs and/or $\overline{TZ1}$ to $\overline{TZ3}$ signals.
- Each trip-zone input and digital compare (DC) submodule DCAEVT1/2 or DCBEVT1/2 force event can be allocated to either one-shot or cycle-by-cycle operation.
- Interrupt generation is possible on any trip-zone input.
- Software-forced tripping is also supported.
- The trip-zone submodule can be fully bypassed if it is not required.

19.2.7.2 Controlling and Monitoring the Trip-Zone Submodule

The trip-zone submodule operation is controlled and monitored through the following registers:

Table 19-18. Trip-Zone Submodule Registers

Register Name	Address Offset	Shadowed	Description [1]
TZSEL	0x0024	No	Trip-Zone Select Register
TZDCSEL	0x0026	No	Trip-zone Digital Compare Select Register [2]
TZCTL	0x0028	No	Trip-Zone Control Register
TZEINT	0x002A	No	Trip-Zone Enable Interrupt Register
TZFLG	0x002C	No	Trip-Zone Flag Register
TZCLR	0x002E	No	Trip-Zone Clear Register
TZFRC	0x0030	No	Trip-Zone Force Register

[1] All trip-zone registers are writable only in privileged mode.

[2] This register is discussed in more detail in Section 19.2.9.

19.2.7.3 Operational Highlights for the Trip-Zone Submodule

The following sections describe the operational highlights and configuration options for the trip-zone submodule.

The trip-zone signals $\overline{\text{TZ1}}$ to $\overline{\text{TZ6}}$ (also collectively referred to as $\overline{\text{TZn}}$) are active low input signals. When one of these signals goes low, or when a DCAEVT1/2 or DCBEVT1/2 force happens based on the TZDCSEL register event selection, it indicates that a trip event has occurred. Each ePWM module can be individually configured to ignore or use each of the trip-zone signals or DC events. Which trip-zone signals or DC events are used by a particular ePWM module is determined by the TZSEL register for that specific ePWM module. The trip-zone signals may or may not be synchronized to the system clock (VCLK4) and digitally filtered within the GPIO MUX block. A minimum of 3*TBCLK low pulse width on $\overline{\text{TZn}}$ inputs is sufficient to trigger a fault condition on the ePWM module. If the pulse width is less than this, the trip condition may not be latched. The asynchronous trip makes sure that if clocks are missing for any reason, the outputs can still be tripped by a valid event present on $\overline{\text{TZn}}$ inputs. The GPIOs or peripherals must be appropriately configured. For more information, see the IOMM chapter of the device technical reference manual.

Each $\overline{\text{TZn}}$ input can be individually configured to provide either a cycle-by-cycle or one-shot trip event for an ePWM module. DCAEVT1 and DCBEVT1 events can be configured to directly trip an ePWM module or provide a one-shot trip event to the module. Likewise, DCAVET2 and DCBEVT2 events can also be configured to directly trip an ePWM module or provide a cycle-by-cycle trip event to the module. This configuration is determined by the TZSEL[DCAEVT1/2], TZSEL[DCBEVT1/2], TZSEL[CBCn], and TZSEL[OSHTn] control bits (where n corresponds to the trip input) respectively.

- **Cycle-by-Cycle (CBC):**

  When a cycle-by-cycle trip event occurs, the action specified in the TZCTL[TZA] and TZCTL[TZB] bits is carried out immediately on the EPWMxA and/or EPWMxB output. Table 19-19 lists the possible actions. In addition, the cycle-by-cycle trip event flag (TZFLG[CBC]) is set and a EPWMx_TZINT interrupt is generated if it is enabled in the TZEINT register and VIM peripheral.

  If the CBC interrupt is enabled via the TZEINT register, and DCAEVT2 or DCBEVT2 are selected as CBC trip sources via the TZSEL register, it is not necessary to also enable the DCAEVT2 or DCBEVT2 interrupts in the TZEINT register, as the DC events trigger interrupts through the CBC mechanism.

  The specified condition on the inputs is automatically cleared when the ePWM time-base counter reaches zero (TBCTR = 0x0000) if the trip event is no longer present. Therefore, in this mode, the trip event is cleared or reset every PWM cycle. The TZFLG[CBC] flag bit will remain set until it is manually cleared by writing to the TZCLR[CBC] bit. If the cycle-by-cycle trip event is still present when the TZFLG[CBC] bit is cleared, then it will again be immediately set.

- **One-Shot (OSHT):**

  When a one-shot trip event occurs, the action specified in the TZCTL[TZA] and TZCTL[TZB] bits is carried out immediately on the EPWMxA and/or EPWMxB output. Table 19-19 lists the possible actions. In addition, the one-shot trip event flag (TZFLG[OST]) is set and a EPWMx_TZINT interrupt is generated if it is enabled in the TZEINT register and VIM peripheral. The one-shot trip condition must be cleared manually by writing to the TZCLR[OST] bit.

  If the one-shot interrupt is enabled via the TZEINT register, and DCAEVT1 or DCBEVT1 are selected as OSHT trip sources via the TZSEL register, it is not necessary to also enable the DCAEVT1 or DCBEVT1 interrupts in the TZEINT register, as the DC events trigger interrupts through the OSHT mechanism.

- **Digital Compare Events (DCAEVT1/2 and DCBEVT1/2):**

  A digital compare DCAEVT1/2 or DCBEVT1/2 event is generated based on a combination of the DCAH/DCAL and DCBH/DCBL signals as selected by the TZDCSEL register. The signals which source the DCAH/DCAL and DCBH/DCBL signals are selected via the DCTRIPSEL register and can be either trip zone input pins. For more information on the digital compare submodule signals, see Section 19.2.9.

  When a digital compare event occurs, the action specified in the TZCTL[DCAEVT1/2] and TZCTL[DCBEVT1/2] bits is carried out immediately on the EPWMxA and/or EPWMxB output. Table 19-19 lists the possible actions. In addition, the relevant DC trip event flag (TZFLG[DCAEVT1/2] / TZFLG[DCBEVT1/2]) is set and a EPWMx_TZINT interrupt is generated if it is enabled in the TZEINT register and VIM peripheral.

  The specified condition on the pins is automatically cleared when the DC trip event is no longer present. The TZFLG[DCAEVT1/2] or TZFLG[DCBEVT1/2] flag bit will remain set until it is manually cleared by writing to the TZCLR[DCAEVT1/2] or TZCLR[DCBEVT1/2] bit. If the DC trip event is still present when the TZFLG[DCAEVT1/2] or TZFLG[DCBEVT1/2] flag is cleared, then it will again be immediately set.

The action taken when a trip event occurs can be configured individually for each of the ePWM output pins by way of the TZCTL register bit fields. One of four possible actions, shown in Table 19-19, can be taken on a trip event.

Table 19-19. Possible Actions On a Trip Event

TZCTL Register bit-field Settings	EPWMxA and/or EPWMxB	Comment
0,0	High-Impedance	Tripped
0,1	Force to High State	Tripped
1,0	Force to Low State	Tripped
1,1	No Change	Do Nothing. No change is made to the output.

Example 19-7. Trip-Zone Configurations

Scenario A:
A one-shot trip event on $\overline{\text{TZ1}}$ pulls both EPWM1A, EPWM1B low and also forces EPWM2A and EPWM2B high.

- Configure the ePWM1 registers as follows:
  - TZSEL[OSHT1] = 1: enables $\overline{\text{TZ1}}$ as a one-shot event source for ePWM1
  - TZCTL[TZA] = 2: EPWM1A will be forced low on a trip event.
  - TZCTL[TZB] = 2: EPWM1B will be forced low on a trip event.
- Configure the ePWM2 registers as follows:
  - TZSEL[OSHT1] = 1: enables $\overline{\text{TZ1}}$ as a one-shot event source for ePWM2
  - TZCTL[TZA] = 1: EPWM2A will be forced high on a trip event.
  - TZCTL[TZB] = 1: EPWM2B will be forced high on a trip event.

Scenario B:
A cycle-by-cycle event on $\overline{\text{TZ5}}$ pulls both EPWM1A, EPWM1B low.
A one-shot event on $\overline{\text{TZ1}}$ or $\overline{\text{TZ6}}$ puts EPWM2A into a high impedance state.

- Configure the ePWM1 registers as follows:
  - TZSEL[CBC5] = 1: enables $\overline{\text{TZ5}}$ 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 $\overline{\text{TZ1}}$ as a one-shot event source for ePWM2
  - TZSEL[OSHT6] = 1: enables $\overline{\text{TZ6}}$ as a one-shot event source for ePWM2
  - TZCTL[TZA] = 0: EPWM2A will be put into a high-impedance state on a trip event.
  - TZCTL[TZB] = 3: EPWM2B will ignore the trip event.

19.2.7.4 Generating Trip Event Interrupts

Figure 19-36 and Figure 19-37 illustrate the trip-zone submodule control and interrupt logic, respectively. DCAEVT1/2 and DCBEVT1/2 signals are described in further detail in Section 19.2.9.

Figure 19-36. Trip-Zone Submodule Mode Control Logic

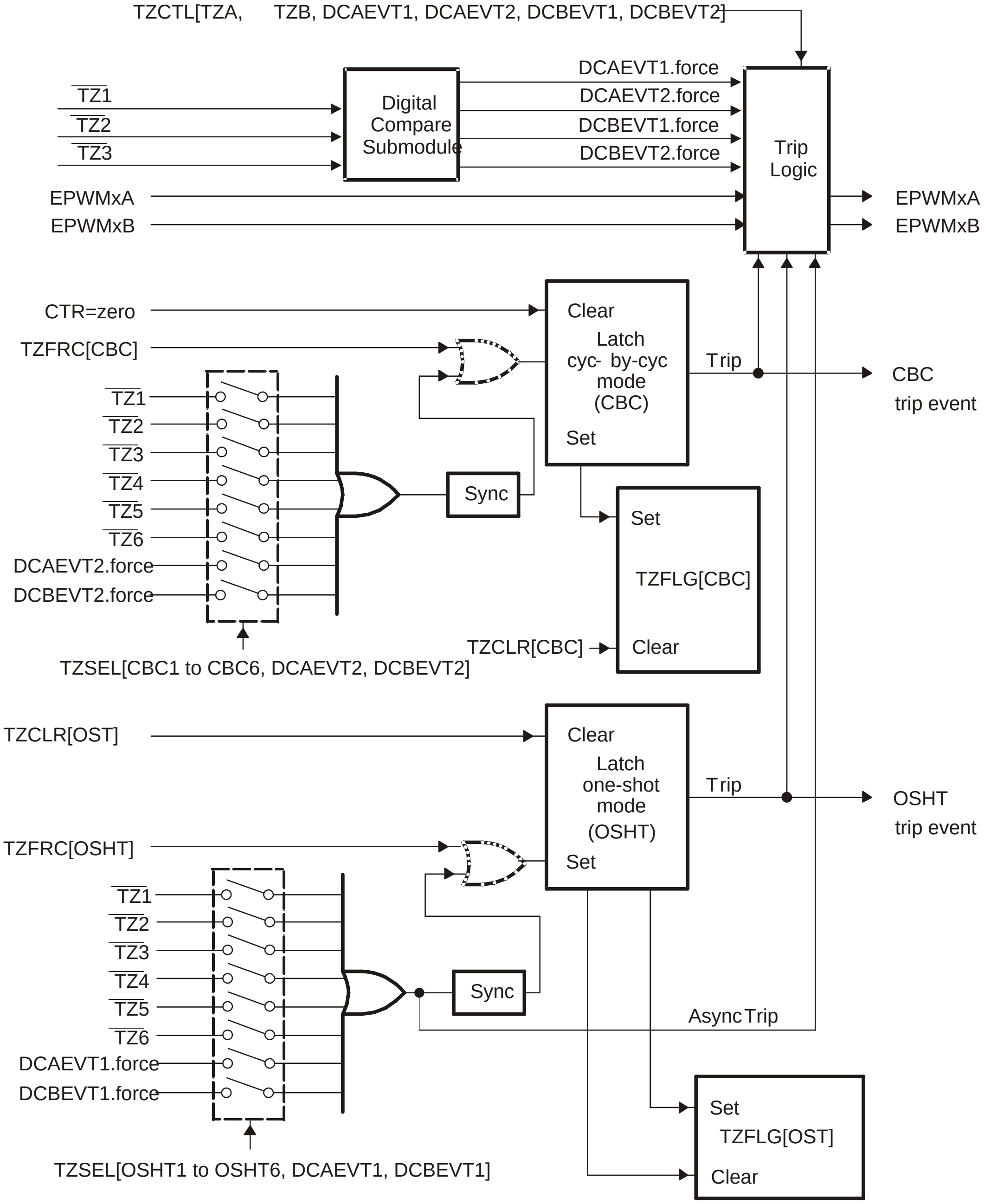
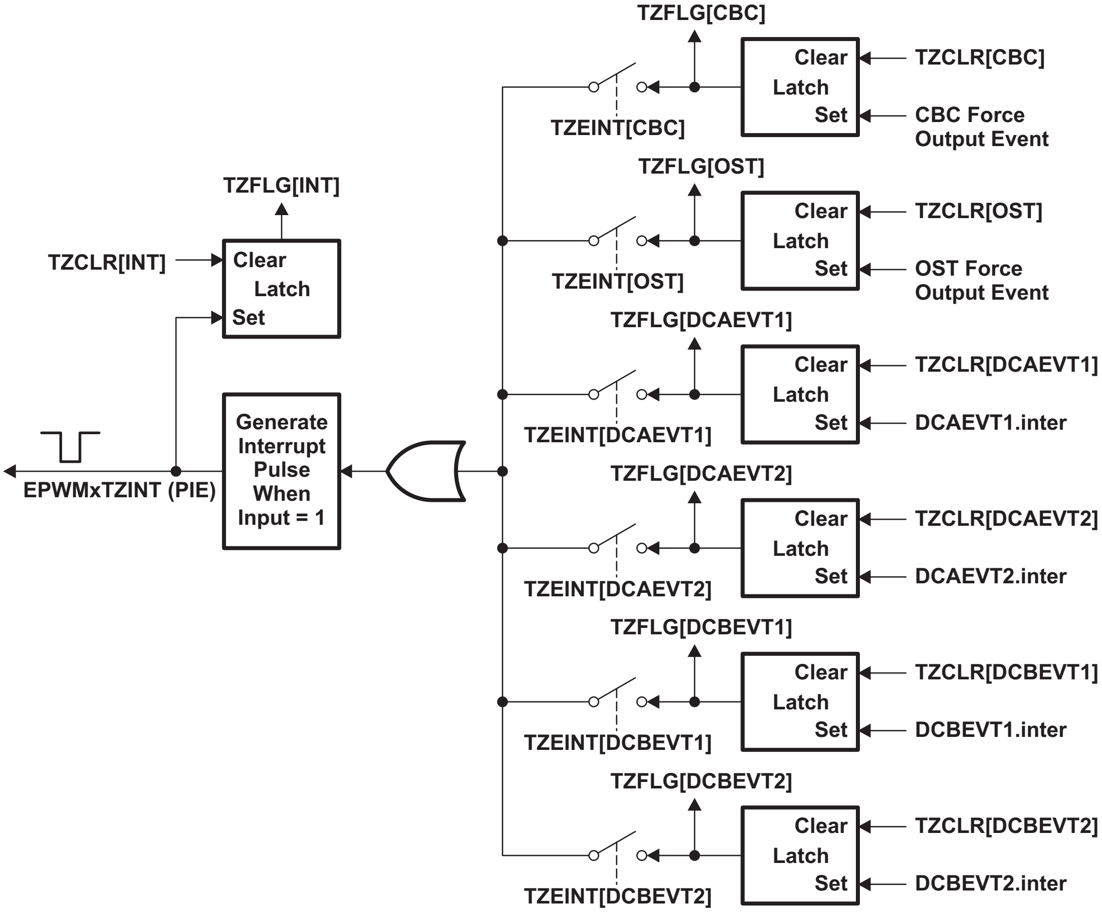


Figure 19-37. Trip-Zone Submodule Interrupt Logic



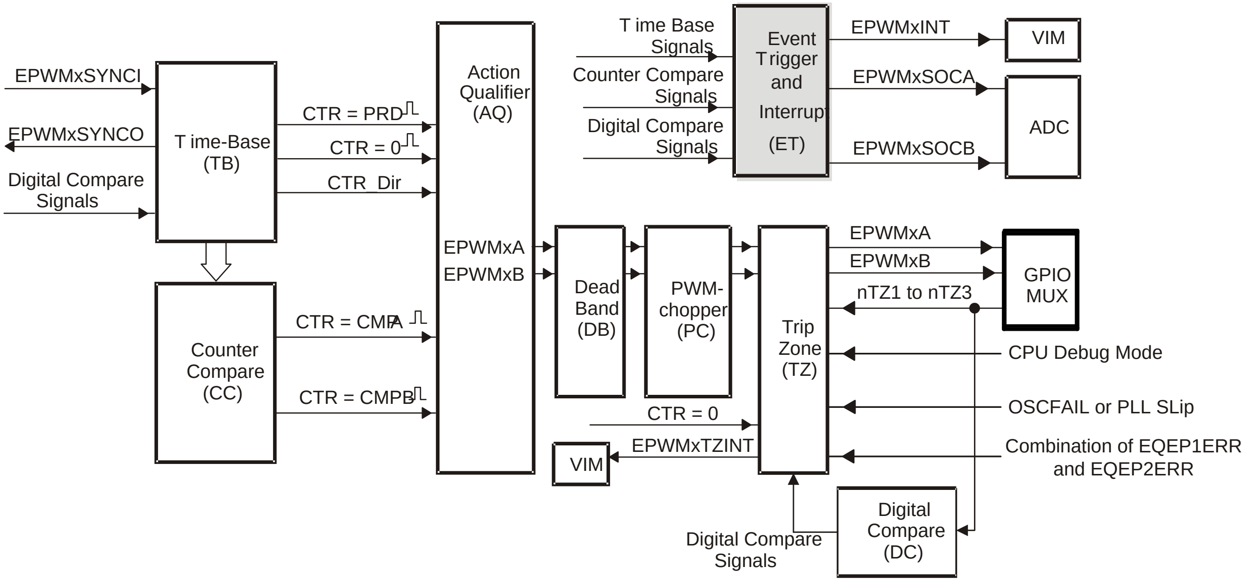
19.2.8 Event-Trigger (ET) Submodule

The key functions of the event-trigger submodule are:

- Receives event inputs generated by the time-base, counter-compare and digital-compare submodules
- Uses the time-base direction information for up/down event qualification
- Uses prescaling logic to issue interrupt requests and ADC start of conversion at:
  - Every event
  - Every second event
  - Every third event
- Provides full visibility of event generation via event counters and flags
- Allows software forcing of Interrupts and ADC start of conversion

The event-trigger submodule manages the events generated by the time-base submodule, the counter-compare submodule, and the digital-compare submodule to generate an interrupt to the CPU and/or a start of conversion pulse to the ADC when a selected event occurs. Figure 19-38 illustrates where the event-trigger submodule fits within the ePWM system.

Figure 19-38. Event-Trigger Submodule



19.2.8.1 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 VIM and two start of conversion signals connected to the ADC module. As shown in Figure 19-39, the ePWMxSOCA and ePWMxSOCB signals are combined to generate four special signals that can be used to trigger an ADC start of conversion, and hence multiple modules can initiate an ADC start of conversion via the ADC trigger inputs.

The event-trigger submodule monitors various event conditions (the left side inputs to event-trigger submodule shown in Figure 19-40) and can be configured to prescale these events before issuing an Interrupt request or an ADC start of conversion. The event-trigger prescaling logic can issue Interrupt requests and ADC start of conversion at:

- Every event
- Every second event
- Every third event

Figure 19-39. Event-Trigger Submodule Inter-Connectivity of ADC Start of Conversion

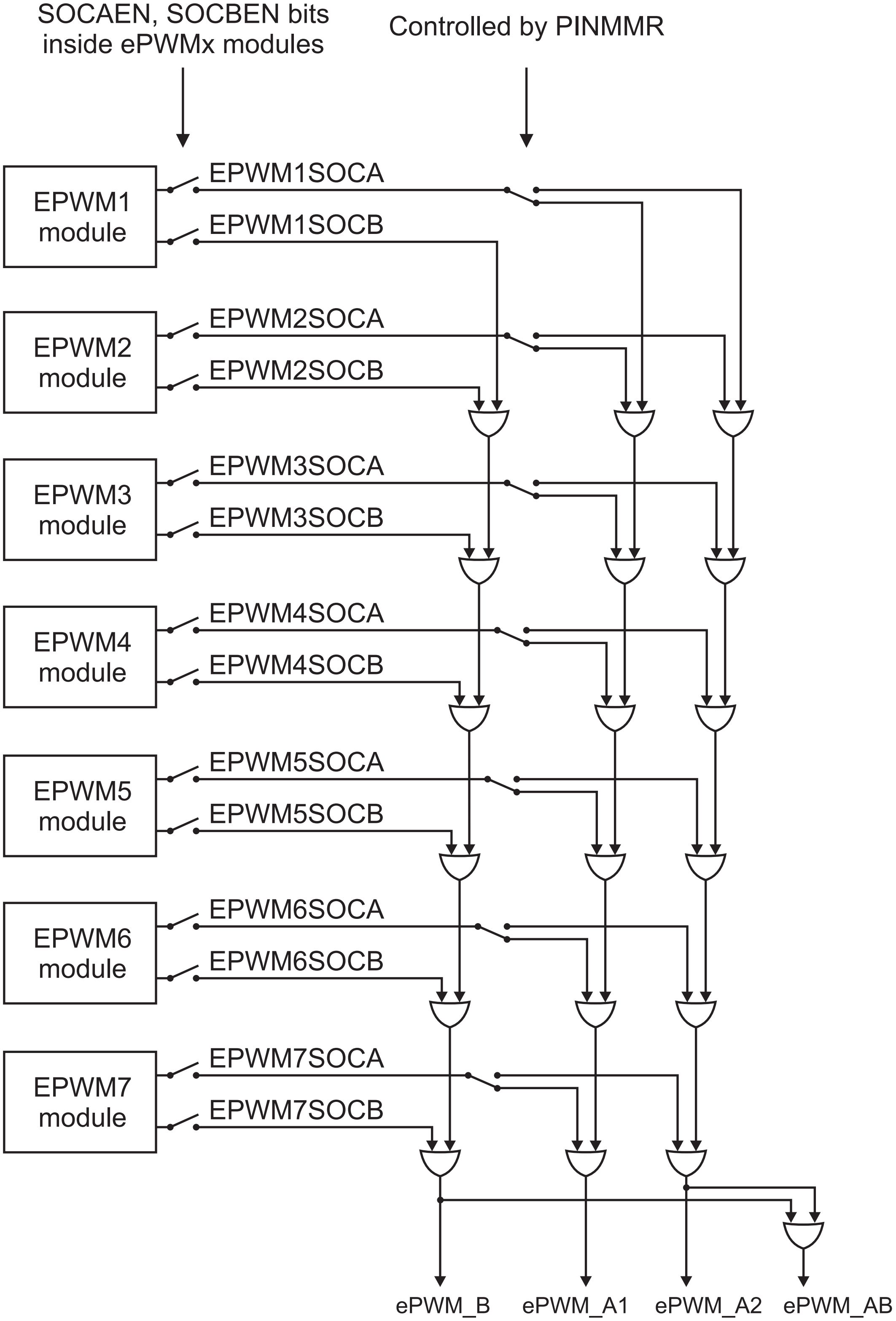
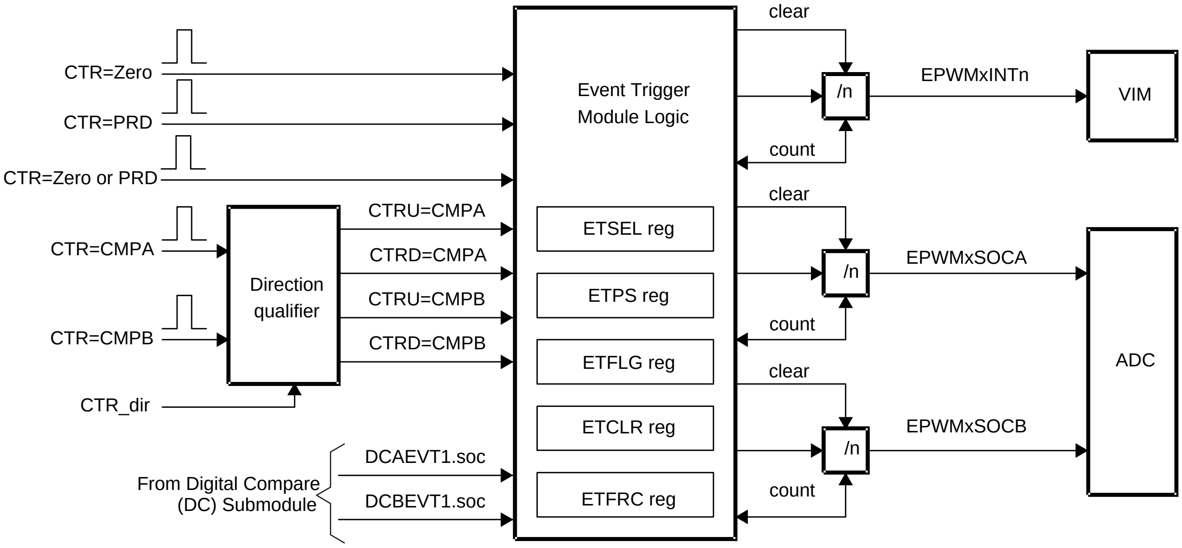


Figure 19-40. Event-Trigger Submodule Showing Event Inputs and Prescaled Outputs



The key registers used to configure the event-trigger submodule are shown in Table 19-20.

Table 19-20. Event-Trigger Submodule Registers

Register Name	Address Offset	Shadowed	Description
ETSEL	0x0032	No	Event-trigger Selection Register
ETPS	0x0034	No	Event-trigger Prescale Register
ETFLG	0x0036	No	Event-trigger Flag Register
ETCLR	0x0038	No	Event-trigger Clear Register
ETFRC	0x003A	No	Event-trigger Force Register

- ETSEL—This selects which of the possible events will trigger an interrupt or start an ADC conversion
- ETPS—This programs the event prescaling options mentioned above.
- ETFLG—These are flag bits indicating status of the selected and prescaled events.
- ETCLR—These bits allow you to clear the flag bits in the ETFLG register via software.
- ETFRC—These bits allow software forcing of an event. Useful for debugging or s/w intervention.

A more detailed look at how the various register bits interact with the Interrupt and ADC start of conversion logic are shown in Figure 19-41, Figure 19-42, and Figure 19-43.

Figure 19-41 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

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 (TBCTR = 0x0000).
- Time-base counter equal to period (TBCTR = TBPRD).
- Time-base counter equal to zero or period (TBCTR = 0x0000 || TBCTR = TBPRD)

- Time-base counter equal to the compare A register (CMPA) when the timer is incrementing.
- Time-base counter equal to the compare A register (CMPA) when the timer is decrementing.
- Time-base counter equal to the compare B register (CMPB) when the timer is incrementing.
- Time-base counter equal to the compare B register (CMPB) when the timer is decrementing.

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 VIM.

When ETPS[INTCNT] reaches ETPS[INTPRD] 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.

The above definition means that you can generate an interrupt on every event, on every second event, or on every third event. An interrupt cannot be generated on every fourth or more events.

Figure 19-41. Event-Trigger Interrupt Generator

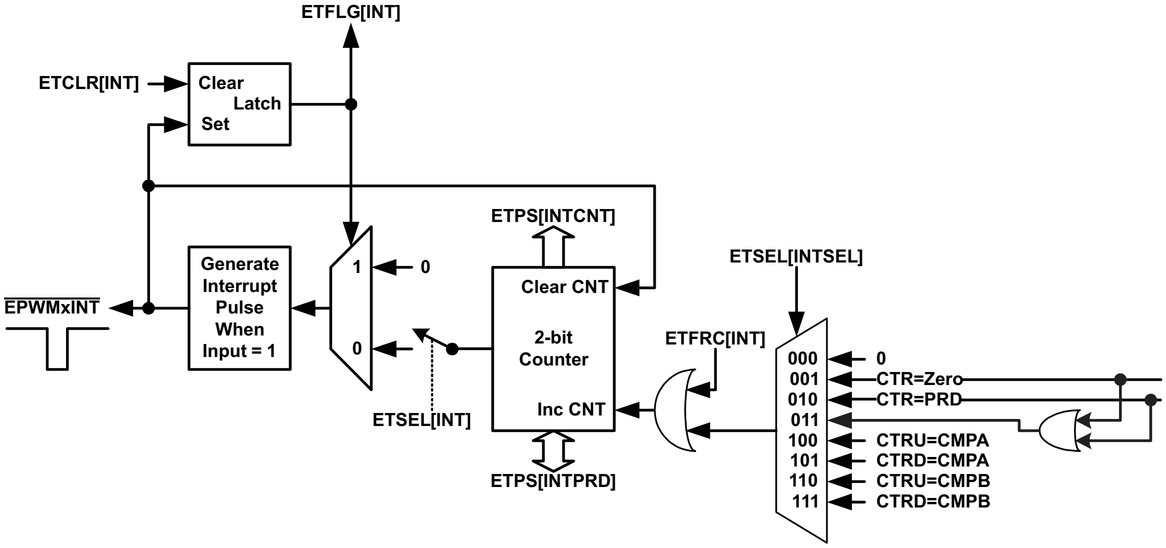
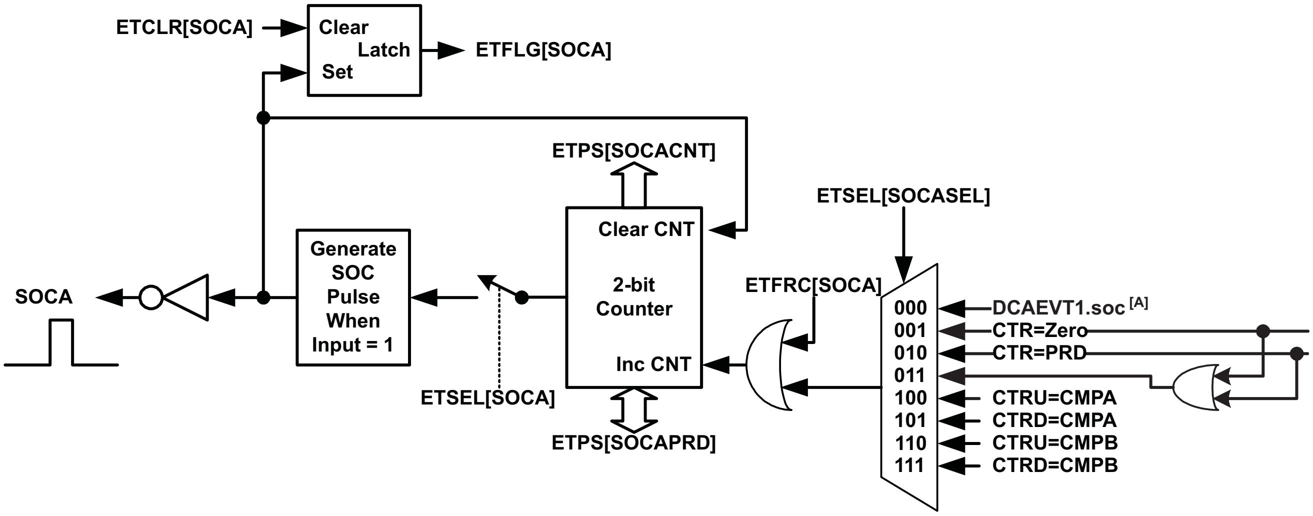


Figure 19-42 shows the operation of the event-trigger's start-of-conversion-A (SOCA) pulse generator. The ETPS[SOCACNT] counter and ETPS[SOCAPRD] period values behave similarly to the interrupt generator except that the pulses are continuously generated. That is, the pulse flag ETFLG[SOCA] is latched when a pulse is generated, but it does not stop further pulse generation. The enable/disable bit ETSEL[SOCAEN] stops pulse generation, but input events can still be counted until the period value is reached as with the interrupt generation logic. The event that will trigger an SOCA and SOCB pulse can be configured separately in the ETSEL[SOCASEL] and ETSEL[SOCBSEL] bits. The possible events are the same events that can be specified for the interrupt generation logic with the addition of the DCAEVT1.soc and DCBEVT1.soc event signals from the digital compare (DC) submodule.

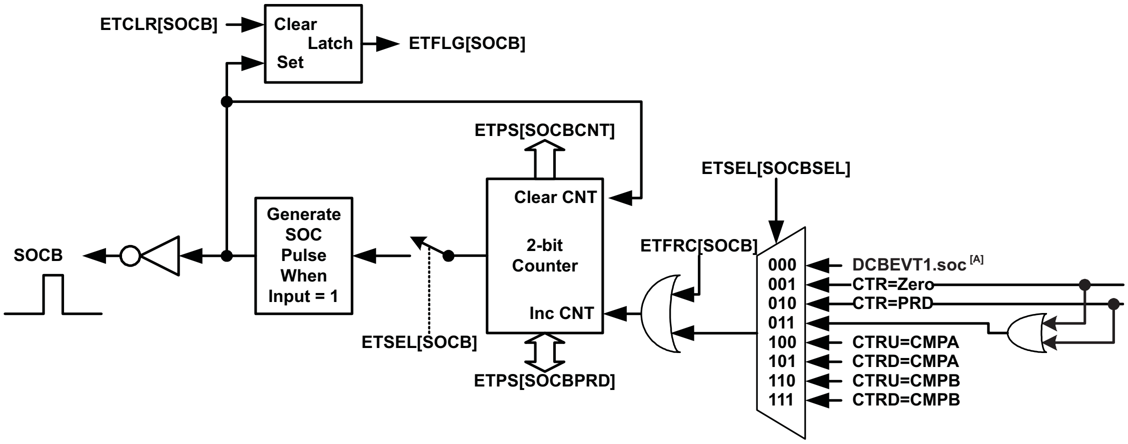
Figure 19-42. Event-Trigger SOCA Pulse Generator



A The DCAEVT1.soc signals are signals generated by the Digital compare (DC) submodule described later in Section 19.2.9.

Figure 19-43 shows the operation of the event-trigger's start-of-conversion-B (SOCB) pulse generator. The event-trigger's SOCB pulse generator operates the same way as the SOCA.

Figure 19-43. Event-Trigger SOCB Pulse Generator



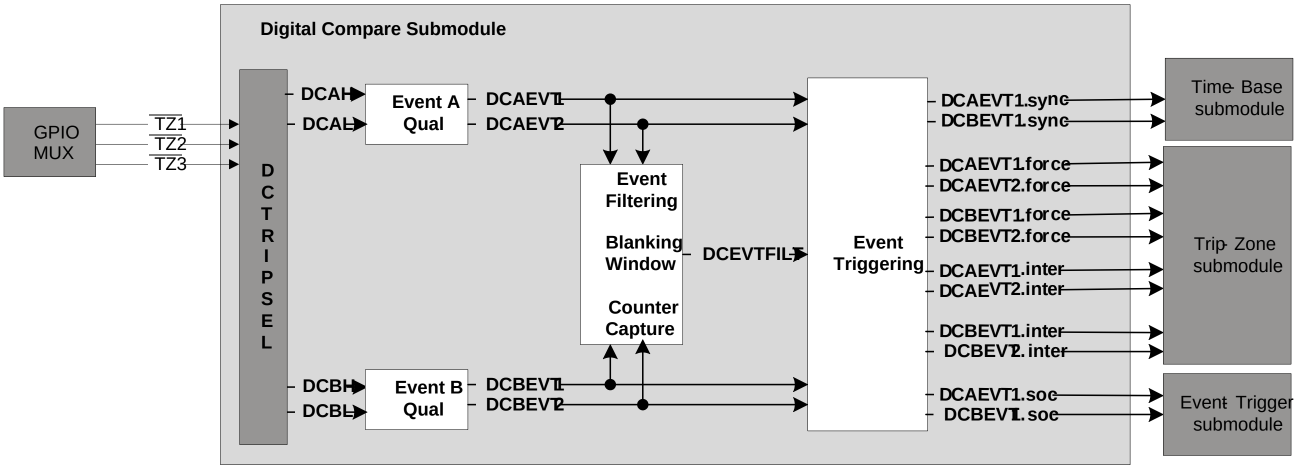
A The DCBEVT1.soc signals are signals generated by the Digital compare (DC) submodule described later in Section 19.2.9.

19.2.9 Digital Compare (DC) Submodule

Figure 19-44 illustrates where the digital compare (DC) submodule signals interface to other submodules in the ePWM system.

The digital compare (DC) submodule compares signals external to the ePWM module to directly generate PWM events/actions that then feed to the event-trigger, trip-zone, and time-base submodules. Additionally, blanking window functionality is supported to filter noise or unwanted pulses from the DC event signals.

Figure 19-44. Digital-Compare Submodule High-Level Block Diagram



19.2.9.1 Purpose of the Digital Compare Submodule

The key functions of the digital compare submodule are:

- $\overline{\text{TZ1}}$, $\overline{\text{TZ2}}$, and $\overline{\text{TZ3}}$ inputs generate Digital Compare A High/Low (DCAH, DCAL) and Digital Compare B High/Low (DCBH, DCBL) signals.
- DCAH/L and DCBH/L signals trigger events which can then either be filtered or fed directly to the trip-zone, event-trigger, and time-base submodules to:
  - generate a trip zone interrupt
  - generate an ADC start of conversion
  - force an event
  - generate a synchronization event for synchronizing the ePWM module TBCTR.
- Event filtering (blanking window logic) can optionally blank the input signal to remove noise.

19.2.9.2 Controlling and Monitoring the Digital Compare Submodule

The digital compare submodule operation is controlled and monitored through the following registers:

Table 19-21. Digital Compare Submodule Registers

Register Name	Address Offset	Shadowed	Description
TZDCSEL[1] [2]	0x0026	No	Trip Zone Digital Compare Select Register
DCTRIPSEL[1]	0x0060	No	Digital Compare Trip Select Register
DCACTL[1]	0x0062	No	Digital Compare A Control Register
DCBCTL[1]	0x0064	No	Digital Compare B Control Register
DCFCTL[1]	0x0066	No	Digital Compare Filter Control Register
DCCAPCTL[1]	0x0068	No	Digital Compare Capture Control Register
DCFOFFSET	0x006A	Writes	Digital Compare Filter Offset Register
DCFOFFSETCNT	0x006C	No	Digital Compare Filter Offset Counter Register
DCFWINDOW	0x006E	No	Digital Compare Filter Window Register
DCFWINDOWCNT	0x0070	No	Digital Compare Filter Window Counter Register
DCCAP	0x0072	Yes	Digital Compare Counter Capture Register

[1] These registers are writable only in privileged mode.

[2] The TZDCSEL register is part of the trip-zone submodule but is mentioned again here because of its functional significance to the digital compare submodule.

19.2.9.3 Operation Highlights of the Digital Compare Submodule

The following sections describe the operational highlights and configuration options for the digital compare submodule.

19.2.9.3.1 Digital Compare Events

As illustrated in Figure 19-44, trip zone inputs ($\overline{\text{TZ1}}$, $\overline{\text{TZ2}}$, and $\overline{\text{TZ3}}$) can be selected via the DCTRIPSEL bits to generate the Digital Compare A High and Low (DCAH/L) and Digital Compare B High and Low (DCBH/L) signals. Then, the configuration of the TZDCSEL register qualifies the actions on the selected DCAH/L and DCBH/L signals, which generate the DCAEVT1/2 and DCBEVT1/2 events (Event Qualification A and B).

NOTE: The $\overline{\text{TZn}}$ signals, when used as a DCEVT tripping functions, are treated as a normal input signal and can be defined to be active high or active low inputs. EPWM outputs are asynchronously tripped when either the $\overline{\text{TZn}}$, DCAEVTx.force, or DCBEVTx.force signals are active. For the condition to remain latched, a minimum of 3 × TBCLK sync pulse width is required. If pulse width is < 3 × TBCLK sync pulse width, the trip condition may or may not get latched by CBC or OST latches.

The DCAEVT1/2 and DCBEVT1/2 events can then be filtered to provide a filtered version of the event signals (DCEVTFILT) or the filtering can be bypassed. Filtering is discussed further in Section 19.2.9.3.2. Either the DCAEVT1/2 and DCBEVT1/2 event signals or the filtered DCEVTFILT event signals can generate a force to the trip zone module, a TZ interrupt, an ADC SOC, or a PWM sync signal.

- **force signal:**

  DCAEVT1/2.force signals force trip zone conditions which either directly influence the output on the EPWMxA pin (via TZCTL[DCAEVT1 or DCAEVT2] configurations) or, if the DCAEVT1/2 signals are selected as one-shot or cycle-by-cycle trip sources (via the TZSEL register), the DCAEVT1/2.force signals can effect the trip action via the TZCTL[TZA] configuration. The DCBEVT1/2.force signals behaves similarly, but affect the EPWMxB output pin instead of the EPWMxA output pin.

  The priority of conflicting actions on the TZCTL register is as follows (highest priority overrides lower priority):

  Output EPWMxA: TZA (highest) -> DCAEVT1 -> DCAEVT2 (lowest)

  Output EPWMxB: TZB (highest) -> DCBEVT1 -> DCBEVT2 (lowest)

- **interrupt signal:**

  DCAEVT1/2.interrupt signals generate trip zone interrupts to the VIM. To enable the interrupt, the user must set the DCAEVT1, DCAEVT2, DCBEVT1, or DCBEVT2 bits in the TZEINT register. Once one of these events occurs, an EPWMxTZINT interrupt is triggered, and the corresponding bit in the TZCLR register must be set in order to clear the interrupt.

- **soc signal:**

  The DCAEVT1.soc signal interfaces with the event-trigger submodule and can be selected as an event which generates an ADC start-of-conversion-A (SOCA) pulse via the ETSEL[SOCASEL] bit. Likewise, the DCBEVT1.soc signal can be selected as an event which generates an ADC start-of-conversion-B (SOCB) pulse via the ETSEL[SOCBSEL] bit.

- **sync signal:**

  The DCAEVT1.sync and DCBEVT1.sync events are ORed with the EPWMxSYNCI input signal and the TBCTL[SWFSYNC] signal to generate a synchronization pulse to the time-base counter.

Figure 19-45 and Figure 19-46 show how the DCAEVT1, DCAEVT2, or DCEVTFLT signals are processed to generate the digital compare A event force, interrupt, soc and sync signals.

Figure 19-45. DCAEVT1 Event Triggering

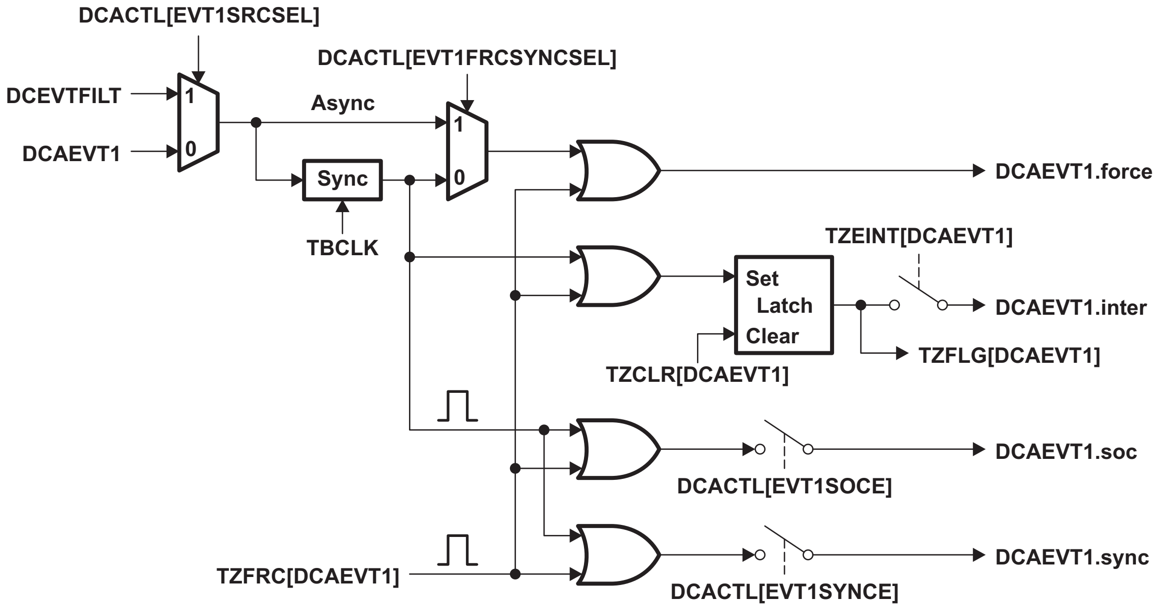


Figure 19-46. DCAEVT2 Event Triggering

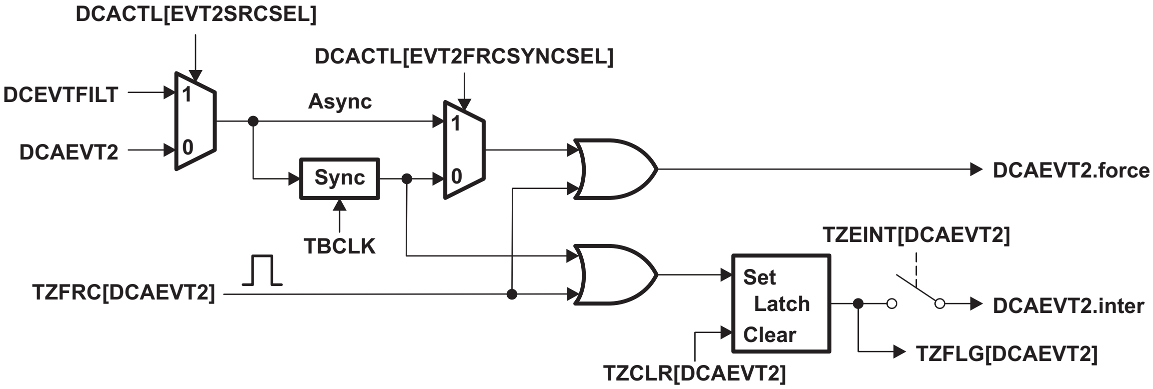


Figure 19-47 and Figure 19-48 show how the DCBEVT1, DCBEVT2, or DCEVTFLT signals are processed to generate the digital compare B event force, interrupt, soc, and sync signals.

Figure 19-47. DCBEVT1 Event Triggering

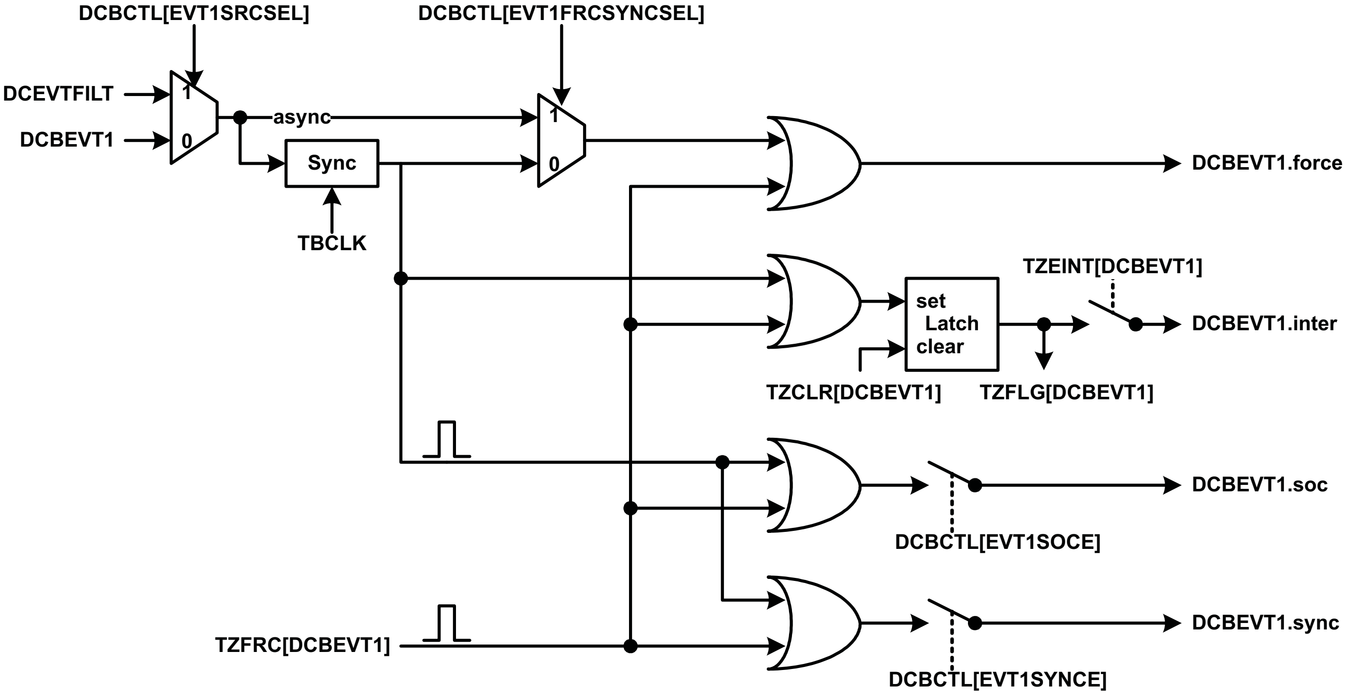
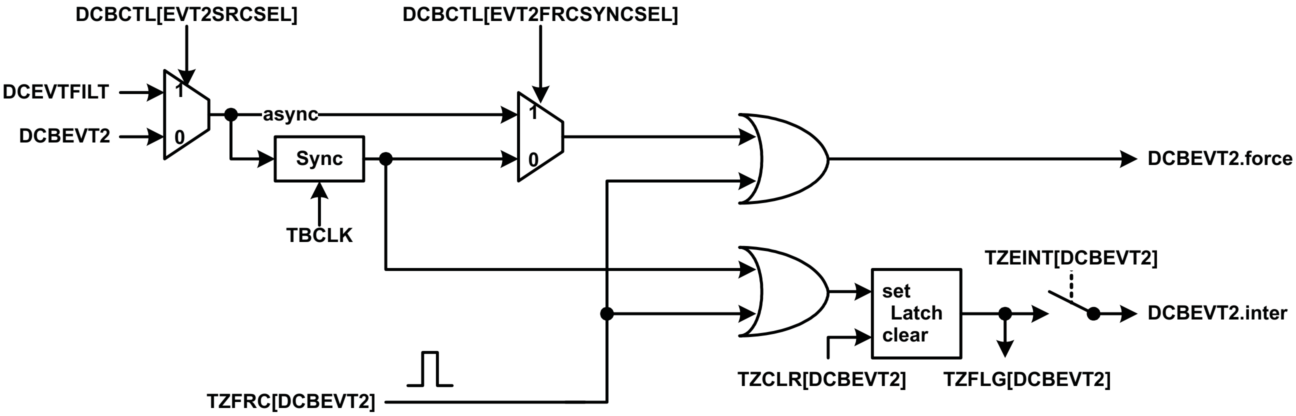


Figure 19-48. DCBEVT2 Event Triggering



19.2.9.3.2 Event Filtering

The DCAEVT1/2 and DCBEVT1/2 events can be filtered via event filtering logic to remove noise by optionally blanking events for a certain period of time. This is useful for cases where the analog comparator outputs may be selected to trigger DCAEVT1/2 and DCBEVT1/2 events, and the blanking logic is used to filter out potential noise on the signal prior to tripping the PWM outputs or generating an interrupt or ADC start-of-conversion. The event filtering can also capture the TBCTR value of the trip event. Figure 19-49 shows the details of the event filtering logic.

If the blanking logic is enabled, one of the digital compare events – DCAEVT1, DCAEVT2, DCBEVT1, DCBEVT2 – is selected for filtering. The blanking window, which filters out all event occurrences on the signal while it is active, will be aligned to either a CTR = PRD pulse or a CTR = 0 pulse (configured by the DCFCTL[PULSESEL] bits). An offset value in TBCLK counts is programmed into the DCFOFFSET register, which determines at what point after the CTR = PRD or CTR = 0 pulse the blanking window starts. The duration of the blanking window, in number of TBCLK counts after the offset counter expires, is written to the DCFWINDOW register by the application. During the blanking window, all events are ignored. Before and after the blanking window ends, events can generate soc, sync, interrupt, and force signals as before.

Figure 19-50 illustrates several timing conditions for the offset and blanking window within an ePWM period. Notice that if the blanking window crosses the CTR = 0 or CTR = PRD boundary, the next window still starts at the same offset value after the CTR = 0 or CTR = PRD pulse.

Figure 19-49. Event Filtering

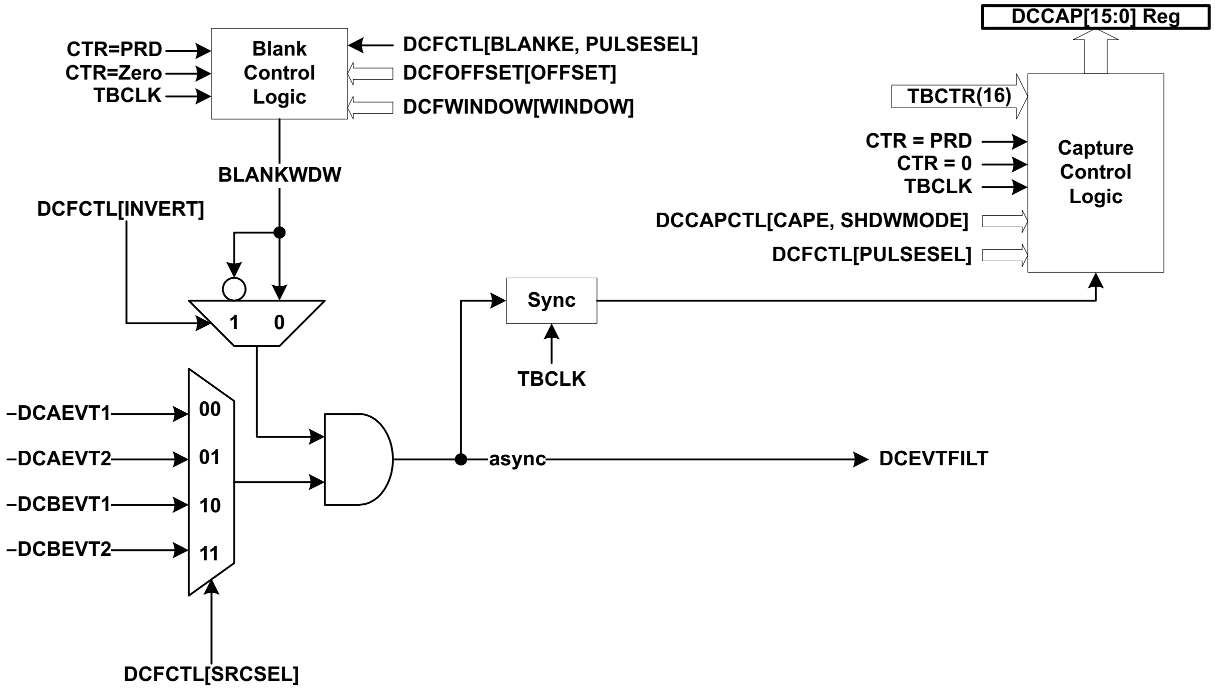
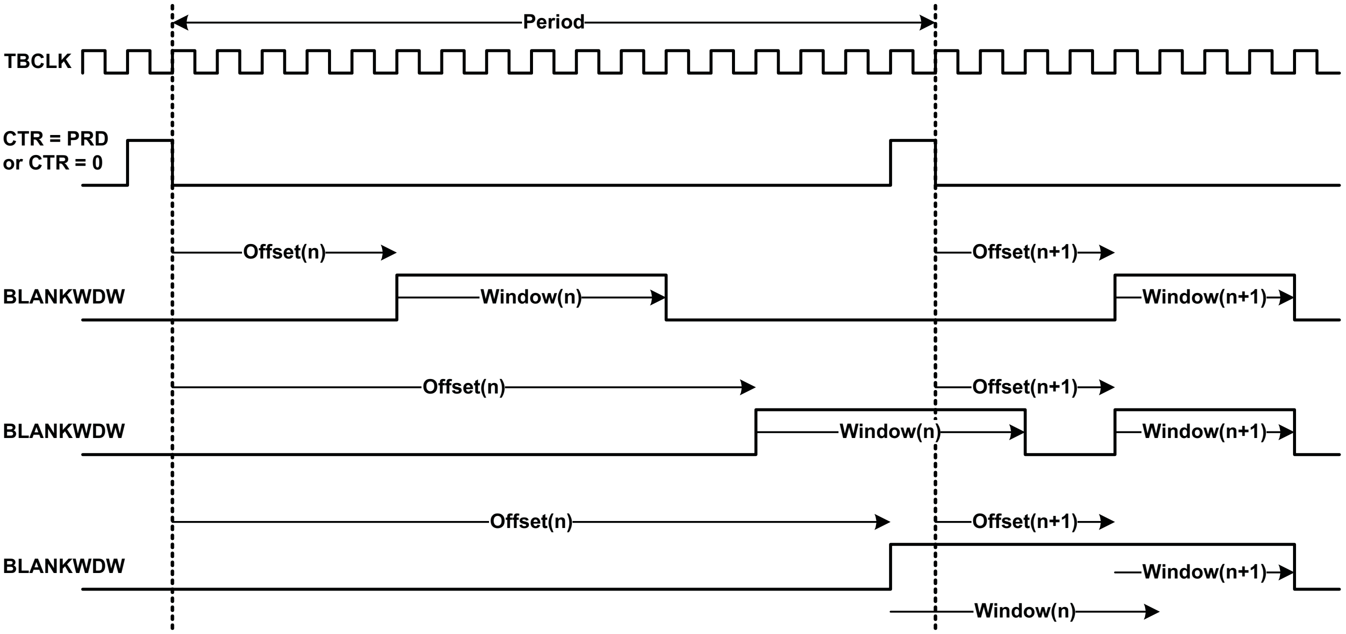


Figure 19-50. Blanking Window Timing Diagram



19.2.10 Proper Interrupt Initialization Procedure

When the ePWM peripheral clock is enabled it may be possible that interrupt flags may be set due to spurious events due to the ePWM registers not being properly initialized. The proper procedure for initializing the ePWM peripheral is as follows:

1. Disable global interrupts (CPU INTM flag)
2. Disable ePWM interrupts
3. Set TBCLKSYNC = 0
4. Initialize peripheral registers
5. Set TBCLKSYNC = 1
6. Clear any spurious ePWM flags (including interrupt flags)
7. Enable ePWM interrupts
8. Enable global interrupts

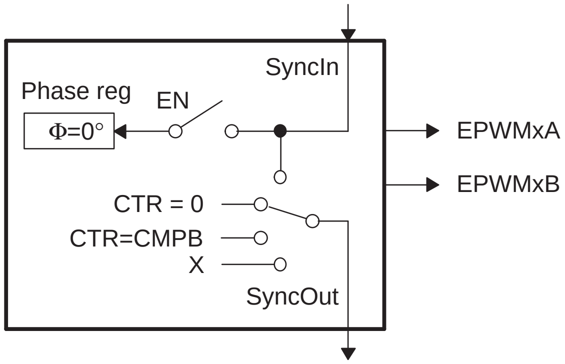
19.3 Application Examples

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.

19.3.1 Overview of Multiple Modules

Previously in this chapter, 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 19-51. 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 19-51. Simplified ePWM Module



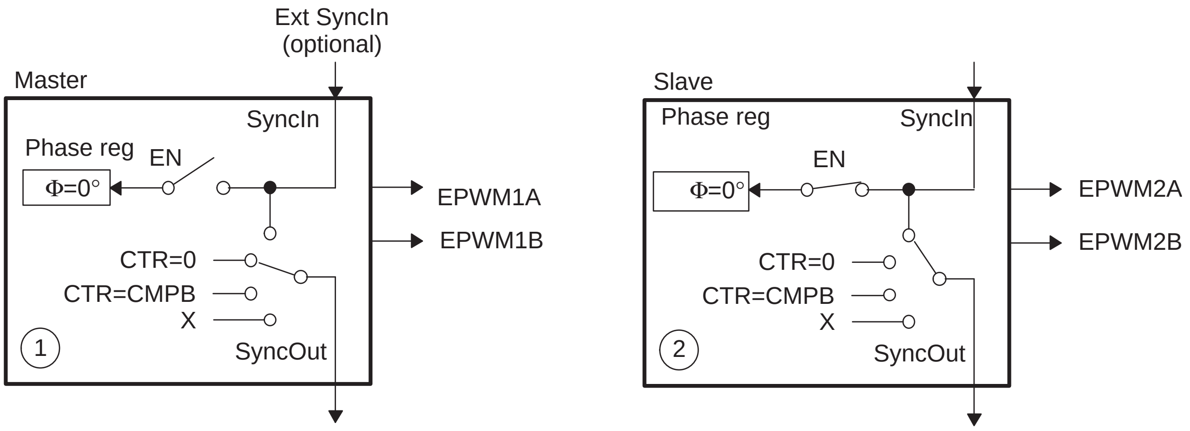
19.3.2 Key Configuration Capabilities

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, that is, via the enable switch. Although various combinations are possible, the two most common—master module and slave module modes—are shown in Figure 19-52.

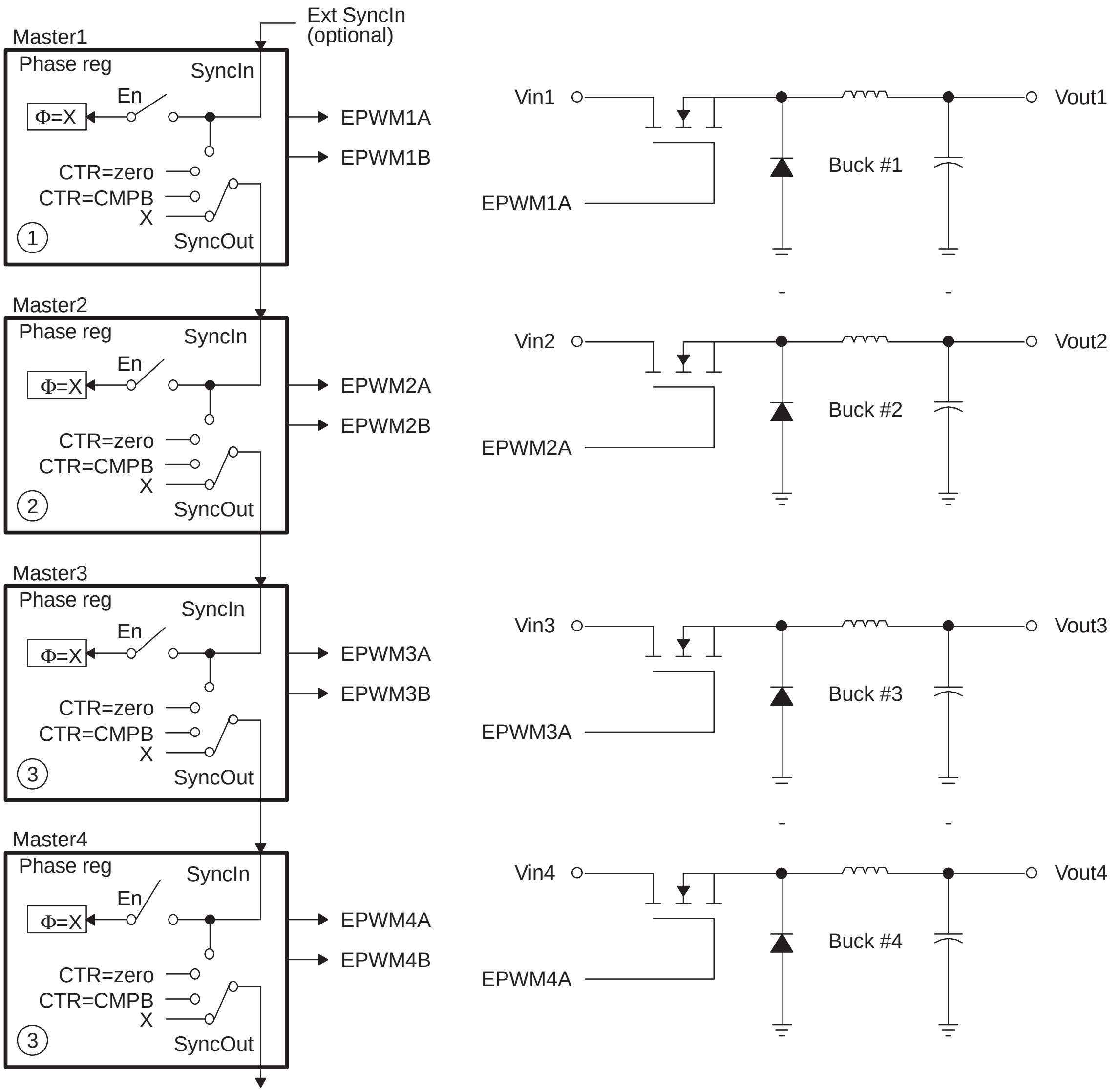
Figure 19-52. EPWM1 Configured as a Typical Master, EPWM2 Configured as a Slave



19.3.3 Controlling Multiple Buck Converters With Independent Frequencies

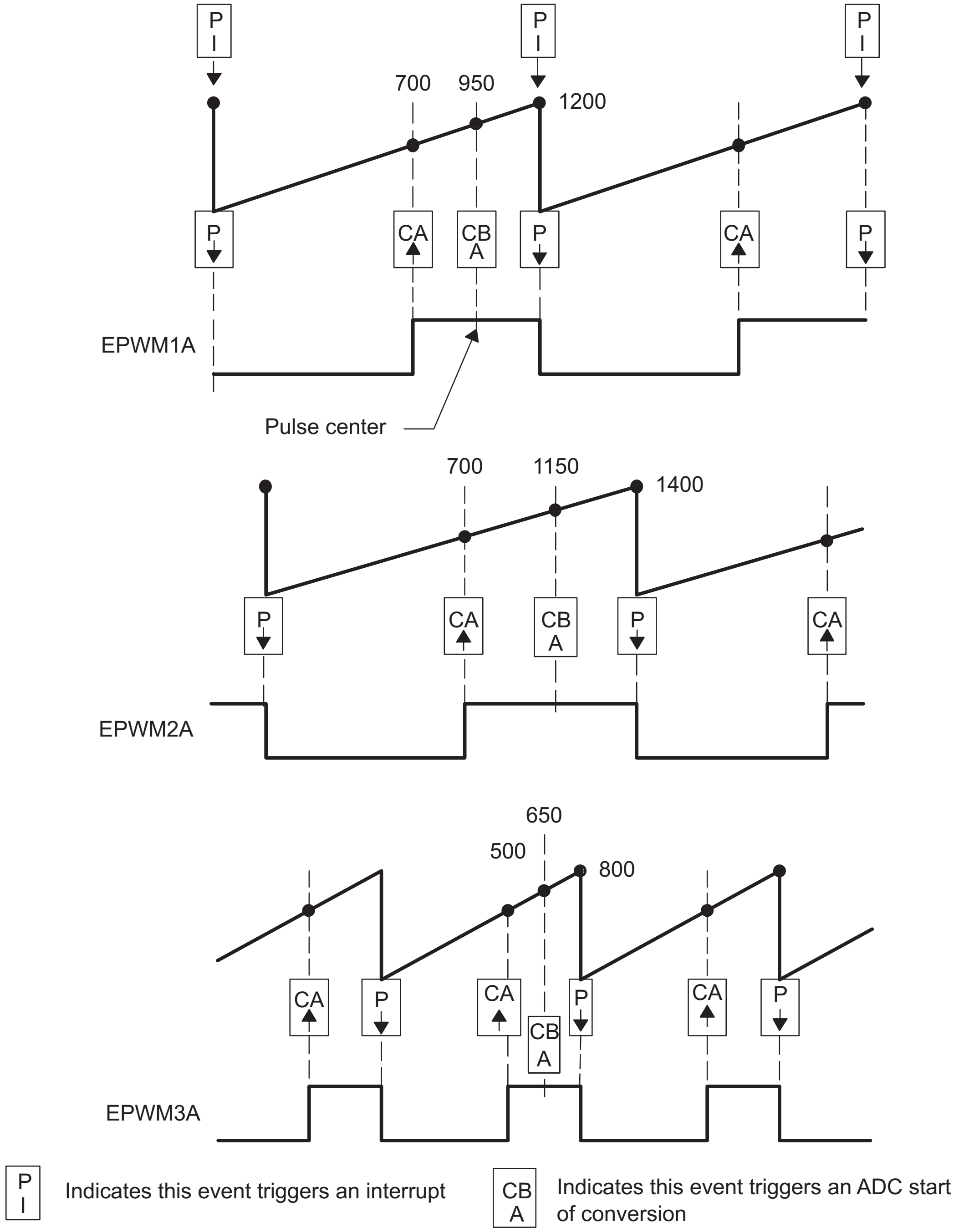
One of the simplest power converter topologies is the buck. A single ePWM module configured as a master can control two buck stages with the same PWM frequency. If independent frequency control is required for each buck converter, then one ePWM module must be allocated for each converter stage. Figure 19-53 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 19-54 shows the waveforms generated by the setup shown in Figure 19-53; note that only three waveforms are shown, although there are four stages.

Figure 19-53. Control of Four Buck Stages. Here $F_{PWM1} \neq F_{PWM2} \neq F_{PWM3} \neq F_{PWM4}$



NOTE: Θ = X indicates value in phase register is a "don't care"

Figure 19-54. Buck Waveforms for Figure 19-53 (Note: Only three bucks shown here)



Example 19-8. Configuration for Example in Figure 19-54

```
//=====================================================================
// (Note: code for only 3 modules shown)
// Initialization Time
//========================
// EPWM Module 1 config
   EPwm1Regs.TBPRD = 1200;                         // Period = 1201 TBCLK counts
   EPwm1Regs.TBPHS.half.TBPHS = 0;                 // Set Phase register to zero
   EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;      // Asymmetrical mode
   EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;         // Phase loading disabled
   EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
   EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
   EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
   EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
   EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;   // load on CTR=Zero
   EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;   // load on CTR=Zero
   EPwm1Regs.AQCTLA.bit.PRD = AQ_CLEAR;
   EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;
// EPWM Module 2 config
   EPwm2Regs.TBPRD = 1400;                         // Period = 1401 TBCLK counts
   EPwm2Regs.TBPHS.half.TBPHS = 0;                 // Set Phase register to zero
   EPwm2Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;      // Asymmetrical mode
   EPwm2Regs.TBCTL.bit.PHSEN = TB_DISABLE;         // Phase loading disabled
   EPwm2Regs.TBCTL.bit.PRDLD = TB_SHADOW;
   EPwm2Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
   EPwm2Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
   EPwm2Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
   EPwm2Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;   // load on CTR=Zero
   EPwm2Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;   // load on CTR=Zero
   EPwm2Regs.AQCTLA.bit.PRD = AQ_CLEAR;
   EPwm2Regs.AQCTLA.bit.CAU = AQ_SET;
// EPWM Module 3 config
   EPwm3Regs.TBPRD = 800;                           // Period = 801 TBCLK counts
   EPwm3Regs.TBPHS.half.TBPHS = 0;                  // Set Phase register to zero
   EPwm3Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;
   EPwm3Regs.TBCTL.bit.PHSEN = TB_DISABLE;          // Phase loading disabled
   EPwm3Regs.TBCTL.bit.PRDLD = TB_SHADOW;
   EPwm3Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
   EPwm3Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
   EPwm3Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
   EPwm3Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;    // load on CTR=Zero
   EPwm3Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;    // load on CTR=Zero
   EPwm3Regs.AQCTLA.bit.PRD = AQ_CLEAR;
   EPwm3Regs.AQCTLA.bit.CAU = AQ_SET;
//
// Run Time (Note: Example execution of one run-time instant)
//=========================================================
   EPwm1Regs.CMPA.half.CMPA = 700;                 // adjust duty for output EPWM1A
   EPwm2Regs.CMPA.half.CMPA = 700;                 // adjust duty for output EPWM2A
   EPwm3Regs.CMPA.half.CMPA = 500;                 // adjust duty for output EPWM3A
```

19.3.4 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 19-55 shows such a configuration; Figure 19-56 shows the waveforms generated by the configuration.

Figure 19-55. Control of Four Buck Stages. (Note: F_{PWM2} = N x F_{PWM1})

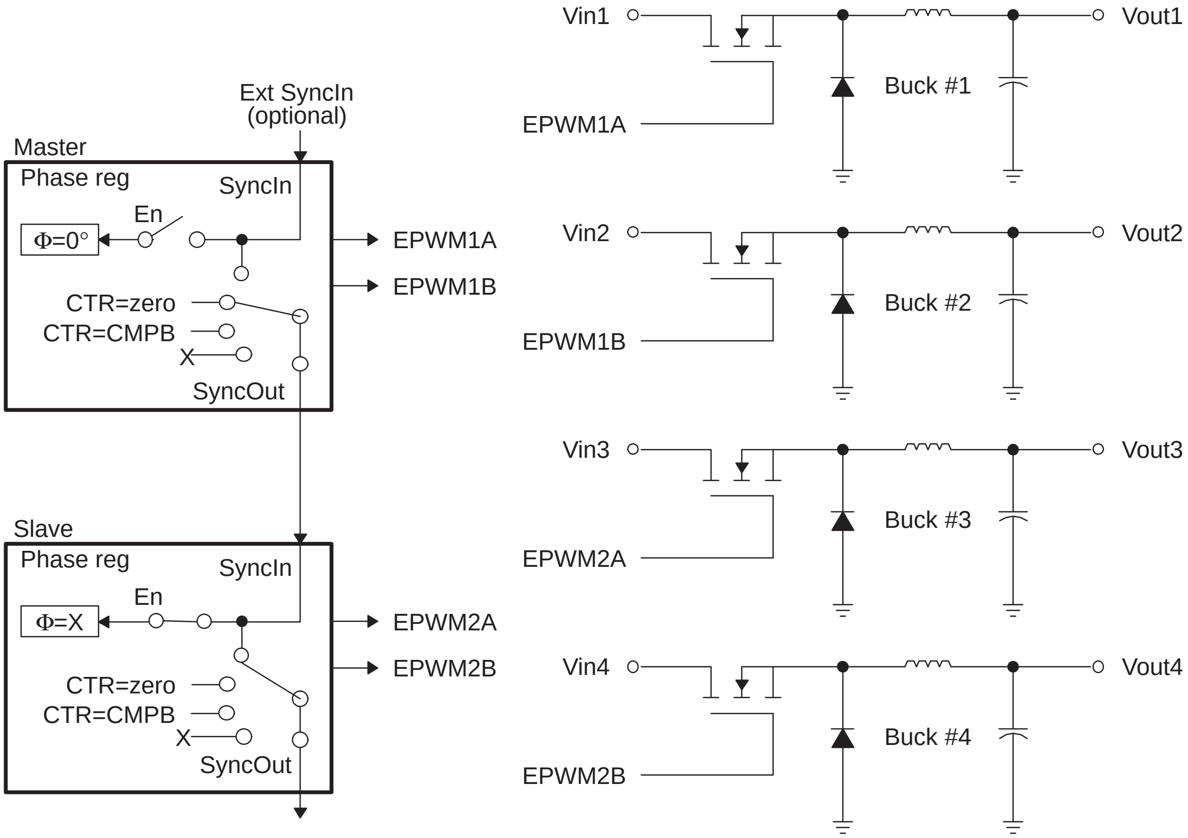
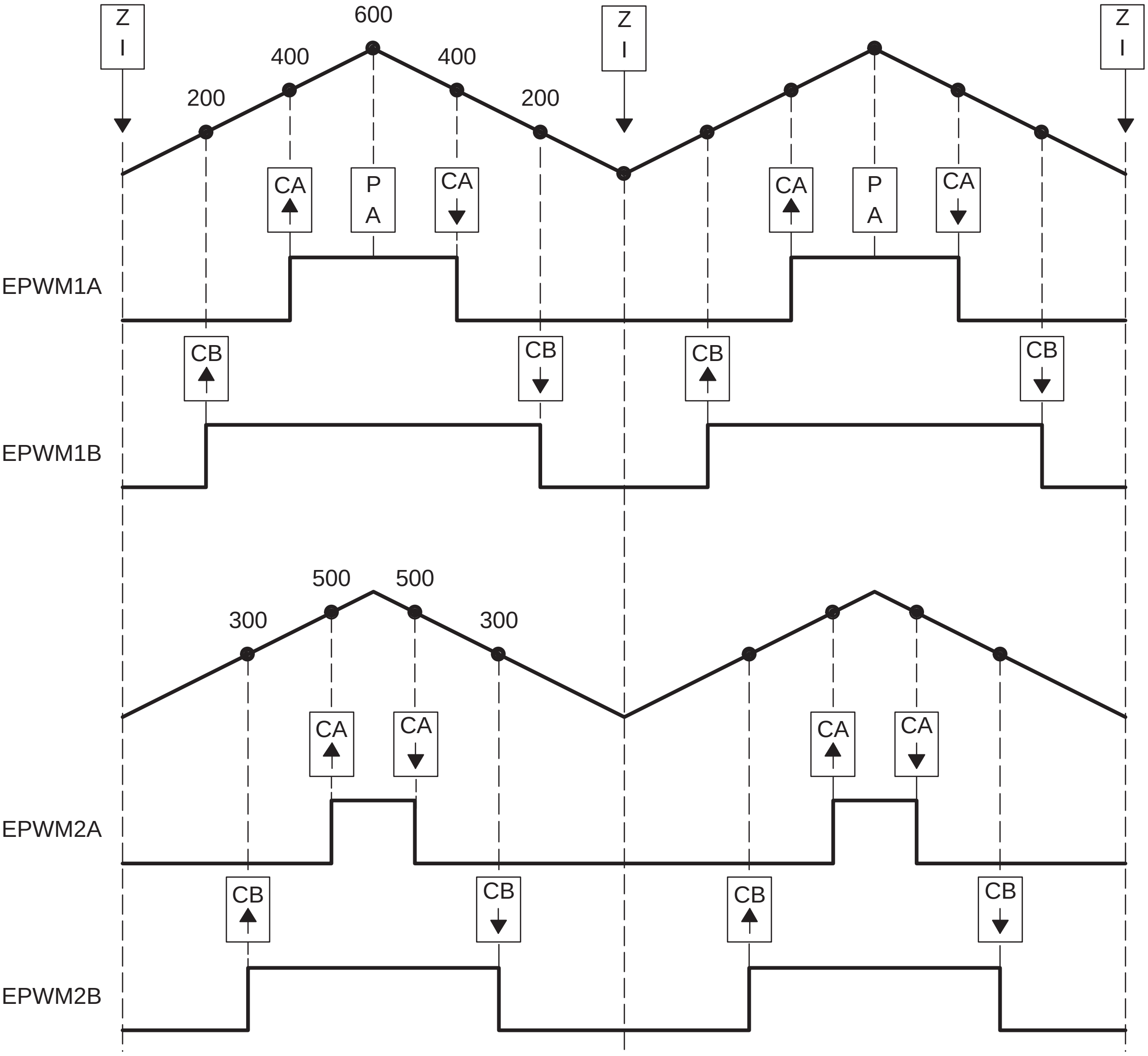


Figure 19-56. Buck Waveforms for Figure 19-55 (Note: $F_{PWM2} = F_{PWM1}$))



Example 19-9. Code Snippet for Configuration in Figure 19-55

```
//========================
// EPWM Module 1 config
     EPwm1Regs.TBPRD = 600;                          // Period = 1200 TBCLK counts
     EPwm1Regs.TBPHS.half.TBPHS = 0;                 // Set Phase register to zero
     EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN; // Symmetrical mode
     EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;         // Master module
     EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
     EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_CTR_ZERO;     // Sync down-stream module
     EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
     EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
     EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;   // load on CTR=Zero
     EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;   // load on CTR=Zero
     EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;              // set actions for EPWM1A
     EPwm1Regs.AQCTLA.bit.CAD = AQ_CLEAR;
     EPwm1Regs.AQCTLB.bit.CBU = AQ_SET;              // set actions for EPWM1B
     EPwm1Regs.AQCTLB.bit.CBD = AQ_CLEAR;
// EPWM Module 2 config
     EPwm2Regs.TBPRD = 600;                          // Period = 1200 TBCLK counts
     EPwm2Regs.TBPHS.half.TBPHS = 0;                 // Set Phase register to zero
     EPwm2Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN; // Symmetrical mode
     EPwm2Regs.TBCTL.bit.PHSEN = TB_ENABLE;          // Slave module
     EPwm2Regs.TBCTL.bit.PRDLD = TB_SHADOW;
     EPwm2Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_IN;      // sync flow-through
     EPwm2Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
     EPwm2Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
     EPwm2Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;   // load on CTR=Zero
     EPwm2Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;   // load on CTR=Zero
     EPwm2Regs.AQCTLA.bit.CAU = AQ_SET;              // set actions for EPWM2A
     EPwm2Regs.AQCTLA.bit.CAD = AQ_CLEAR;
     EPwm2Regs.AQCTLB.bit.CBU = AQ_SET;              // set actions for EPWM2B
     EPwm2Regs.AQCTLB.bit.CBD = AQ_CLEAR;
//
// Run Time (Note: Example execution of one run-time instance)
//===========================================================
    EPwm1Regs.CMPA.half.CMPA = 400;                // adjust duty for output EPWM1A
    EPwm1Regs.CMPB = 200;                          // adjust duty for output EPWM1B
    EPwm2Regs.CMPA.half.CMPA = 500;                // adjust duty for output EPWM2A
    EPwm2Regs.CMPB = 300;                          // adjust duty for output EPWM2B
```

19.3.5 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 19-57 shows control of two synchronized Half-H bridge stages where stage 2 can operate at integer multiple (N) frequencies of stage 1. Figure 19-58 shows the waveforms generated by the configuration shown in Figure 19-57.

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 19-57. Control of Two Half-H Bridge Stages (F_{PWM2} = N x F_{PWM1})

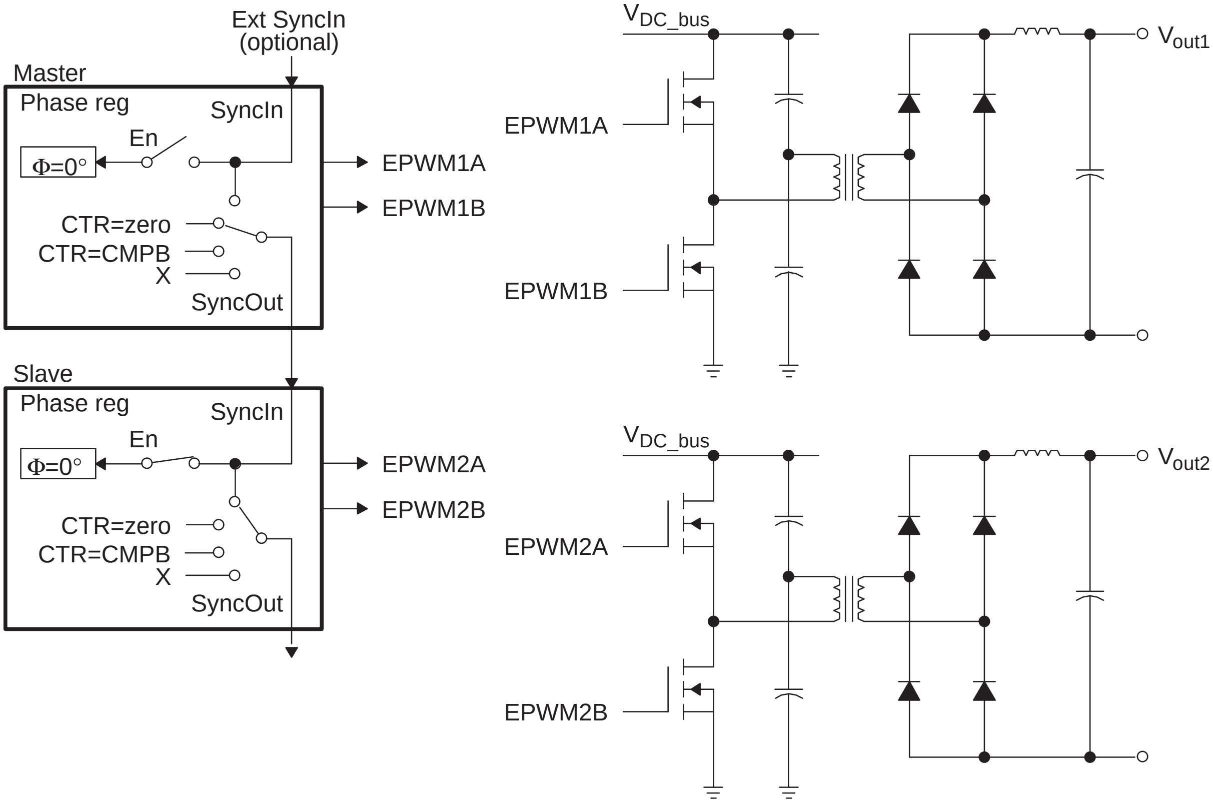
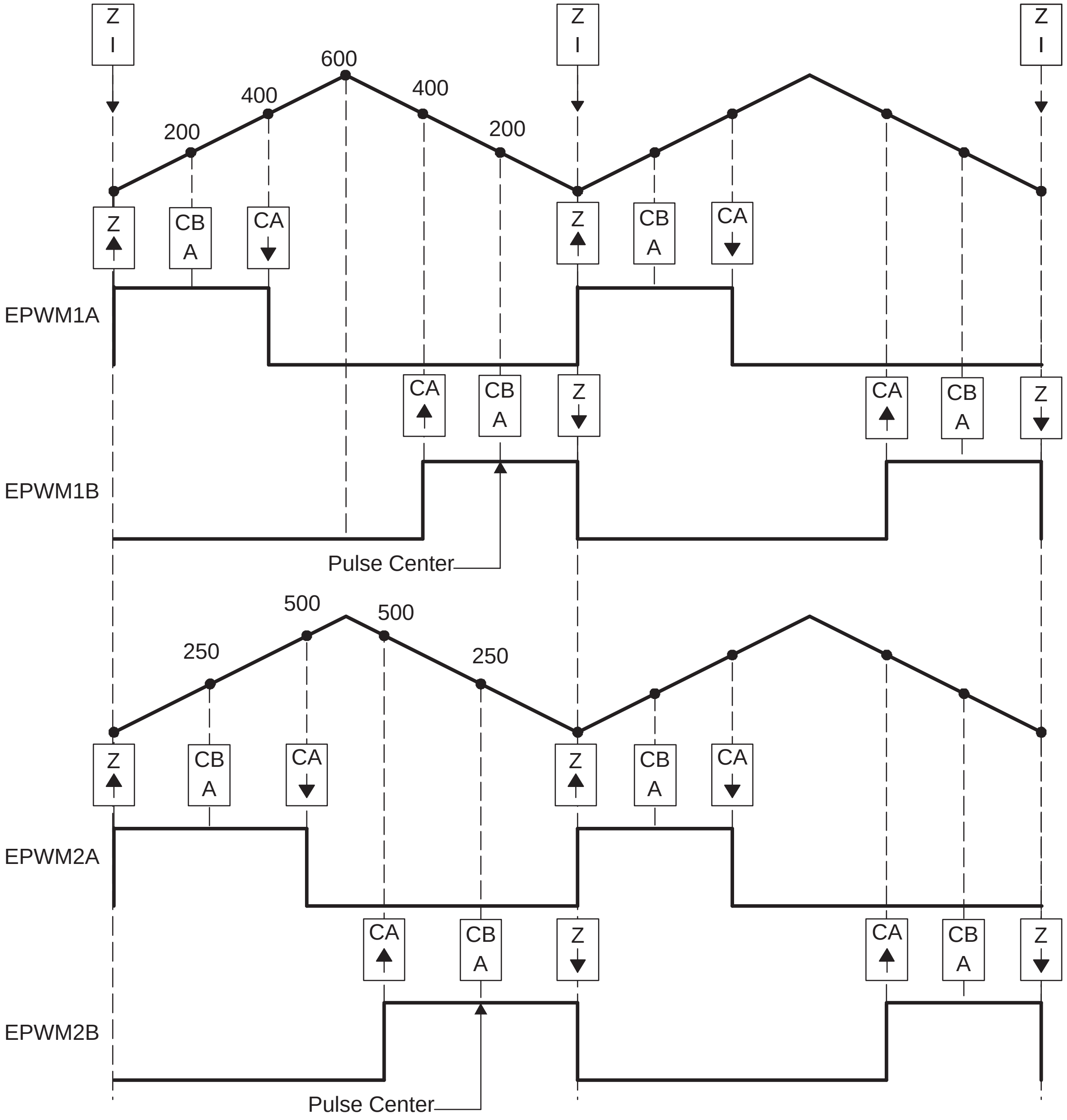


Figure 19-58. Half-H Bridge Waveforms for Figure 19-57 (Note: Here F_{PWM2} = F_{PWM1})



Example 19-10. Code Snippet for Configuration in Figure 19-57

```
//=====================================================================
// Config
//=====================================================================
// Initialization Time
//========================
// EPWM Module 1 config
     EPwm1Regs.TBPRD = 600;                         // Period = 1200 TBCLK counts
     EPwm1Regs.TBPHS.half.TBPHS = 0;                // Set Phase register to zero
     EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN; // Symmetrical mode
     EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;        // Master module
     EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
     EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_CTR_ZERO;    // Sync down-stream module
     EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
     EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
     EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;  // load on CTR=Zero
     EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;  // load on CTR=Zero
     EPwm1Regs.AQCTLA.bit.ZRO = AQ_SET;             // set actions for EPWM1A
     EPwm1Regs.AQCTLA.bit.CAU = AQ_CLEAR;
     EPwm1Regs.AQCTLB.bit.ZRO = AQ_CLEAR;           // set actions for EPWM1B
     EPwm1Regs.AQCTLB.bit.CAD = AQ_SET;
// EPWM Module 2 config
     EPwm2Regs.TBPRD = 600;                         // Period = 1200 TBCLK counts
     EPwm2Regs.TBPHS.half.TBPHS = 0;                // Set Phase register to zero
     EPwm2Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN; // Symmetrical mode
     EPwm2Regs.TBCTL.bit.PHSEN = TB_ENABLE;         // Slave module
     EPwm2Regs.TBCTL.bit.PRDLD = TB_SHADOW;
     EPwm2Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_IN;     // sync flow-through
     EPwm2Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
     EPwm2Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
     EPwm2Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;  // load on CTR=Zero
     EPwm2Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;  // load on CTR=Zero
     EPwm2Regs.AQCTLA.bit.ZRO = AQ_SET;             // set actions for EPWM1A
     EPwm2Regs.AQCTLA.bit.CAU = AQ_CLEAR;
     EPwm2Regs.AQCTLB.bit.ZRO = AQ_CLEAR;           // set actions for EPWM1B
     EPwm2Regs.AQCTLB.bit.CAD = AQ_SET;
//============================================================
     EPwm1Regs.CMPA.half.CMPA = 400;                // adjust duty for output EPWM1A & EPWM1B

     EPwm1Regs.CMPB = 200;                          // adjust point-in-time for ADCSOC trigger
     EPwm2Regs.CMPA.half.CMPA = 500;                // adjust duty for output EPWM2A & EPWM2B
     EPwm2Regs.CMPB = 250;                          // adjust point-in-time for ADCSOC trigger
```

19.3.6 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 19-59 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 19-59), 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 19-59. Control of Dual 3-Phase Inverter Stages as Is Commonly Used in Motor Control

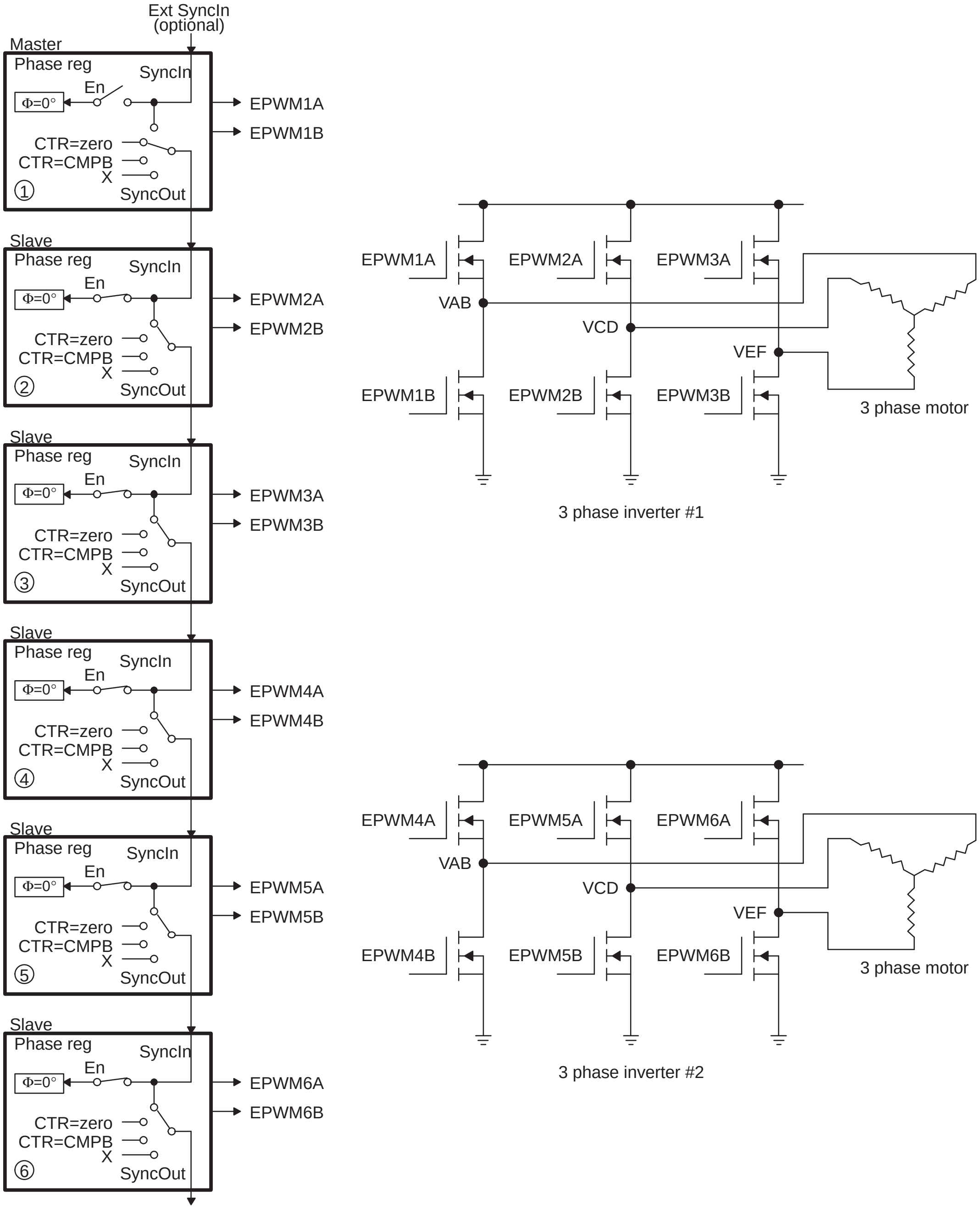
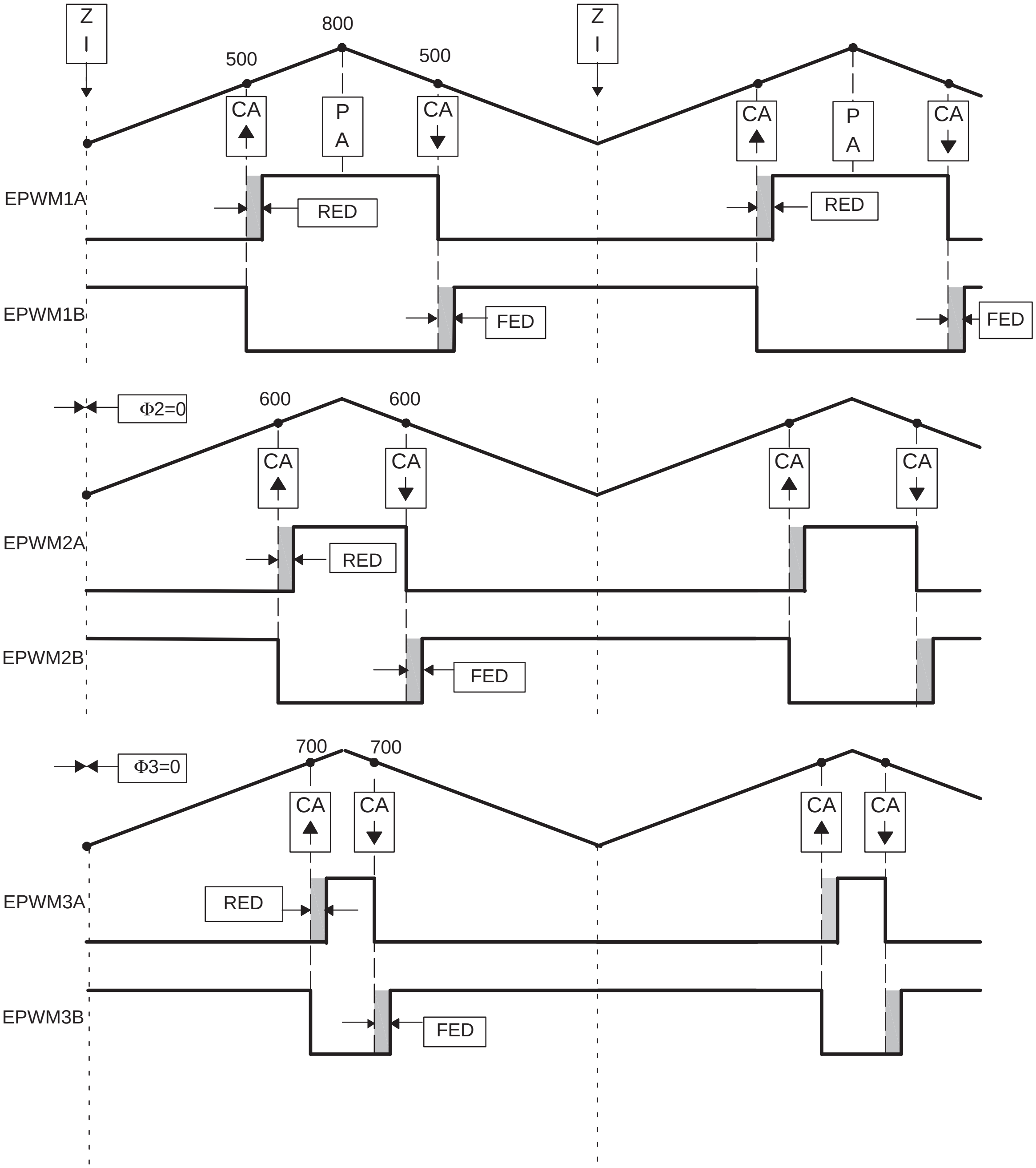


Figure 19-60. 3-Phase Inverter Waveforms for Figure 19-59 (Only One Inverter Shown)



19.3.7 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 applications that rely on phase relationship between stages for correct operation. As described in the TB module section, a PWM module can be configured to allow a SyncIn pulse to cause the TBPHS register to be loaded into the TBCTR register. To illustrate this concept, Figure 19-61 shows a master and slave module with a phase relationship of 120°, that is, the slave leads the master.

Figure 19-61. Configuring Two PWM Modules for Phase Control

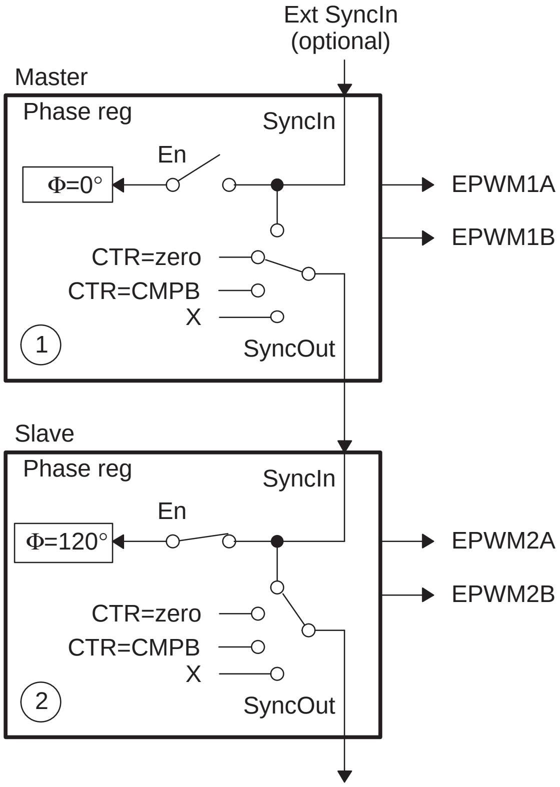
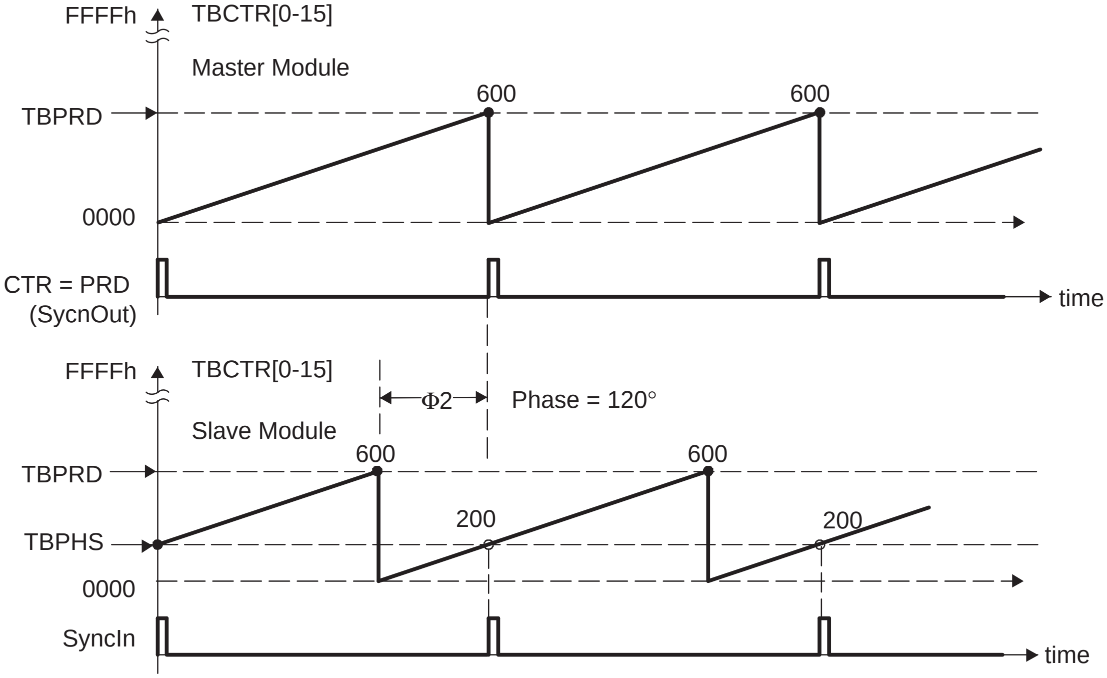


Figure 19-62 shows the associated timing waveforms for this configuration. Here, TBPRD = 600 for both master and slave. For the slave, TBPHS = 200 (that is, 200/600 X 360° = 120°). Whenever the master generates a SyncIn pulse (CTR = PRD), the value of TBPHS = 200 is loaded into the slave TBCTR register so the slave time-base is always leading the master's time-base by 120°.

Figure 19-62. Timing Waveforms Associated With Phase Control Between 2 Modules



19.4 EPWM_MODULE_CONTROL_AND_STATUS_REGISTER Registers

Table 19-22 lists the memory-mapped registers for the EPWM_MODULE_CONTROL_AND_STATUS_REGISTER. All register offset addresses not listed in Table 19-22 should be considered as reserved locations and the register contents should not be modified.

The base address for the control registers is FCF7 8C00h for ePWM1, FCF7 8D00h for ePWM2, FCF7 8E00h for ePWM3, FCF7 8F00h for ePWM4, FCF7 9000h for ePWM5, FCF7 9100h for ePWM6, and FCF7 9200h for ePWM7.

Table 19-22. EPWM_MODULE_CONTROL_AND_STATUS_REGISTER Registers

Offset	Acronym	Register Name	Section
0h	TBCTL	Time-Base Control Register	Section 19.4.1
2h	TBSTS	Time-Base Status Register	Section 19.4.2
6h	TBPHS	Time-Base Phase Register	Section 19.4.3
8h	TBCTR	Time-Base Counter Register	Section 19.4.4
Ah	TBPRD	Time-Base Period Register	Section 19.4.5
Eh	CMPCTL	Counter-Compare Control Register	Section 19.4.6
12h	CMPA	Counter-Compare A Register	Section 19.4.7
14h	CMPB	Counter-Compare B Register	Section 19.4.8
16h	AQCTLA	Action-Qualifier Control Register for Output A (EPWMxA)	Section 19.4.9
18h	AQCTLB	Action-Qualifier Control Register for Output B (EPWMxB)	Section 19.4.10
1Ah	AQSFRC	Action-Qualifier Software Force Register	Section 19.4.11
1Ch	AQCSFRC	Action-Qualifier Continuous S/W Force Register Set	Section 19.4.12
1Eh	DBCTL	Dead-Band Generator Control Register	Section 19.4.13
20h	DBRED	Dead-Band Generator Rising Edge Delay Count Register	Section 19.4.14
22h	DBFED	Dead-Band Generator Falling Edge Delay Count Register	Section 19.4.15
24h	TZSEL	Trip-Zone Select Register	Section 19.4.16
26h	TZDCSEL	Trip Zone Digital Compare Event Select Register	Section 19.4.17
28h	TZCTL	Trip-Zone Control Register	Section 19.4.18
2Ah	TZEINT	Trip-Zone Enable Interrupt Register	Section 19.4.19
2Ch	TZFLG	Trip-Zone Flag Register	Section 19.4.20
2Eh	TZCLR	Trip-Zone Clear Register	Section 19.4.21
30h	TZFRC	Trip-Zone Force Register	Section 19.4.22
32h	ETSEL	Event-Trigger Selection Register	Section 19.4.23
34h	ETPS	Event-Trigger Pre-Scale Register	Section 19.4.24
36h	ETFLG	Event-Trigger Flag Register	Section 19.4.25
38h	ETCLR	Event-Trigger Clear Register	Section 19.4.26
3Ah	ETFRC	Event-Trigger Force Register	Section 19.4.27
3Ch	PCCTL	PWM-Chopper Control Register	Section 19.4.28
60h	DCTRIPSEL	Digital Compare Trip Select Register	Section 19.4.29
62h	DCACTL	Digital Compare A Control Register	Section 19.4.30
64h	DCBCTL	Digital Compare B Control Register	Section 19.4.31
66h	DCFCTL	Digital Compare Filter Control Register	Section 19.4.32
68h	DCCAPCTL	Digital Compare Capture Control Register	Section 19.4.33
6Ah	DCFOFFSET	Digital Compare Filter Offset Register	Section 19.4.34
6Ch	DCFOFFSETCNT	Digital Compare Filter Offset Counter Register	Section 19.4.35
6Eh	DCFWINDOW	Digital Compare Filter Window Register	Section 19.4.36
70h	DCFWINDOWCNT	Digital Compare Filter Window Counter Register	Section 19.4.37
72h	DCCAP	Digital Compare Counter Capture Register	Section 19.4.38

Complex bit access types are encoded to fit into small table cells. Table 19-23 shows the codes that are used for access types in this section.

Table 19-23. EPWM_MODULE_CONTROL_AND_STATUS_REGISTER Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
W1C	1C W	1 to clear Write
Reset or Default Value		
-*n*		Value after reset or the default value

19.4.1 TBCTL Register (Offset = 0h) [reset = 83h]

TBCTL is shown in Figure 19-63 and described in Table 19-24.

Return to Summary Table.

Figure 19-63. TBCTL Register

15	14	13	12	11	10	9	8
FREE_SOFT		PHSDIR	CLKDIV			HSPCLKDIV	
R/W-0h		R/W-0h	R/W-0h			R/W-1h	

7	6	5	4	3	2	1	0
HSPCLKDIV	SWFSYNC	SYNCOSEL		PRDLD	PHSEN	CTRMODE	
R/W-1h	R/W-0h	R/W-0h		R/W-0h	R/W-0h	R/W-3h	

Table 19-24. TBCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-14	FREE_SOFT	R/W	0h	Emulation Mode Bits. These bits select the behavior of the ePWM time-base counter during emulation events. 0h = Stop after the next time-base counter increment or decrement. 1h = Stop when counter completes a whole cycle. In up-count mode, stop when the time-base counter = period (TBCTR = TBPRD). In down-count mode, stop when the time-base counter = 0x0000 (TBCTR = 0x0000). In up-down-count mode, stop when the time-base counter = 0x0000 (TBCTR = 0x0000). 2h = Free run 3h = Free run
13	PHSDIR	R/W	0h	Phase Direction Bit. This bit is only used when the time-base counter is configured in the up-down-count mode. The PHSDIR bit indicates the direction the time-base counter (TBCTR) counts 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. 0h = Count down after the synchronization event. 1h = Count up after the synchronization event.
12-10	CLKDIV	R/W	0h	Time-base Clock Prescale Bits. These bits determine part of the time-base clock prescale value. TBCLK = VCLK4 / (HSPCLKDIV x CLKDIV) 0h = /1 (default on reset) 1h = /2 2h = /4 3h = /8 4h = /16 5h = /32 6h = /64 7h = /128

Table 19-24. TBCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9-7	HSPCLKDIV	R/W	1h	High Speed Time-base Clock Prescale Bits. These bits determine part of the time-base clock prescale value. TBCLK = VCLK4 / (HSPCLKDIV x CLKDIV) 0h = /1 1h = /2 (default on reset) 2h = /4 3h = /6 4h = /8 5h = /10 6h = /12 7h = /14
6	SWFSYNC	R/W	0h	Software Forced Synchronization Pulse. This event is ORed with the EPWMxSYNCI input of the ePWM module. SWFSYNC is valid (operates) only when EPWMxSYNCI is selected by SYNCOSEL = 0. 0h = Writing a 0 has no effect and reads always return a 0. 1h = Writing a 1 forces a one-time synchronization pulse to be generated.
5-4	SYNCOSEL	R/W	0h	Synchronization Output Select. These bits select the source of the EPWMxSYNCO signal. 0h = EPWMxSYNC 1h = CTR = zero: Time-base counter equal to zero (TBCTR = 0x0000) 2h = CTR = CMPB : Time-base counter equal to counter-compare B (TBCTR = CMPB) 3h = Disable EPWMxSYNCO signal
3	PRDLD	R/W	0h	Active Period Register Load From Shadow Register Select. 0h = The period register (TBPRD) is loaded from its shadow register when the time-base counter, TBCTR, is equal to 0. A write or read to the TBPRD register accesses the shadow register. 1h = 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	R/W	0h	Counter Register Load From Phase Register Enable. 0h = Do not load the time-base counter (TBCTR) from the time-base phase register (TBPHS) 1h = 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, or when a digital compare sync event occurs.
1-0	CTRMODE	R/W	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 takes 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: 0h = Up-count mode 1h = Down-count mode 2h = Up-down-count mode 3h = Stop-freeze counter operation (default on reset)

19.4.2 TBSTS Register (Offset = 2h) [reset = 1h]

TBSTS is shown in Figure 19-64 and described in Table 19-25.

Return to Summary Table.

Figure 19-64. TBSTS Register

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="5">RESERVED</td><td>CTRMAX</td><td>SYNCI</td><td>CTRDIR</td></tr>
<tr><td colspan="5">R-0h</td><td>R-0h</td><td>R/W-0h</td><td>R-1h</td></tr>
</table>

Table 19-25. TBSTS Register Field Descriptions

Bit	Field	Type	Reset	Description
15-3	RESERVED	R	0h	Reserved
2	CTRMAX	R	0h	Time-Base Counter Max Latched Status Bit. 0h = Reading a 0 indicates the time-base counter never reached its maximum value. Writing a 0 has no effect. 1h = Reading a 1 on this bit indicates that the time-base counter reached the max value 0xFFFF. Writing a 1 to this bit clears the latched event.
1	SYNCI	R/W	0h	Input Synchronization Latched Status Bit. 0h = Writing a 0 has no effect. Reading a 0 indicates no external synchronization event has occurred. 1h = Reading a 1 on this bit indicates that an external synchronization event has occurred (EPWMxSYNCI). Writing a 1 to this bit clears the latched event.
0	CTRDIR	R	1h	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]. 0h = Time-Base Counter is currently counting down. 1h = Time-Base Counter is currently counting up.

19.4.3 TBPHS Register (Offset = 6h) [reset = 0h]

TBPHS is shown in Figure 19-65 and described in Table 19-26.

Return to Summary Table.

Figure 19-65. TBPHS Register

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">TBPHS</td></tr>
<tr><td colspan="16">R/W-0h</td></tr>
</table>

Table 19-26. TBPHS Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	TBPHS	R/W	0h	These bits set time-base counter phase of the selected ePWM relative to the time-base that is supplying the synchronization input signal. Valid values: 0-FFFFh. If TBCTL[PHSEN] = 0, then the synchronization event is ignored and the time-base counter is not loaded with the phase. If TBCTL[PHSEN] = 1, then the time-base counter (TBCTR) is loaded with the phase (TBPHS) when a synchronization event occurs. The synchronization event is initiated by the input synchronization signal (EPWMxSYNCI) or by a software forced synchronization.

19.4.4 TBCTR Register (Offset = 8h) [reset = 0h]

TBCTR is shown in Figure 19-66 and described in Table 19-27.

Return to Summary Table.

Figure 19-66. TBCTR Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBCTR															
R/W-0h															

Table 19-27. TBCTR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	TBCTR	R/W	0h	Reading these bits gives the current time-base counter value. Valid values: 0-FFFFh. 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.

19.4.5 TBPRD Register (Offset = Ah) [reset = 0h]

TBPRD is shown in Figure 19-67 and described in Table 19-28.

Return to Summary Table.

Figure 19-67. TBPRD Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBPRD															
R/W-0h															

Table 19-28. TBPRD Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	TBPRD	R/W	0h	These bits determine the period of the time-base counter. This sets the PWM frequency. Valid values: 0-FFFFh. 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 automatically goes to the shadow register. In this case, the active register is loaded from the shadow register when the time-base counter equals 0. If TBCTL[PRDLD] = 1, then the shadow is disabled and any write or read goes directly to the active register, that is the register actively controlling the hardware. The active and shadow registers share the same memory map address.

19.4.6 CMPCTL Register (Offset = Eh) [reset = 0h]

CMPCTL is shown in Figure 19-68 and described in Table 19-29.

Return to Summary Table.

Figure 19-68. CMPCTL Register

15	14	13	12	11	10	9	8
RESERVED						SHDWBFULL	SHDWAFULL
R-0h						R-0h	R-0h

7	6	5	4	3	2	1	0
RESERVED	SHDWBMODE	RESERVED	SHDWAMODE	LOADBMODE		LOADAMODE	
R-0h	R/W-0h	R-0h	R/W-0h	R/W-0h		R/W-0h	

Table 19-29. CMPCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9	SHDWBFULL	R	0h	Counter-compare B (CMPB) Shadow Register Full Status Flag. This bit self clears once a load-strobe occurs. 0h = CMPB shadow FIFO not full yet. 1h = Indicates the CMPB shadow FIFO is full, a CPU write overwrites current shadow value.
8	SHDWAFULL	R	0h	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 does not affect the flag. This bit self clears once a load-strobe occurs. 0h = CMPA shadow FIFO not full yet. 1h = Indicates the CMPA shadow FIFO is full, a CPU write overwrites the current shadow value.
7	RESERVED	R	0h	Reserved
6	SHDWBMODE	R/W	0h	Counter-compare B (CMPB) Register Operating Mode. 0h = Shadow mode. Operates as a double buffer. All writes via the CPU access the shadow register. 1h = 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	R	0h	Reserved
4	SHDWAMODE	R/W	0h	Counter-compare A (CMPA) Register Operating Mode. 0h = Shadow mode. Operates as a double buffer. All writes via the CPU access the shadow register. 1h = 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	R/W	0h	Active Counter-Compare B (CMPB) Load From Shadow Select Mode. This bit has no effect in immediate mode (CMPCTL[SHDWBMODE] = 1). 0h = Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000) 1h = Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD) 2h = Load on either CTR = Zero or CTR = PRD 3h = Freeze (no loads possible)

Table 19-29. CMPCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1-0	LOADAMODE	R/W	0h	Active Counter-Compare A (CMPA) Load From Shadow Select Mode. This bit has no effect in immediate mode (CMPCTL[SHDWAMODE] = 1). 0h = Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000) 1h = Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD) 2h = Load on either CTR = Zero or CTR = PRD 3h = Freeze (no loads possible)

19.4.7 CMPA Register (Offset = 12h) [reset = 0h]

CMPA is shown in Figure 19-69 and described in Table 19-30.

Return to Summary Table.

Figure 19-69. CMPA Register

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">CMPA</td></tr>
<tr><td colspan="16">R/W-0h</td></tr>
</table>

Table 19-30. CMPA Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	CMPA	R/W	0h	The value in the active CMPA register is continuously compared to the time-base counter (TBCTR). When the values are equal, the counter-compare module generates a "time-base counter equal to counter compare A" event. This event is sent to the action-qualifier where it is qualified and converted it into one or more actions. These actions are applied to either the EPWMxA or the EPWMxB output depending on the configuration of the AQCTLA and AQCTLB registers. The actions that are 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 automatically goes to the shadow register. In this case, the CMPCTL[LOADAMODE] bit field determines which event loads the active register from the shadow register. Before a write, the CMPCTL[SHDWAFULL] bit is 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 goes 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.

19.4.8 CMPB Register (Offset = 14h) [reset = 0h]

CMPB is shown in Figure 19-70 and described in Table 19-31.

Return to Summary Table.

Figure 19-70. CMPB Register

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">CMPB</td></tr>
<tr><td colspan="16">R/W-0h</td></tr>
</table>

Table 19-31. CMPB Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	CMPB	R/W	0h	The value in the active CMPB register is continuously compared to the time-base counter (TBCTR). When the values are equal, the counter-compare module generates a "time-base counter equal to counter compare B" event. This event is sent to the action-qualifier where it is qualified and converted it into one or more actions. These actions are applied to either the EPWMxA or the EPWMxB output depending on the configuration of the AQCTLA and AQCTLB registers. The actions that are 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 automatically goes to the shadow register. In this case, the CMPCTL[LOADBMODE] bit field determines which event loads the active register from the shadow register. Before a write, the CMPCTL[SHDWBFULL] bit is 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 goes 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.

19.4.9 AQCTLA Register (Offset = 16h) [reset = 0h]

AQCTLA is shown in Figure 19-71 and described in Table 19-32.

Return to Summary Table.

Figure 19-71. AQCTLA Register

15	14	13	12	11	10	9	8
RESERVED				CBD		CBU	
R-0h				R/W-0h		R/W-0h	

7	6	5	4	3	2	1	0
CAD		CAU		PRD		ZRO	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

Table 19-32. AQCTLA Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-10	CBD	R/W	0h	Action when the time-base counter equals the active CMPB register and the counter is decrementing. 0h = Do nothing (action is disabled). 1h = Clear: force EPWMxA output low. 2h = Set: force EPWMxA output high. 3h = Toggle EPWMxA output: low output signal is forced high and a high signal is forced low.
9-8	CBU	R/W	0h	Action when the counter equals the active CMPB register and the counter is incrementing. 0h = Do nothing (action is disabled). 1h = Clear: force EPWMxA output low. 2h = Set: force EPWMxA output high. 3h = Toggle EPWMxA output: low output signal is forced high and a high signal is forced low.
7-6	CAD	R/W	0h	Action when the counter equals the active CMPA register and the counter is decrementing. 0h = Do nothing (action is disabled). 1h = Clear: force EPWMxA output low. 2h = Set: force EPWMxA output high. 3h = Toggle EPWMxA output: low output signal is forced high and a high signal is forced low.
5-4	CAU	R/W	0h	Action when the counter equals the active CMPA register and the counter is incrementing. 0h = Do nothing (action is disabled). 1h = Clear: force EPWMxA output low. 2h = Set: force EPWMxA output high. 3h = Toggle EPWMxA output: low output signal is forced high and a high signal is forced low.
3-2	PRD	R/W	0h	Action when the counter equals the period. By definition, in count up-down mode when the counter equals period the direction is defined as 0 or counting down. 0h = Do nothing (action is disabled). 1h = Clear: force EPWMxA output low. 2h = Set: force EPWMxA output high. 3h = Toggle EPWMxA output: low output signal is forced high and a high signal is forced low.

Table 19-32. AQCTLA Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1-0	ZRO	R/W	0h	Action when counter equals zero. By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up. 0h = Do nothing (action is disabled). 1h = Clear: force EPWMxA output low. 2h = Set: force EPWMxA output high. 3h = Toggle EPWMxA output: low output signal is forced high and a high signal is forced low.

19.4.10 AQCTLB Register (Offset = 18h) [reset = 0h]

AQCTLB is shown in Figure 19-72 and described in Table 19-33.

Return to Summary Table.

Figure 19-72. AQCTLB Register

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="4">RESERVED</td><td colspan="2">CBD</td><td colspan="2">CBU</td></tr>
<tr><td colspan="4">R-0h</td><td colspan="2">R/W-0h</td><td colspan="2">R/W-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">CAD</td><td colspan="2">CAU</td><td colspan="2">PRD</td><td colspan="2">ZRO</td></tr>
<tr><td colspan="2">R/W-0h</td><td colspan="2">R/W-0h</td><td colspan="2">R/W-0h</td><td colspan="2">R/W-0h</td></tr>
</table>

Table 19-33. AQCTLB Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-10	CBD	R/W	0h	Action when the counter equals the active CMPB register and the counter is decrementing. 0h = Do nothing (action is disabled). 1h = Clear: force EPWMxB output low. 2h = Set: force EPWMxB output high. 3h = Toggle EPWMxB output: low output signal is forced high and a high signal is forced low.
9-8	CBU	R/W	0h	Action when the counter equals the active CMPB register and the counter is incrementing. 0h = Do nothing (action is disabled). 1h = Clear: force EPWMxB output low. 2h = Set: force EPWMxB output high. 3h = Toggle EPWMxB output: low output signal is forced high and a high signal is forced low.
7-6	CAD	R/W	0h	Action when the counter equals the active CMPA register and the counter is decrementing. 0h = Do nothing (action is disabled). 1h = Clear: force EPWMxB output low. 2h = Set: force EPWMxB output high. 3h = Toggle EPWMxB output: low output signal is forced high and a high signal is forced low.
5-4	CAU	R/W	0h	Action when the counter equals the active CMPA register and the counter is incrementing. 0h = Do nothing (action is disabled). 1h = Clear: force EPWMxB output low. 2h = Set: force EPWMxB output high. 3h = Toggle EPWMxB output: low output signal is forced high and a high signal is forced low.
3-2	PRD	R/W	0h	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. 0h = Do nothing (action is disabled). 1h = Clear: force EPWMxB output low. 2h = Set: force EPWMxB output high. 3h = Toggle EPWMxB output: low output signal is forced high and a high signal is forced low.

Table 19-33. AQCTLB Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1-0	ZRO	R/W	0h	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. 0h = Do nothing (action is disabled). 1h = Clear: force EPWMxB output low. 2h = Set: force EPWMxB output high. 3h = Toggle EPWMxB output: low output signal is forced high and a high signal is forced low.

19.4.11 AQSFRC Register (Offset = 1Ah) [reset = 0h]

AQSFRC is shown in Figure 19-73 and described in Table 19-34.

Return to Summary Table.

Figure 19-73. AQSFRC Register

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">RLDCSF</td><td>OTSFB</td><td colspan="2">ACTSFB</td><td>OTSFA</td><td colspan="2">ACTSFA</td></tr>
<tr><td colspan="2">R/W-0h</td><td>R/W-0h</td><td colspan="2">R/W-0h</td><td>R/W-0h</td><td colspan="2">R/W-0h</td></tr>
</table>

Table 19-34. AQSFRC Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-6	RLDCSF	R/W	0h	AQCSFRC Active Register Reload From Shadow Options. 0h = 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	R/W	0h	One-Time Software Forced Event on Output B. 0h = Writing a 0 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 is overridden by another subsequent event on output B. 1h = Initiates a single software forced event.
4-3	ACTSFB	R/W	0h	Action when One-Time Software Force B Is invoked. 0h = Does nothing (action is disabled) 1h = Clear (low) 2h = Set (high) 3h = Toggle (low to high, high to low). This action is not qualified by counter direction (CNT_dir).
2	OTSFA	R/W	0h	One-Time Software Forced Event on Output A. 0h = Writing a 0 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). 1h = Initiates a single software forced event.
1-0	ACTSFA	R/W	0h	Action When One-Time Software Force A Is Invoked. 0h = Does nothing (action is disabled) 1h = Clear (low) 2h = Set (high) 3h = Toggle (low to high, high to low). This action is not qualified by counter direction (CNT_dir).

19.4.12 AQCSFRC Register (Offset = 1Ch) [reset = 0h]

AQCSFRC is shown in Figure 19-74 and described in Table 19-35.

Return to Summary Table.

Figure 19-74. AQCSFRC Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RESERVED				CSFB		CSFA	
R-0h				R/W-0h		R/W-0h	

Table 19-35. AQCSFRC Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	Reserved
3-2	CSFB	R/W	0h	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]. 0h = Forcing is disabled and 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	R/W	0h	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. 0h = Forcing is disabled and 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.

19.4.13 DBCTL Register (Offset = 1Eh) [reset = 0h]

DBCTL is shown in Figure 19-75 and described in Table 19-36.

Return to Summary Table.

Figure 19-75. DBCTL Register

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td>HALFCYCLE</td><td colspan="7">RESERVED</td></tr>
<tr><td>R/W-0h</td><td colspan="7">R-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">RESERVED</td><td colspan="2">IN_MODE</td><td colspan="2">POLSEL</td><td colspan="2">OUT_MODE</td></tr>
<tr><td colspan="2">R-0h</td><td colspan="2">R/W-0h</td><td colspan="2">R/W-0h</td><td colspan="2">R/W-0h</td></tr>
</table>

Table 19-36. DBCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	HALFCYCLE	R/W	0h	Half Cycle Clocking Enable Bit. 0h = Full cycle clocking is enabled. The dead-band counters are clocked at the TBCLK rate. 1h = Half cycle clocking is enabled. The dead-band counters are clocked at TBCLK x 2.
14-6	RESERVED	R	0h	Reserved
5-4	IN_MODE	R/W	0h	Dead Band Input Mode Control. This bit selects the input source to the falling-edge and rising-edge delay. To produce classical dead-band waveforms, the default is EPWMxA In. 0h = 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	R/W	0h	Polarity Select Control. This bit selectively inverts one of the delayed signals before it is sent out of the dead-band submodule. The following descriptions correspond to classical upper and 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. 0h = 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.

Table 19-36. DBCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1-0	OUT_MODE	R/W	0h	Dead-band Output Mode Control. This bit enables or bypasses the dead-band generation for the falling-edge and rising-edge delay. 0h = 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 = The rising-edge delayed signal is seen on output EPWMxA. The input signal for the delay is determined by DBCTL[IN_MODE]. Disable falling-edge delay. The EPWMxB signal from the action-qualifier is passed straight through to the EPWMxB input of the PWM-chopper submodule. 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].

19.4.14 DBRED Register (Offset = 20h) [reset = 0h]

DBRED is shown in Figure 19-76 and described in Table 19-37.

Return to Summary Table.

Figure 19-76. DBRED Register

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="6">RESERVED</td><td colspan="2">DEL</td></tr>
<tr><td colspan="6">R-0h</td><td colspan="2">R/W-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">DEL</td></tr>
<tr><td colspan="8">R/W-0h</td></tr>
</table>

Table 19-37. DBRED Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9-0	DEL	R/W	0h	Rising Edge Delay Count. 10-bit counter.

19.4.15 DBFED Register (Offset = 22h) [reset = 0h]

DBFED is shown in Figure 19-77 and described in Table 19-38.

Return to Summary Table.

Figure 19-77. DBFED Register

15	14	13	12	11	10	9	8
RESERVED						DEL	
R-0h						R/W-0h	

7	6	5	4	3	2	1	0
DEL							
R/W-0h							

Table 19-38. DBFED Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9-0	DEL	R/W	0h	Falling Edge Delay Count. 10-bit counter.

19.4.16 TZSEL Register (Offset = 24h) [reset = 0h]

TZSEL is shown in Figure 19-78 and described in Table 19-39.

Return to Summary Table.

One-Shot (OSHT) Trip-zone enable/disable (bits 15-8). 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 is taken on the EPWMxA and EPWMxB outputs. The one-shot trip condition remains latched until the user clears the condition via the TZCLR register. Cycle-by-Cycle (CBC) Trip-zone enable/disable (bits 7-0). 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 is taken on the EPWMxA and EPWMxB outputs. A cycle-by-cycle trip condition is automatically cleared when the time-base counter reaches zero.

Figure 19-78. TZSEL Register

15	14	13	12	11	10	9	8
DCBEVT1	DCAEVT1	OSHT6	OSHT5	OSHT4	OSHT3	OSHT2	OSHT1
R-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
DCBEVT2	DCAEVT2	CBC6	CBC5	CBC4	CBC3	CBC2	CBC1
R-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

Table 19-39. TZSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	DCBEVT1	R	0h	Digital Compare Output B Event 1 Select. 0h = Disable DCBEVT1 as one-shot-trip source for this ePWM module. 1h = Enable DCBEVT1 as one-shot-trip source for this ePWM module.
14	DCAEVT1	R	0h	Digital Compare Output A Event 1 Select. 0h = Disable DCAEVT1 as one-shot-trip source for this ePWM module. 1h = Enable DCAEVT1 as one-shot-trip source for this ePWM module.
13	OSHT6	R/W	0h	Trip-zone 6 (/TZ6) Select. 0h = Disable /TZ6 as a one-shot trip source for this ePWM module. 1h = Enable /TZ6 as a one-shot trip source for this ePWM module.
12	OSHT5	R/W	0h	Trip-zone 5 (/TZ5) Select. 0h = Disable /TZ5 as a one-shot trip source for this ePWM module. 1h = Enable /TZ5 as a one-shot trip source for this ePWM module.
11	OSHT4	R/W	0h	Trip-zone 4 (/TZ4) Select. 0h = Disable /TZ4 as a one-shot trip source for this ePWM module. 1h = Enable /TZ4 as a one-shot trip source for this ePWM module.
10	OSHT3	R/W	0h	Trip-zone 3 (/TZ3) Select. 0h = Disable /TZ3 as a one-shot trip source for this ePWM module. 1h = Enable /TZ3 as a one-shot trip source for this ePWM module.
9	OSHT2	R/W	0h	Trip-zone 2 (/TZ2) Select. 0h = Disable /TZ2 as a one-shot trip source for this ePWM module. 1h = Enable /TZ2 as a one-shot trip source for this ePWM module.
8	OSHT1	R/W	0h	Trip-zone 1 (/TZ1) Select. 0h = Disable /TZ1 as a one-shot trip source for this ePWM module. 1h = Enable /TZ1 as a one-shot trip source for this ePWM module.
7	DCBEVT2	R	0h	Digital Compare Output B Event 2 Select. 0h = Disable DCBEVT2 as a CBC trip source for this ePWM module. 1h = Enable DCBEVT2 as a CBC trip source for this ePWM module.

Table 19-39. TZSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6	DCAEVT2	R	0h	Digital Compare Output A Event 2 Select. 0h = Disable DCAEVT2 as a CBC trip source for this ePWM module. 1h = Enable DCAEVT2 as a CBC trip source for this ePWM module.
5	CBC6	R/W	0h	Trip-zone 6 (/TZ6) Select. 0h = Disable /TZ6 as a CBC trip source for this ePWM module. 1h = Enable /TZ6 as a CBC trip source for this ePWM module.
4	CBC5	R/W	0h	Trip-zone 5 (/TZ5) Select. 0h = Disable /TZ5 as a CBC trip source for this ePWM module. 1h = Enable /TZ5 as a CBC trip source for this ePWM module.
3	CBC4	R/W	0h	Trip-zone 4 (/TZ4) Select. 0h = Disable /TZ4 as a CBC trip source for this ePWM module. 1h = Enable /TZ4 as a CBC trip source for this ePWM module.
2	CBC3	R/W	0h	Trip-zone 3 (/TZ3) Select. 0h = Disable /TZ3 as a CBC trip source for this ePWM module. 1h = Enable /TZ3 as a CBC trip source for this ePWM module.
1	CBC2	R/W	0h	Trip-zone 2 (/TZ2) Select. 0h = Disable /TZ2 as a CBC trip source for this ePWM module. 1h = Enable /TZ2 as a CBC trip source for this ePWM module.
0	CBC1	R/W	0h	Trip-zone 1 (/TZ1) Select. 0h = Disable /TZ1 as a CBC trip source for this ePWM module. 1h = Enable /TZ1 as a CBC trip source for this ePWM module.

19.4.17 TZDCSEL Register (Offset = 26h) [reset = 0h]

TZDCSEL is shown in Figure 19-79 and described in Table 19-40.

Return to Summary Table.

Figure 19-79. TZDCSEL Register

15	14	13	12	11	10	9	8
RESERVED				DCBEVT2			DCBEVT1
R-0h				R/W-0h			R/W-0h
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DCBEVT1		DCAEVT2			DCAEVT1		
R/W-0h		R/W-0h			R/W-0h		

Table 19-40. TZDCSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-9	DCBEVT2	R/W	0h	Digital Compare Output B Event 2 Selection. 0h = Event is disabled. 1h = DCBH = low, DCBL = don't care 2h = DCBH = high, DCBL = don't care 3h = DCBL = low, DCBH = don't care 4h = DCBL = high, DCBH = don't care 5h = DCBL = high, DCBH = low 6h = Reserved 7h = Reserved
8-6	DCBEVT1	R/W	0h	Digital Compare Output B Event 1 Selection. 0h = Event is disabled. 1h = DCBH = low, DCBL = don't care 2h = DCBH = high, DCBL = don't care 3h = DCBL = low, DCBH = don't care 4h = DCBL = high, DCBH = don't care 5h = DCBL = high, DCBH = low 6h = Reserved 7h = Reserved
5-3	DCAEVT2	R/W	0h	Digital Compare Output A Event 2 Selection. 0h = Event is disabled. 1h = DCAH = low, DCAL = don't care 2h = DCAH = high, DCAL = don't care 3h = DCAL = low, DCAH = don't care 4h = DCAL = high, DCAH = don't care 5h = DCAL = high, DCAH = low 6h = Reserved 7h = Reserved
2-0	DCAEVT1	R/W	0h	Digital Compare Output A Event 1 Selection. 0h = Event is disabled. 1h = DCAH = low, DCAL = don't care 2h = DCAH = high, DCAL = don't care 3h = DCAL = low, DCAH = don't care 4h = DCAL = high, DCAH = don't care 5h = DCAL = high, DCAH = low 6h = Reserved 7h = Reserved

19.4.18 TZCTL Register (Offset = 28h) [reset = 0h]

TZCTL is shown in Figure 19-80 and described in Table 19-41.

Return to Summary Table.

Figure 19-80. TZCTL Register

15	14	13	12	11	10	9	8
RESERVED				DCBEVT2		DCBEVT1	
R-0h				R/W-0h		R/W-0h	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DCAEVT2		DCAEVT1		TZB		TZA	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	

Table 19-41. TZCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Reserved
11-10	DCBEVT2	R/W	0h	Digital Compare Output B Event 2 Action On EPWMxB. 0h = High-impedance (EPWMxB = High-impedance state). 1h = Force EPWMxB to a high state. 2h = Force EPWMxB to a low state. 3h = Do nothing, trip action is disabled.
9-8	DCBEVT1	R/W	0h	Digital Compare Output B Event 1 Action On EPWMxB. 0h = High-impedance (EPWMxB = High-impedance state). 1h = Force EPWMxB to a high state. 2h = Force EPWMxB to a low state. 3h = Do nothing, trip action is disabled.
7-6	DCAEVT2	R/W	0h	Digital Compare Output A Event 2 Action On EPWMxA. 0h = High-impedance (EPWMxA = High-impedance state). 1h = Force EPWMxA to a high state. 2h = Force EPWMxA to a low state. 3h = Do nothing, trip action is disabled.
5-4	DCAEVT1	R/W	0h	Digital Compare Output A Event 1 Action On EPWMxA. 0h = High-impedance (EPWMxA = High-impedance state). 1h = Force EPWMxA to a high state. 2h = Force EPWMxA to a low state. 3h = Do nothing, trip action is disabled.
3-2	TZB	R/W	0h	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. 0h = 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	R/W	0h	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. 0h = 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.

19.4.19 TZEINT Register (Offset = 2Ah) [reset = 0h]

TZEINT is shown in Figure 19-81 and described in Table 19-42.

Return to Summary Table.

Figure 19-81. TZEINT Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED	DCBEVT2	DCBEVT1	DCAEVT2	DCAEVT1	OST	CBC	RESERVED
R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R-0h

Table 19-42. TZEINT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-7	RESERVED	R	0h	Reserved
6	DCBEVT2	R/W	0h	Digital Comparator Output B Event 2 Interrupt Enable. 0h = Disabled 1h = Enabled
5	DCBEVT1	R/W	0h	Digital Comparator Output B Event 1 Interrupt Enable. 0h = Disabled 1h = Enabled
4	DCAEVT2	R/W	0h	Digital Comparator Output A Event 2 Interrupt Enable. 0h = Disabled 1h = Enabled
3	DCAEVT1	R/W	0h	Digital Comparator Output A Event 1 Interrupt Enable. 0h = Disabled 1h = Enabled
2	OST	R/W	0h	Trip-zone One-Shot Interrupt Enable. 0h = Disable one-shot interrupt generation. 1h = Enable Interrupt generation. A one-shot trip event causes a EPWMx_TZINT VIM interrupt.
1	CBC	R/W	0h	Trip-zone Cycle-by-Cycle Interrupt Enable. 0h = Disable cycle-by-cycle interrupt generation. 1h = Enable interrupt generation. A cycle-by-cycle trip event causes an EPWMx_TZINT VIM interrupt.
0	RESERVED	R	0h	Reserved

19.4.20 TZFLG Register (Offset = 2Ch) [reset = 0h]

TZFLG is shown in Figure 19-82 and described in Table 19-43.

Return to Summary Table.

Figure 19-82. TZFLG Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							

7	6	5	4	3	2	1	0
RESERVED	DCBEVT2	DCBEVT1	DCAEVT2	DCAEVT1	OST	CBC	INT
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

Table 19-43. TZFLG Register Field Descriptions

Bit	Field	Type	Reset	Description
15-7	RESERVED	R	0h	Reserved
6	DCBEVT2	R	0h	Latched Status Flag for Digital Compare Output B Event 2. 0h = No trip event has occurred on DCBEVT2. 1h = A trip event has occurred for the event defined for DCBEVT2.
5	DCBEVT1	R	0h	Latched Status Flag for Digital Compare Output B Event 1. 0h = No trip event has occurred on DCBEVT1. 1h = A trip event has occurred for the event defined for DCBEVT1.
4	DCAEVT2	R	0h	Latched Status Flag for Digital Compare Output A Event 2. 0h = No trip event has occurred on DCAEVT2. 1h = A trip event has occurred for the event defined for DCAEVT2.
3	DCAEVT1	R	0h	Latched Status Flag for Digital Compare Output A Event 1. 0h = No trip event has occurred on DCAEVT1. 1h = A trip event has occurred for the event defined for DCAEVT1.
2	OST	R	0h	Latched Status Flag for A One-Shot Trip Event. This bit is cleared by writing the appropriate value to the TZCLR register. 0h = No one-shot trip event has occurred. 1h = A trip event has occurred on a pin selected as a one-shot trip source.
1	CBC	R	0h	Latched Status Flag for Cycle-By-Cycle Trip Event. This bit is cleared by writing the appropriate value to the TZCLR register. 0h = No cycle-by-cycle trip event has occurred. 1h = A trip event has occurred on a signal selected as a cycle-by-cycle trip source. The TZFLG.CBC bit remains 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 is immediately set again. The specified condition on the signal is automatically cleared when the ePWM time-base counter reaches zero (TBCTR = 0x0000), if the trip condition is no longer present. The condition on the signal is only cleared when the TBCTR = 0x0000, no matter where in the cycle the CBC flag is cleared.
0	INT	R	0h	Latched Trip Interrupt Status Flag. No further EPWMx_TZINT VIM interrupts are generated until this flag is cleared. If the interrupt flag is cleared when either CBC or OST is set, then another interrupt pulse is generated. Clearing all flag bits prevents further interrupts. This bit is cleared by writing the appropriate value to the TZCLR register. 0h = No interrupt is generated. 1h = An EPWMx_TZINT VIM interrupt is generated because of a trip condition.

19.4.21 TZCLR Register (Offset = 2Eh) [reset = 0h]

TZCLR is shown in Figure 19-83 and described in Table 19-44.

Return to Summary Table.

Figure 19-83. TZCLR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED	DCBEVT2	DCBEVT1	DCAEVT2	DCAEVT1	OST	CBC	INT
R-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W-0h	R/W-0h	R/W-0h

Table 19-44. TZCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-7	RESERVED	R	0h	Reserved
6	DCBEVT2	R/W1C	0h	Clear Flag for Digital Compare Output B Event 2. 0h = Writing 0 has no effect. This bit always reads back 0. 1h = Writing 1 clears the DCBEVT2 event trip condition.
5	DCBEVT1	R/W1C	0h	Clear Flag for Digital Compare Output B Event 1. 0h = Writing 0 has no effect. This bit always reads back 0. 1h = Writing 1 clears the DCBEVT1 event trip condition.
4	DCAEVT2	R/W1C	0h	Clear Flag for Digital Compare Output A Event 2. 0h = Writing 0 has no effect. This bit always reads back 0. 1h = Writing 1 clears the DCAEVT2 event trip condition.
3	DCAEVT1	R/W1C	0h	Clear Flag for Digital Compare Output A Event 1. 0h = Writing 0 has no effect. This bit always reads back 0. 1h = Writing 1 clears the DCAEVT1 event trip condition.
2	OST	R/W	0h	Clear Flag for One-Shot Trip (OST) Latch. 0h = Has no effect. Always reads back a 0. 1h = Writing 1 clears the Trip (set) condition.
1	CBC	R/W	0h	Clear Flag for Cycle-By-Cycle (CBC) Trip Latch. 0h = Has no effect. Always reads back a 0. 1h = Writing 1 clears the Trip (set) condition.
0	INT	R/W	0h	Global Interrupt Clear Flag. NOTE: No further EPWMx_TZINT VIM interrupts are 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 is generated. Clearing all flag bits prevents further interrupts. 0h = Has no effect. Always reads back a 0. 1h = Writing 1 clears the trip-interrupt flag for this ePWM module, TZFLG.INT.

19.4.22 TZFRC Register (Offset = 30h) [reset = 0h]

TZFRC is shown in Figure 19-84 and described in Table 19-45.

Return to Summary Table.

Figure 19-84. TZFRC Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RESERVED	DCBEVT2	DCBEVT1	DCAEVT2	DCAEVT1	OST	CBC	RESERVED
R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R-0h

Table 19-45. TZFRC Register Field Descriptions

Bit	Field	Type	Reset	Description
15-7	RESERVED	R	0h	Reserved
6	DCBEVT2	R/W	0h	Force Flag for Digital Compare Output B Event 2. 0h = Writing 0 has no effect. This bit always reads back 0. 1h = Writing 1 forces the DCBEVT2 event trip condition and sets the TZFLG[DCBEVT2] bit.
5	DCBEVT1	R/W	0h	Force Flag for Digital Compare Output B Event 1. 0h = Writing 0 has no effect. This bit always reads back 0. 1h = Writing 1 forces the DCBEVT1 event trip condition and sets the TZFLG[DCBEVT1] bit.
4	DCAEVT2	R/W	0h	Force Flag for Digital Compare Output A Event 2. 0h = Writing 0 has no effect. This bit always reads back 0. 1h = Writing 1 forces the DCAEVT2 event trip condition and sets the TZFLG[DCAEVT2] bit.
3	DCAEVT1	R/W	0h	Force Flag for Digital Compare Output A Event 1. 0h = Writing 0 has no effect. This bit always reads back 0. 1h = Writing 1 forces the DCAEVT1 event trip condition and sets the TZFLG[DCAEVT1] bit.
2	OST	R/W	0h	Force a One-Shot Trip Event via Software. 0h = Writing of 0 is ignored. Always reads back a 0. 1h = Writing 1 forces a one-shot trip event and sets the TZFLG[OST] bit.
1	CBC	R/W	0h	Force a Cycle-by-Cycle Trip Event via Software. 0h = Writing of 0 is ignored. Always reads back a 0. 1h = Writing 1 forces a cycle-by-cycle trip event and sets the TZFLG[CBC] bit.
0	RESERVED	R	0h	Reserved

19.4.23 ETSEL Register (Offset = 32h) [reset = 0h]

ETSEL is shown in Figure 19-85 and described in Table 19-46.

Return to Summary Table.

Figure 19-85. ETSEL Register

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td>SOCBEN</td><td colspan="3">SOCBSEL</td><td>SOCAEN</td><td colspan="3">SOCASEL</td></tr>
<tr><td>R/W-0h</td><td colspan="3">R/W-0h</td><td>R/W-0h</td><td colspan="3">R/W-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="4">RESERVED</td><td>INTEN</td><td colspan="3">INTSEL</td></tr>
<tr><td colspan="4">R-0h</td><td>R/W-0h</td><td colspan="3">R/W-0h</td></tr>
</table>

Table 19-46. ETSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	SOCBEN	R/W	0h	Enable the ADC Start of Conversion B (EPWMxSOCB) Pulse. 0h = Disable EPWMxSOCB. 1h = Enable EPWMxSOCB pulse.
14-12	SOCBSEL	R/W	0h	EPWMxSOCB Selection Options. These bits determine when a EPWMxSOCB pulse is generated. 0h = Enable DCBEVT1.soc event. 1h = Enable event time-base counter equal to zero. (TBCTR = 0x0000). 2h = Enable event time-base counter equal to period (TBCTR = TBPRD). 3h = Enable event time-base counter equal to zero or period (TBCTR = 0x0000 or TBCTR = TBPRD). This mode is useful in up-down count mode. 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.
11	SOCAEN	R/W	0h	Enable the ADC Start of Conversion A (EPWMxSOCA) Pulse. 0h = Disable EPWMxSOCA. 1h = Enable EPWMxSOCA pulse.
10-8	SOCASEL	R/W	0h	EPWMxSOCA Selection Options. These bits determine when a EPWMxSOCA pulse is generated. 0h = Enable DCAEVT1.soc event. 1h = Enable event time-base counter equal to zero. (TBCTR = 0x0000). 2h = Enable event time-base counter equal to period (TBCTR = TBPRD). 3h = Enable event time-base counter equal to zero or period (TBCTR = 0x0000 or TBCTR = TBPRD). This mode is useful in up-down count mode. 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.
7-4	RESERVED	R	0h	Reserved

Table 19-46. ETSEL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	INTEN	R/W	0h	Enable ePWM Interrupt (EPWMx_INT) Generation. 0h = Disable EPWMx_INT generation. 1h = Enable EPWMx_INT generation.
2-0	INTSEL	R/W	0h	ePWM Interrupt (EPWMx_INT) Selection Options. 0h = Reserved 1h = Enable event time-base counter equal to zero. (TBCTR = 0x0000). 2h = Enable event time-base counter equal to period (TBCTR = TBPRD). 3h = Enable event time-base counter equal to zero or period (TBCTR = 0x0000 or TBCTR = TBPRD). This mode is useful in up-down count mode. 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.

19.4.24 ETPS Register (Offset = 34h) [reset = 0h]

ETPS is shown in Figure 19-86 and described in Table 19-47.

Return to Summary Table.

Figure 19-86. ETPS Register

15	14	13	12	11	10	9	8
SOCBCNT		SOCBPRD		SOCACNT		SOCAPRD	
R-0h		R/W-0h		R-0h		R/W-0h	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RESERVED				INTCNT		INTPRD	
R-0h				R-0h		R/W-0h	

Table 19-47. ETPS Register Field Descriptions

Bit	Field	Type	Reset	Description
15-14	SOCBCNT	R	0h	ePWM ADC Start-of-Conversion B Event (EPWMxSOCB) Counter Register. These bits indicate how many selected ETSEL[SOCBSEL] events have occurred. 0h = No events have occurred. 1h = 1 event has occurred. 2h = 2 events have occurred. 3h = 3 events have occurred.
13-12	SOCBPRD	R/W	0h	ePWM ADC Start-of-Conversion B Event (EPWMxSOCB) Period Select. These bits determine how many selected ETSEL[SOCBSEL] events need to occur before an EPWMxSOCB pulse is generated. To be generated, the pulse must be enabled (ETSEL[SOCBEN] = 1). The SOCB pulse is generated even if the status flag is set from a previous start of conversion (ETFLG[SOCB] = 1). Once the SOCB pulse is generated, the ETPS[SOCBCNT] bits is automatically cleared. 0h = Disable the SOCB event counter. No EPWMxSOCB pulse is generated. 1h = Generate the EPWMxSOCB pulse on the first event: ETPS[SOCBCNT] = 0,1. 2h = Generate the EPWMxSOCB pulse on the second event: ETPS[SOCBCNT] = 1,0. 3h = Generate the EPWMxSOCB pulse on the third event: ETPS[SOCBCNT] = 1,1.
11-10	SOCACNT	R	0h	ePWM ADC Start-of-Conversion A Event (EPWMxSOCA) Counter Register. These bits indicate how many selected ETSEL[SOCASEL] events have occurred. 0h = No events have occurred. 1h = 1 event has occurred. 2h = 2 events have occurred. 3h = 3 events have occurred.

Table 19-47. ETPS Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9-8	SOCAPRD	R/W	0h	ePWM ADC Start-of-Conversion A Event (EPWMxSOCA) Period Select. These bits determine how many selected ETSEL[SOCASEL] events need to occur before an EPWMxSOCA pulse is generated. To be generated, the pulse must be enabled (ETSEL[SOCAEN] = 1). The SOCA pulse is generated even if the status flag is set from a previous start of conversion (ETFLG[SOCA] = 1). Once the SOCA pulse is generated, the ETPS[SOCACNT] bits are automatically cleared. 0h = Disable the SOCA event counter. No EPWMxSOCA pulse is generated. 1h = Generate the EPWMxSOCA pulse on the first event: ETPS[SOCACNT] = 0,1. 2h = Generate the EPWMxSOCA pulse on the second event: ETPS[SOCACNT] = 1,0. 3h = Generate the EPWMxSOCA pulse on the third event: ETPS[SOCACNT] = 1,1.
7-4	RESERVED	R	0h	Reserved
3-2	INTCNT	R	0h	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 stops counting events when it reaches the period value ETPS[INTCNT] = ETPS[INTPRD]. 0h = No events have occurred. 1h = 1 event has occurred. 2h = 2 events have occurred. 3h = 3 events have occurred.
1-0	INTPRD	R/W	0h	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 is 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 are automatically cleared. Writing a INTPRD value that is the same as the current counter value triggers an interrupt if it is enabled and the status flag is clear. Writing a INTPRD value that is less than the current counter value results 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. 0h = Disable the interrupt event counter. No interrupt is 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).

19.4.25 ETFLG Register (Offset = 36h) [reset = 0h]

ETFLG is shown in Figure 19-87 and described in Table 19-48.

Return to Summary Table.

Figure 19-87. ETFLG Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				SOCB	SOCA	RESERVED	INT
R-0h				R-0h	R-0h	R-0h	R-0h

Table 19-48. ETFLG Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	Reserved
3	SOCB	R	0h	Latched ePWM ADC Start-of-Conversion B (EPWMxSOCB) Status Flag. 0h = No EPWMxSOCB event occurred. 1h = A start of conversion pulse is generated on EPWMxSOCB. The EPWMxSOCB output continues to be generated even if the flag bit is set.
2	SOCA	R	0h	Latched ePWM ADC Start-of-Conversion A (EPWMxSOCA) Status Flag. Unlike the ETFLG[INT] flag, the EPWMxSOCA output continues to pulse even if the flag bit is set. 0h = No event occurred. 1h = A start of conversion pulse is generated on EPWMxSOCA. The EPWMxSOCA output continues to be generated even if the flag bit is set.
1	RESERVED	R	0h	Reserved
0	INT	R	0h	Latched ePWM Interrupt (EPWMx_INT) Status Flag. 0h = No event occurred. 1h = An ePWMx interrupt (EWPMx_INT) is generated. No further interrupts are 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 is not generated until after the ETFLG[INT] bit is cleared.

19.4.26 ETCLR Register (Offset = 38h) [reset = 0h]

ETCLR is shown in Figure 19-88 and described in Table 19-49.

Return to Summary Table.

Figure 19-88. ETCLR Register

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="4">RESERVED</td><td>SOCB</td><td>SOCA</td><td>RESERVED</td><td>INT</td></tr>
<tr><td colspan="4">R-0h</td><td>R/W-0h</td><td>R/W-0h</td><td>R-0h</td><td>R/W-0h</td></tr>
</table>

Table 19-49. ETCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	Reserved
3	SOCB	R/W	0h	ePWM ADC Start-of-Conversion B (EPWMxSOCB) Flag Clear Bit. 0h = Writing a 0 has no effect. Always reads back a 0. 1h = Clears the ETFLG[SOCB] flag bit.
2	SOCA	R/W	0h	ePWM ADC Start-of-Conversion A (EPWMxSOCA) Flag Clear Bit. 0h = Writing a 0 has no effect. Always reads back a 0. 1h = Clears the ETFLG[SOCA] flag bit.
1	RESERVED	R	0h	Reserved
0	INT	R/W	0h	ePWM Interrupt (EPWMx_INT) Flag Clear Bit. 0h = Writing a 0 has no effect. Always reads back a 0. 1h = Clears the ETFLG[INT] flag bit and enables further interrupts pulses to be generated.

19.4.27 ETFRC Register (Offset = 3Ah) [reset = 0h]

ETFRC is shown in Figure 19-89 and described in Table 19-50.

Return to Summary Table.

Figure 19-89. ETFRC Register

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="4">RESERVED</td><td>SOCB</td><td>SOCA</td><td>RESERVED</td><td>INT</td></tr>
<tr><td colspan="4">R-0h</td><td>R/W-0h</td><td>R/W-0h</td><td>R-0h</td><td>R/W-0h</td></tr>
</table>

Table 19-50. ETFRC Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	Reserved
3	SOCB	R/W	0h	SOCB Force Bit. The SOCB pulse is only generated if the event is enabled in the ETSEL register. The ETFLG[SOCB] flag bit is set regardless. 0h = Writing 0 to this bit is ignored. Always reads back a 0. 1h = Generates a pulse on EPWMxSOCB and sets the SOCBFLG bit. This bit is used for test purposes.
2	SOCA	R/W	0h	SOCA Force Bit. The SOCA pulse is only generated if the event is enabled in the ETSEL register. The ETFLG[SOCA] flag bit is set regardless. 0h = Writing 0 to this bit is ignored. Always reads back a 0. 1h = Generates a pulse on EPWMxSOCA and set the SOCAFLG bit. This bit is used for test purposes.
1	RESERVED	R	0h	Reserved
0	INT	R/W	0h	INT Force Bit. The interrupt is only generated if the event is enabled in the ETSEL register. The INT flag bit is set regardless. 0h = Writing 0 to this bit is ignored. Always reads back a 0. 1h = Generates an interrupt on /EPWMxINT and set the INT flag bit. This bit is used for test purposes.

19.4.28 PCCTL Register (Offset = 3Ch) [reset = 0h]

PCCTL is shown in Figure 19-90 and described in Table 19-51.

Return to Summary Table.

Figure 19-90. PCCTL Register

15	14	13	12	11	10	9	8
RESERVED					CHPDUTY		
R-0h					R/W-0h		

7	6	5	4	3	2	1	0
CHPFREQ			OSHTWTH				CHPEN
R/W-0h			R/W-0h				R/W-0h

Table 19-51. PCCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-11	RESERVED	R	0h	Reserved
10-8	CHPDUTY	R/W	0h	Chopping Clock Duty Cycle. 0h = 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	R/W	0h	Chopping Clock Frequency. 0h = Divide by 1 (no prescale, = 12.5 MHz at 100 MHz VCLK4) 1h = Divide by 2 (6.25 MHz at 100 MHz VCLK4) 2h = Divide by 3 (4.16 MHz at 100 MHz VCLK4) 3h = Divide by 4 (3.12 MHz at 100 MHz VCLK4) 4h = Divide by 5 (2.50 MHz at 100 MHz VCLK4) 5h = Divide by 6 (2.08 MHz at 100 MHz VCLK4) 6h = Divide by 7 (1.78 MHz at 100 MHz VCLK4) 7h = Divide by 8 (1.56 MHz at 100 MHz VCLK4)
4-1	OSHTWTH	R/W	0h	One-Shot Pulse Width. 0h = 1 x VCLK4 / 8 wide (= 80 nS at 100 MHz VCLK4) 1h = 2 x VCLK4 / 8 wide (= 160 nS at 100 MHz VCLK4) 2h = 3 x VCLK4 / 8 wide (= 240 nS at 100 MHz VCLK4) 3h = 4 x VCLK4 / 8 wide (= 320 nS at 100 MHz VCLK4) 4h = 5 x VCLK4 / 8 wide (= 400 nS at 100 MHz VCLK4) 5h = 6 x VCLK4 / 8 wide (= 480 nS at 100 MHz VCLK4) 6h = 7 x VCLK4 / 8 wide (= 560 nS at 100 MHz VCLK4) 7h = 8 x VCLK4 / 8 wide (= 640 nS at 100 MHz VCLK4) 8h = 9 x VCLK4 / 8 wide (= 720 nS at 100 MHz VCLK4) 9h = 10 x VCLK4 / 8 wide (= 800 nS at 100 MHz VCLK4) Ah = 11 x VCLK4 / 8 wide (= 880 nS at 100 MHz VCLK4) Bh = 12 x VCLK4 / 8 wide (= 960 nS at 100 MHz VCLK4) Ch = 13 x VCLK4 / 8 wide (= 1040 nS at 100 MHz VCLK4) Dh = 14 x VCLK4 / 8 wide (= 1120 nS at 100 MHz VCLK4) Eh = 15 x VCLK4 / 8 wide (= 1200 nS at 100 MHz VCLK4) Fh = 16 x VCLK4 / 8 wide (= 1280 nS at 100 MHz VCLK4)
0	CHPEN	R/W	0h	PWM-chopping Enable. 0h = Disable (bypass) PWM chopping function. 1h = Enable chopping function.

19.4.29 DCTRIPSEL Register (Offset = 60h) [reset = 0h]

DCTRIPSEL is shown in Figure 19-91 and described in Table 19-52.

Return to Summary Table.

Figure 19-91. DCTRIPSEL Register

15	14	13	12	11	10	9	8
DCBLCOMPSEL				DCBHCOMPSEL			
R/W-0h				R/W-0h			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DCALCOMPSEL				DCAHCOMPSEL			
R/W-0h				R/W-0h			

Table 19-52. DCTRIPSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	DCBLCOMPSEL	R/W	0h	Digital Compare B Low Input Select. Defines the source for the DCBL input. The TZ signals, when used as trip signals, are treated as normal inputs and are defined as active high or active low. Values not shown are reserved. If a device does not have a particular comparitor, then that option is reserved. 0h = /TZ1 input 1h = /TZ2 input 2h = /TZ3 input
11-8	DCBHCOMPSEL	R/W	0h	Digital Compare B High Input Select. Defines the source for the DCBH input. The TZ signals, when used as trip signals, are treated as normal inputs and are defined as active high or active low. Values not shown are reserved. If a device does not have a particular comparitor, then that option is reserved. 0h = /TZ1 input 1h = /TZ2 input 2h = /TZ3 input
7-4	DCALCOMPSEL	R/W	0h	Digital Compare A Low Input Select. Defines the source for the DCAL input. The TZ signals, when used as trip signals, are treated as normal inputs and are defined as active high or active low. Values not shown are reserved. If a device does not have a particular comparitor, then that option is reserved. 0h = /TZ1 input 1h = /TZ2 input 2h = /TZ3 input
3-0	DCAHCOMPSEL	R/W	0h	Digital Compare A High Input Select. Defines the source for the DCAH input. The TZ signals, when used as trip signals, are treated as normal inputs and are defined as active high or active low. Values not shown are reserved. If a device does not have a particular comparitor, then that option is reserved. 0h = /TZ1 input 1h = /TZ2 input 2h = /TZ3 input

19.4.30 DCACTL Register (Offset = 62h) [reset = 0h]

DCACTL is shown in Figure 19-92 and described in Table 19-53.

Return to Summary Table.

Figure 19-92. DCACTL Register

15	14	13	12	11	10	9	8
RESERVED						EVT2FRC_SYNCSEL	EVT2SRCSEL
R-0h						R/W-0h	R/W-0h

7	6	5	4	3	2	1	0
RESERVED				EVT1SYNCE	EVT1SOCE	EVT1FRC_SYNCSEL	EVT1SRCSEL
R-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

Table 19-53. DCACTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9	EVT2FRC_SYNCSEL	R/W	0h	DCAEVT2 Force Synchronization Signal Select. 0h = Source Is Synchronous Signal. 1h = Source Is Asynchronous Signal.
8	EVT2SRCSEL	R/W	0h	DCAEVT2 Source Signal Select. 0h = Source Is DCAEVT2 Signal. 1h = Source Is DCEVTFILT Signal.
7-4	RESERVED	R	0h	Reserved
3	EVT1SYNCE	R/W	0h	DCAEVT1 SYNC Enable. 0h = SYNC generation is disabled. 1h = SYNC generation is enabled.
2	EVT1SOCE	R/W	0h	DCAEVT1 SOC Enable. 0h = SOC generation is disabled. 1h = SOC generation is enabled.
1	EVT1FRC_SYNCSEL	R/W	0h	DCAEVT1 Force Synchronization Signal Select. 0h = Source Is Synchronous Signal. 1h = Source Is Asynchronous Signal.
0	EVT1SRCSEL	R/W	0h	DCAEVT1 Source Signal Select. 0h = Source Is DCAEVT1 Signal. 1h = Source Is DCEVTFILT Signal.

19.4.31 DCBCTL Register (Offset = 64h) [reset = 0h]

DCBCTL is shown in Figure 19-93 and described in Table 19-54.

Return to Summary Table.

Figure 19-93. DCBCTL Register

15	14	13	12	11	10	9	8
RESERVED						EVT2FRC_SYNCSEL	EVT2SRCSEL
R-0h						R/W-0h	R/W-0h

7	6	5	4	3	2	1	0
RESERVED				EVT1SYNCE	EVT1SOCE	EVT1FRC_SYNCSEL	EVT1SRCSEL
R-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

Table 19-54. DCBCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-10	RESERVED	R	0h	Reserved
9	EVT2FRC_SYNCSEL	R/W	0h	DCBEVT2 Force Synchronization Signal Select. 0h = Source Is Synchronous Signal. 1h = Source Is Asynchronous Signal.
8	EVT2SRCSEL	R/W	0h	DCBEVT2 Source Signal Select. 0h = Source Is DCBEVT2 Signal. 1h = Source Is DCEVTFILT Signal.
7-4	RESERVED	R	0h	Reserved
3	EVT1SYNCE	R/W	0h	DCBEVT1 SYNC Enable. 0h = SYNC generation is disabled. 1h = SYNC generation is enabled.
2	EVT1SOCE	R/W	0h	DCBEVT1 SOC Enable. 0h = SOC generation is disabled. 1h = SOC generation is enabled.
1	EVT1FRC_SYNCSEL	R/W	0h	DCBEVT1 Force Synchronization Signal Select. 0h = Source Is Synchronous Signal. 1h = Source Is Asynchronous Signal.
0	EVT1SRCSEL	R/W	0h	DCBEVT1 Source Signal Select. 0h = Source Is DCBEVT1 Signal. 1h = Source Is DCEVTFILT Signal.

19.4.32 DCFCTL Register (Offset = 66h) [reset = 0h]

DCFCTL is shown in Figure 19-94 and described in Table 19-55.

Return to Summary Table.

Figure 19-94. DCFCTL Register

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="3">RESERVED</td><td colspan="5">RESERVED</td></tr>
<tr><td colspan="3">R-0h</td><td colspan="5">R-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>RESERVED</td><td>RESERVED</td><td colspan="2">PULSESEL</td><td>BLANKINV</td><td>BLANKE</td><td colspan="2">SRCSEL</td></tr>
<tr><td>R-0h</td><td>R-0h</td><td colspan="2">R/W-0h</td><td>R/W-0h</td><td>R/W-0h</td><td colspan="2">R/W-0h</td></tr>
</table>

Table 19-55. DCFCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-13	RESERVED	R	0h	Reserved
12-8	RESERVED	R	0h	Reserved for TI Test
7	RESERVED	R	0h	Reserved
6	RESERVED	R	0h	Reserved for TI Test
5-4	PULSESEL	R/W	0h	Pulse Select For Blanking and Capture Alignment. 0h = Time-base counter equal to period (TBCTR = TBPRD) 1h = Time-base counter equal to zero (TBCTR = 0x0000) 2h = Reserved 3h = Reserved
3	BLANKINV	R/W	0h	Blanking Window Inversion. 0h = Blanking window is not inverted. 1h = Blanking window is inverted.
2	BLANKE	R/W	0h	Blanking Window Enable. 0h = Blanking window is disabled. 1h = Blanking window is enabled.
1-0	SRCSEL	R/W	0h	Filter Block Signal Source Select. 0h = Source Is DCAEVT1 Signal. 1h = Source Is DCAEVT2 Signal. 2h = Source Is DCBEVT1 Signal. 3h = Source Is DCBEVT2 Signal.

19.4.33 DCCAPCTL Register (Offset = 68h) [reset = 0h]

DCCAPCTL is shown in Figure 19-95 and described in Table 19-56.

Return to Summary Table.

Figure 19-95. DCCAPCTL Register

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="6">RESERVED</td><td>SHDWMODE</td><td>CAPE</td></tr>
<tr><td colspan="6">R-0h</td><td>R/W-0h</td><td>R/W-0h</td></tr>
</table>

Table 19-56. DCCAPCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-2	RESERVED	R	0h	Reserved
1	SHDWMODE	R/W	0h	TBCTR Counter Capture Shadow Select Mode. 0h = Enable shadow mode. The DCCAP active register is copied to shadow register on a TBCTR = TBPRD or TBCTR = zero event as defined by the DCFCTL[PULSESEL] bit. CPU reads of the DCCAP register return the shadow register contents. 1h = Active Mode. In this mode the shadow register is disabled. CPU reads from the DCCAP register always returns the active register contents.
0	CAPE	R/W	0h	TBCTR Counter Capture Enable. 0h = Disable the time-base counter capture. 1h = Enable the time-base counter capture.

19.4.34 DCFOFFSET Register (Offset = 6Ah) [reset = 0h]

DCFOFFSET is shown in Figure 19-96 and described in Table 19-57.

Return to Summary Table.

Figure 19-96. DCFOFFSET Register

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">OFFSET</td></tr>
<tr><td colspan="16">R-0h</td></tr>
</table>

Table 19-57. DCFOFFSET Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	OFFSET	R	0h	Blanking Window Offset. Valid values: 0-FFFFh These 16-bits specify the number of TBCLK cycles from the blanking window reference to the point when the blanking window is applied. The blanking window reference is either period or zero as defined by the DCFCTL[PULSESEL] bit. This offset register is shadowed and the active register is loaded at the reference point defined by DCFCTL[PULSESEL]. The offset counter is also initialized and begins to count down when the active register is loaded. When the counter expires, the blanking window is applied. If the blanking window is currently active, then the blanking window counter is restarted.

19.4.35 DCFOFFSETCNT Register (Offset = 6Ch) [reset = 0h]

DCFOFFSETCNT is shown in Figure 19-97 and described in Table 19-58.

Return to Summary Table.

Figure 19-97. DCFOFFSETCNT Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSETCNT															
R-0h															

Table 19-58. DCFOFFSETCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	OFFSETCNT	R	0h	Blanking Offset Counter. Valid values: 0-FFFFh These 16-bits are read-only and indicate the current value of the offset counter. The counter counts down to zero and then stops until it is re-loaded on the next period or zero event as defined by the DCFCTL[PULSESEL] bit. The offset counter is not affected by the FREE and SOFT emulation bits. That is, it always continues to count down if the device is halted by an emulation stop.

19.4.36 DCFWINDOW Register (Offset = 6Eh) [reset = 0h]

DCFWINDOW is shown in Figure 19-98 and described in Table 19-59.

Return to Summary Table.

Figure 19-98. DCFWINDOW Register

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">WINDOW</td></tr>
<tr><td colspan="8">R/W-0h</td></tr>
</table>

Table 19-59. DCFWINDOW Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7-0	WINDOW	R/W	0h	Blanking Window Width. Valid values: 0-FFh These 16-bits specify the width of the blanking window in TBCLK cycles. The blanking window begins when the offset counter expires. When this occurs, the window counter is loaded and begins to count down. If the blanking window is currently active and the offset counter expires, the blanking window counter is restarted. The blanking window can cross a PWM period boundary. 0h = No blanking window is generated.

19.4.37 DCFWINDOWCNT Register (Offset = 70h) [reset = 0h]

DCFWINDOWCNT is shown in Figure 19-99 and described in Table 19-60.

Return to Summary Table.

Figure 19-99. DCFWINDOWCNT Register

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">WINDOWCNT</td></tr>
<tr><td colspan="8">R-0h</td></tr>
</table>

Table 19-60. DCFWINDOWCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Any writes to these bit(s) must always have a value of 0.
7-0	WINDOWCNT	R	0h	Blanking Window Counter. Valid value: 0-FFh These 8 bits are read only and indicate the current value of the window counter. The counter counts down to zero and then stops until it is re-loaded when the offset counter reaches zero again.

19.4.38 DCCAP Register (Offset = 72h) [reset = 0h]

DCCAP is shown in Figure 19-100 and described in Table 19-61.

Return to Summary Table.

Figure 19-100. DCCAP Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCCAP															
R-0h															

Table 19-61. DCCAP Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	DCCAP	R	0h	Digital Compare Time-Base Counter Capture. Valid values: 0-FFFFh To enable time-base counter capture, set the DCCAPCLT[CAPE] bit to 1. If enabled, reflects the value of the time-base counter (TBCTR) on the low-to-high edge transition of a filtered (DCEVTFLT) event. Further capture events are ignored until the next period or zero as selected by the DCFCTL[PULSESEL] bit. Shadowing of DCCAP is enabled and disabled by the DCCAPCTL[SHDWMODE] bit. By default this register is shadowed. If DCCAPCTL[SHDWMODE] = 0, then the shadow is enabled. In this mode, the active register is copied to the shadow register on the TBCTR = TBPRD or TBCTR = zero as defined by the DCFCTL[PULSESEL] bit. CPU reads of this register return the shadow register value. If DCCAPCTL[SHDWMODE] = 1, then the shadow register is disabled. In this mode, CPU reads return the active register value. The active and shadow registers share the same memory map address.

Enhanced Capture (eCAP) Module

The enhanced Capture (eCAP) module is essential in systems where accurate timing of external events is important. This microcontroller implements 6 instances of the eCAP module.

Topic **Page**

20.1 Introduction ... 852
20.2 Description ... 852
20.3 Capture and APWM Operating Mode ... 853
20.4 Capture Mode Description ... 854
20.5 ECAP Registers ... 860
20.6 Application of the ECAP Module ... 875
20.7 Application of the APWM Mode ... 884

20.1 Introduction

Uses for eCAP include:

- Speed measurements of rotating machinery (for example, toothed sprockets sensed via Hall sensors)
- Elapsed time measurements between position sensor pulses
- Period and duty cycle measurements of pulse train signals
- Decoding current or voltage amplitude derived from duty cycle encoded current/voltage sensors

The eCAP module described in this guide includes the following features:

- 4-event time-stamp registers (each 32 bits)
- Edge polarity selection for up to four sequenced time-stamp capture events
- Interrupt on either of the four events
- Single shot capture of up to four event time-stamps
- Continuous mode capture of time-stamps in a four-deep circular buffer
- Absolute time-stamp capture
- Difference (Delta) mode time-stamp capture
- All above resources dedicated to a single input pin
- When not used in capture mode, the ECAP module can be configured as a single channel PWM output

20.2 Description

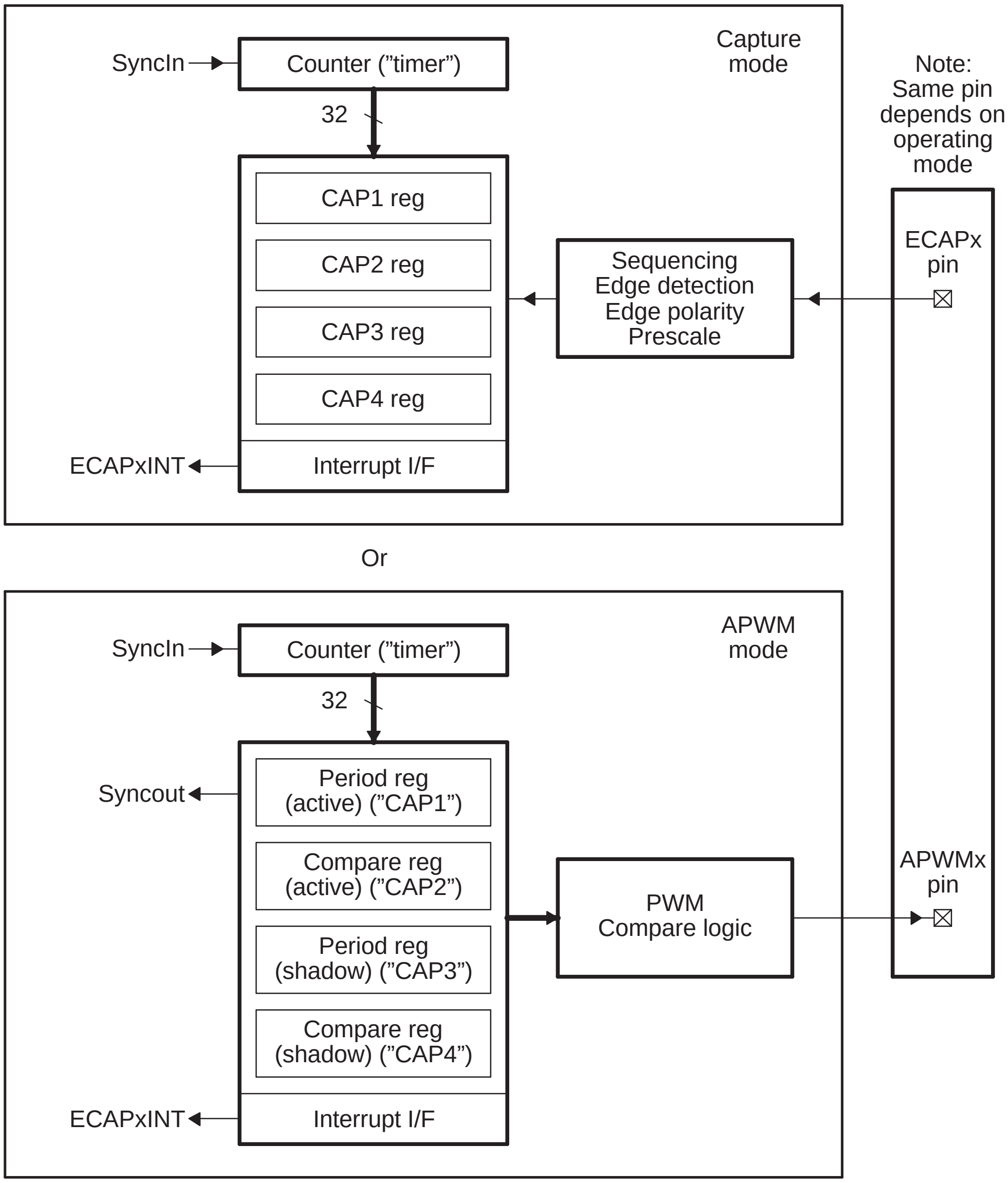
In the context of this chapter, one eCAP channel has the following independent key resources:

- 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 to 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

20.3 Capture and APWM Operating Mode

You can use the eCAP module resources to implement a single-channel PWM generator (with 32 bit capabilities) when it is not being used for input captures. The counter operates in count-up mode, providing a time-base for asymmetrical pulse width modulation (PWM) waveforms. The CAP1 and CAP2 registers become the active period and compare registers, respectively, while CAP3 and CAP4 registers become the period and capture shadow registers, respectively. Figure 20-1 is a high-level view of both the capture and auxiliary pulse-width modulator (APWM) modes of operation.

Figure 20-1. Capture and APWM Modes of Operation



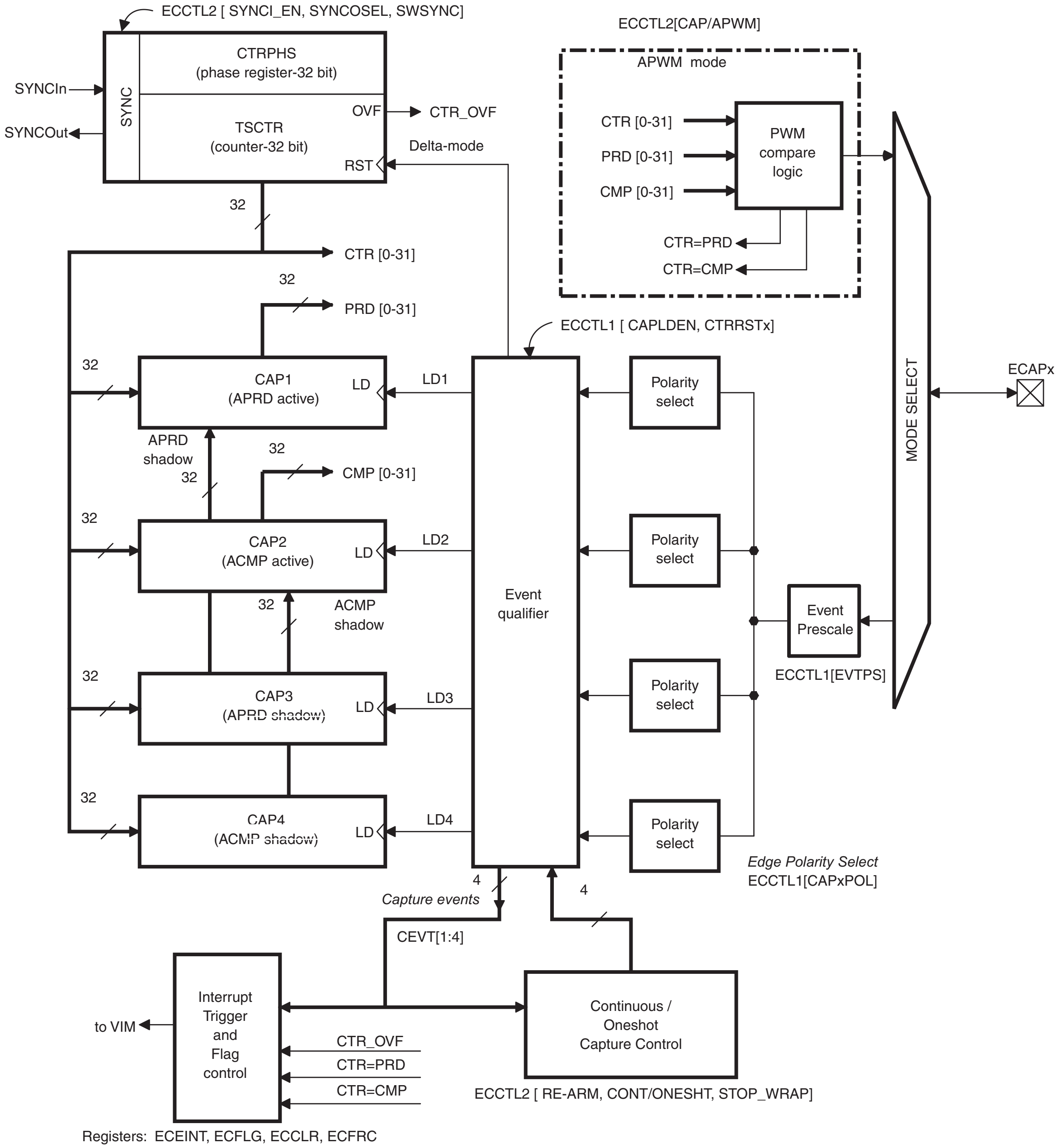
A A single pin is shared between CAP and APWM functions. In capture mode, it is an input; in APWM mode, it is an output.

B In APWM mode, writing any value to CAP1/CAP2 active registers also writes the same value to the corresponding shadow registers CAP3/CAP4. This emulates immediate mode. Writing to the shadow registers CAP3/CAP4 invokes the shadow mode.

20.4 Capture Mode Description

Figure 20-2 shows the various components that implement the capture function.

Figure 20-2. Capture Function Diagram

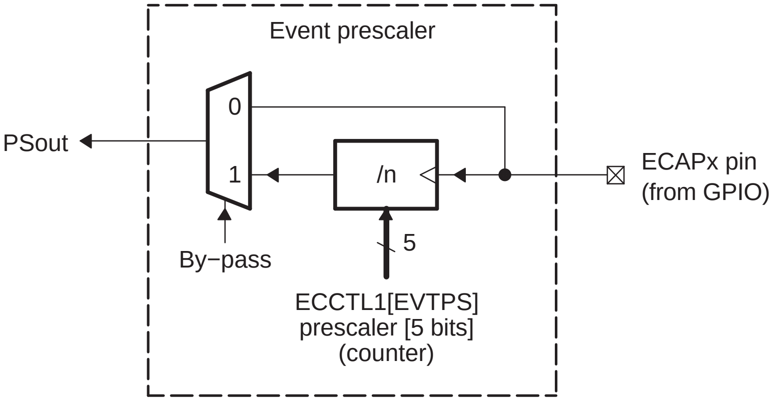


20.4.1 Event Prescaler

- An input capture signal (pulse train) can be prescaled by N = 2-62 (in multiples of 2) or can bypass the prescaler.

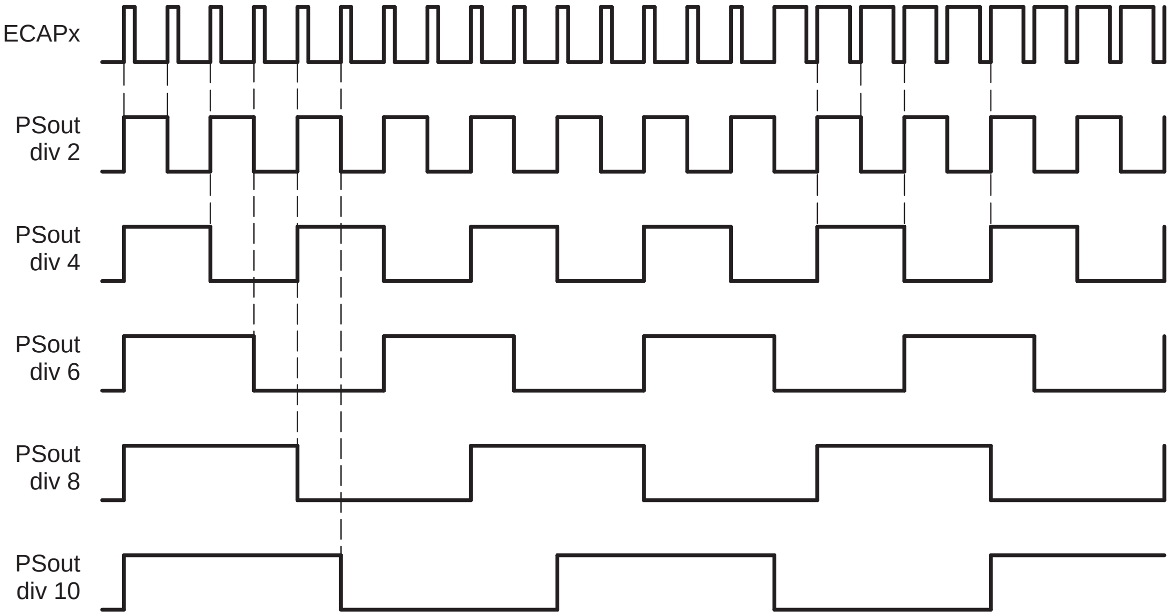
  This is useful when very high frequency signals are used as inputs. Figure 20-3 shows a functional diagram and Figure 20-4 shows the operation of the prescale function.

Figure 20-3. Event Prescale Control



A When a prescale value of 1 is chosen (that is, ECCTL1[13:9] = 0,0,0,0,0) the input capture signal by-passes the prescale logic completely.

Figure 20-4. Prescale Function Waveforms



20.4.2 Edge Polarity Select and Qualifier

- Four independent edge polarity (rising edge/falling edge) selection MUXes are used, one for each capture event.
- Each edge (up to 4) is event qualified by the Modulo4 sequencer.
- The edge event is gated to its respective CAPx register by the Mod4 counter. The CAPx register is loaded on the falling edge.

20.4.3 Continuous/One-Shot Control

- The Mod4 (2 bit) counter is incremented via edge qualified events (CEVT1-CEVT4).
- The Mod4 counter continues counting (0->1->2->3->0) and wraps around unless stopped.
- A 2-bit stop register is used to compare the Mod4 counter output, and when equal stops the Mod4 counter and inhibits further loads of the CAP1-CAP4 registers. This occurs during one-shot operation.

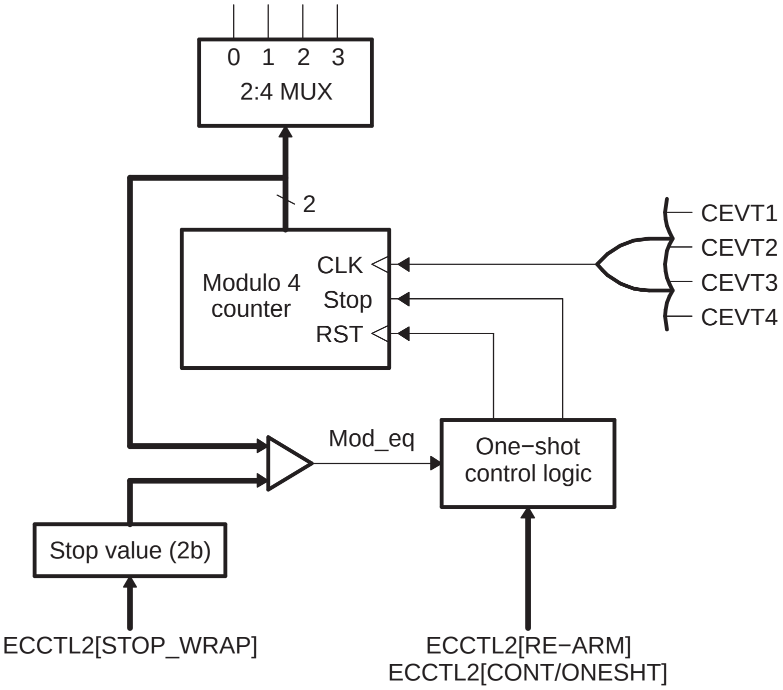
The continuous/one-shot block controls the start/stop and reset (zero) functions of the Mod4 counter via a mono-shot type of action that can be triggered by the stop-value comparator and re-armed via software control.

Once armed, the eCAP module waits for 1-4 (defined by stop-value) capture events before freezing both the Mod4 counter and contents of CAP1-4 registers (that is, 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 20-5. Details of the Continuous/One-shot Block



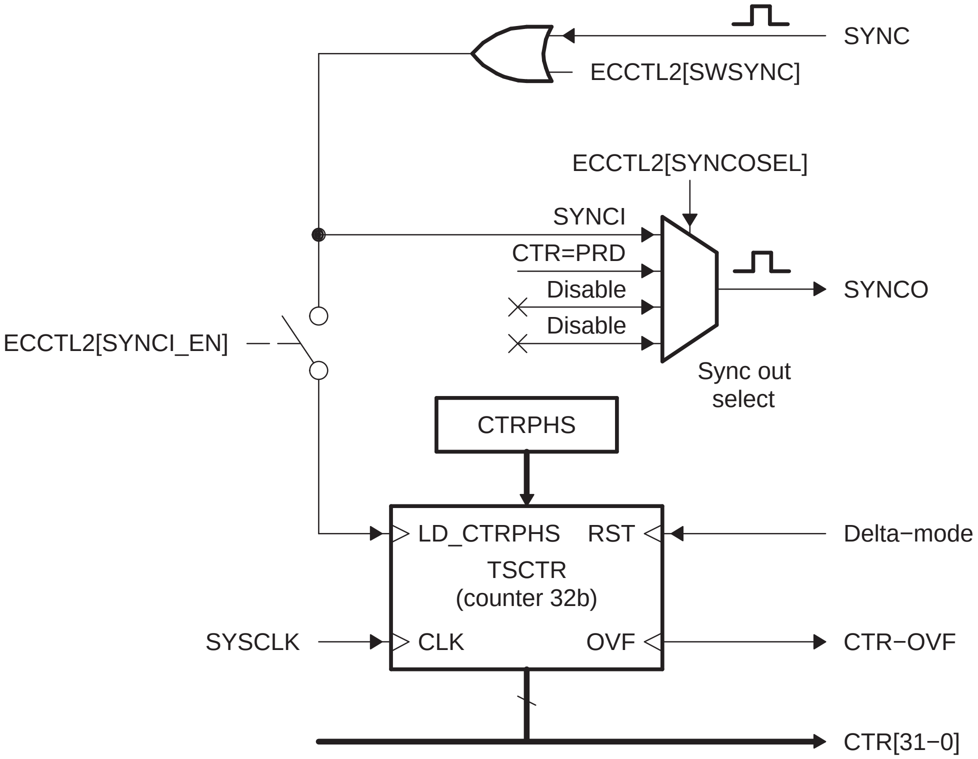
20.4.4 32-Bit Counter and Phase Control

This counter provides the time-base for event captures, and is clocked via the system clock.

A phase register is provided to achieve synchronization with other counters, via a hardware and software forced sync. This is useful in APWM mode when a phase offset between modules is needed.

On any of the four event loads, an option to reset the 32-bit counter is given. This is useful for time difference capture. The 32-bit counter value is captured first, then it is reset to 0 by any of the LD1-LD4 signals.

Figure 20-6. Details of the Counter and Synchronization Block



20.4.5 CAP1-CAP4 Registers

These 32-bit registers are fed by the 32-bit counter timer bus, CTR[0-31] and are loaded (that is, 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, that is, 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.

20.4.6 Interrupt Control

An interrupt can be generated on capture events (CEVT1-CEVT4, CTROVF) or APWM events (CTR = PRD, CTR = CMP).

A counter overflow event (FFFFFFFF->00000000) is also provided as an interrupt source (CTROVF).

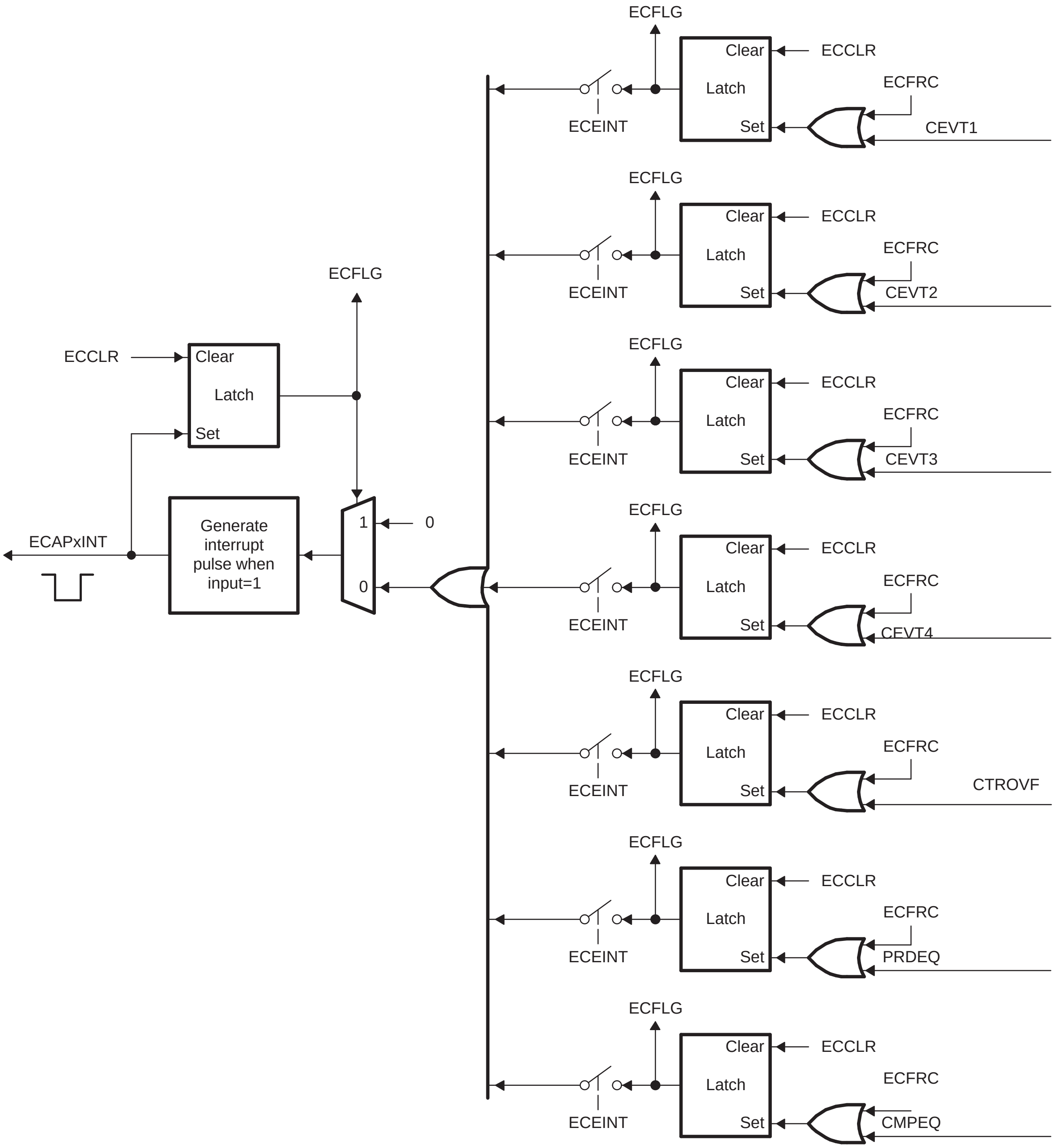
The capture events are edge and sequencer qualified (that is, ordered in time) by the polarity select and Mod4 gating, respectively.

One of these events can be selected as the interrupt source (from the eCAPx module) going to the PIE.

Seven interrupt events (CEVT1, CEVT2, CEVT3, CEVT4, CNTOVF, CTR=PRD, CTR=CMP) can be generated. The interrupt enable register (ECEINT) is used to enable/disable individual interrupt event sources. The interrupt flag register (ECFLG) indicates if any interrupt event has been latched and contains the global interrupt flag bit (INT). An interrupt pulse is generated to the PIE only if any of the interrupt events are enabled, the flag bit is 1, and the INT flag bit is 0. The interrupt service routine must clear the global interrupt flag bit and the serviced event via the interrupt clear register (ECCLR) before any other interrupt pulses are generated. You can force an interrupt event via the interrupt force register (ECFRC). This is useful for test purposes.

Note: The CEVT1, CEVT2, CEVT3, CEVT4 flags are only active in capture mode (ECCTL2[CAP_APWM == 0]). The CTR_PRD, CTR_CMP flags are only valid in APWM mode (ECCTL2[CAP_APWM == 1]). CNTOVF flag is valid in both modes.

Figure 20-7. Interrupts in eCAP Module



20.4.7 Shadow Load and Lockout Control

In capture mode, this logic inhibits (locks out) any shadow loading of CAP1 or CAP2 from APRD and ACMP registers, respectively.

In APWM mode, shadow loading is active and two choices are permitted:

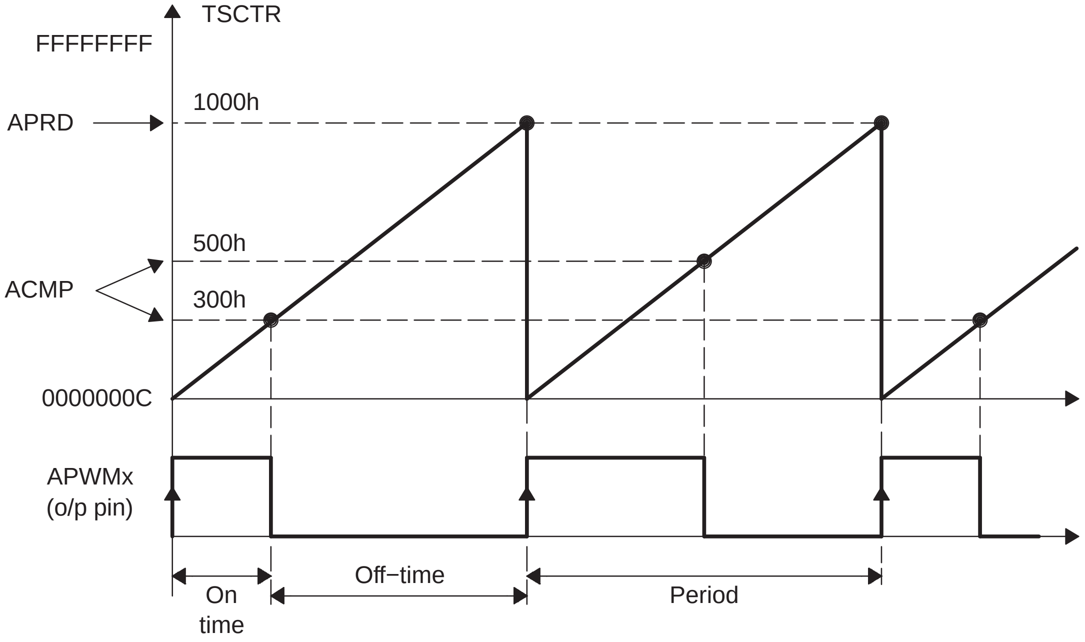
- Immediate - APRD or ACMP are transferred to CAP1 or CAP2 immediately upon writing a new value.
- On period equal, that is, CTR[31:0] = PRD[31:0]

20.4.8 APWM Mode Operation

Main operating highlights of the APWM section:

- The time-stamp counter bus is made available for comparison via 2 digital (32-bit) comparators.
- When CAP1/2 registers are not used in capture mode, their contents can be used as Period and Compare values in APWM mode.
- Double buffering is achieved via shadow registers APRD and ACMP (CAP3/4). The shadow register contents are transferred over to CAP1/2 registers either immediately upon a write, or on a CTR = PRD trigger.
- In APWM mode, writing to CAP1/CAP2 active registers will also write the same value to the corresponding shadow registers CAP3/CAP4. This emulates immediate mode. Writing to the shadow registers CAP3/CAP4 will invoke the shadow mode.
- During initialization, you must write to the active registers for both period and compare. This automatically copies the initial values into the shadow values. For subsequent compare updates, that is, during run-time, you only need to use the shadow registers.

Figure 20-8. PWM Waveform Details Of APWM Mode Operation



The behavior of APWM active high mode (APWMPOL == 0) is as follows:

```
CMP = 0x00000000, output low for duration of period (0% duty)
CMP = 0x00000001, output high 1 cycle
CMP = 0x00000002, output high 2 cycles
CMP = PERIOD, output high except for 1 cycle (<100% duty)
CMP = PERIOD+1, output high for complete period (100% duty)
CMP > PERIOD+1, output high for complete period
```

The behavior of APWM active low mode (APWMPOL == 1) is as follows:

```
CMP = 0x00000000, output high for duration of period (0% duty)
CMP = 0x00000001, output low 1 cycle
CMP = 0x00000002, output low 2 cycles
CMP = PERIOD, output low except for 1 cycle (<100% duty)
CMP = PERIOD+1, output low for complete period (100% duty)
CMP > PERIOD+1, output low for complete period
```

20.5 ECAP Registers

Table 20-1 lists the memory-mapped registers for the ECAP. All register offset addresses not listed in Table 20-1 should be considered as reserved locations and the register contents should not be modified.

The base address for the control registers is FCF7 9300h for eCAP1, FCF7 9400h for eCAP2, FCF7 9500h for eCAP3, FCF7 9600h for eCAP4, FCF7 9700h for eCAP5, and FCF7 9800h for eCAP6.

Table 20-1. ECAP Registers

Offset	Acronym	Register Name	Section
0h	TSCTR	Time-Stamp Counter	Section 20.5.1
4h	CTRPHS	Counter Phase Offset Value Register	Section 20.5.2
8h	CAP1	Capture 1 Register	Section 20.5.3
Ch	CAP2	Capture 2 Register	Section 20.5.4
10h	CAP3	Capture 3 Register	Section 20.5.5
14h	CAP4	Capture 4 Register	Section 20.5.6
28h	ECCTL1	Capture Control Register 1	Section 20.5.7
2Ah	ECCTL2	Capture Control Register 2	Section 20.5.8
2Ch	ECEINT	Capture Interrupt Enable Register	Section 20.5.9
2Eh	ECFLG	Capture Interrupt Flag Register	Section 20.5.10
30h	ECCLR	Capture Interrupt Clear Register	Section 20.5.11
32h	ECFRC	Capture Interrupt Force Register	Section 20.5.12

Complex bit access types are encoded to fit into small table cells. Table 20-2 shows the codes that are used for access types in this section.

Table 20-2. ECAP Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

20.5.1 TSCTR Register (Offset = 0h) [reset = 0h]

TSCTR is shown in Figure 20-9 and described in Table 20-3.

Return to Summary Table.

Figure 20-9. TSCTR Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="32">TSCTR</td></tr>
<tr><td colspan="32">R/W-0h</td></tr>
</table>

Table 20-3. TSCTR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TSCTR	R/W	0h	Active 32-bit counter register that is used as the capture time-base.

20.5.2 CTRPHS Register (Offset = 4h) [reset = 0h]

CTRPHS is shown in Figure 20-10 and described in Table 20-4.

Return to Summary Table.

Figure 20-10. CTRPHS Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="32">CTRPHS</td></tr>
<tr><td colspan="32">R/W-0h</td></tr>
</table>

Table 20-4. CTRPHS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CTRPHS	R/W	0h	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 software force via a control bit. Used to achieve phase control synchronization with respect to other eCAP and ePWM time-bases.

20.5.3 CAP1 Register (Offset = 8h) [reset = 0h]

CAP1 is shown in Figure 20-11 and described in Table 20-5.

Return to Summary Table.

In APWM mode, writing to CAP1/CAP2 active registers also writes the same value to the corresponding shadow registers CAP3/CAP4. This emulates immediate mode. Writing to the shadow registers CAP3/CAP4 invokes the shadow mode.

Figure 20-11. CAP1 Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="32">CAP1</td></tr>
<tr><td colspan="32">R/W-0h</td></tr>
</table>

Table 20-5. CAP1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CAP1	R/W	0h	This register can be loaded (written) by: Time-Stamp (that is, counter value) during a capture event. Software - may be useful for test purposes. APRD shadow register (that is, CAP4) when used in APWM mode.

20.5.4 CAP2 Register (Offset = Ch) [reset = 0h]

CAP2 is shown in Figure 20-12 and described in Table 20-6.

Return to Summary Table.

In APWM mode, writing to CAP1/CAP2 active registers also writes the same value to the corresponding shadow registers CAP3/CAP4. This emulates immediate mode. Writing to the shadow registers CAP3/CAP4 invokes the shadow mode.

Figure 20-12. CAP2 Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="32">CAP2</td></tr>
<tr><td colspan="32">R/W-0h</td></tr>
</table>

Table 20-6. CAP2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CAP2	R/W	0h	This register can be loaded (written) by: Time-Stamp (that is, counter value) during a capture event. Software - may be useful for test purposes. APRD shadow register (that is, CAP4) when used in APWM mode.

20.5.5 CAP3 Register (Offset = 10h) [reset = 0h]

CAP3 is shown in Figure 20-13 and described in Table 20-7.

Return to Summary Table.

Figure 20-13. CAP3 Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="32">CAP3</td></tr>
<tr><td colspan="32">R/W-0h</td></tr>
</table>

Table 20-7. CAP3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CAP3	R/W	0h	In CMP mode, this is a time-stamp capture register. In APWM mode, this is the period shadow (APRD) register. Update the PWM period value through this register. In this mode, CAP3 (APRD) shadows CAP1.

20.5.6 CAP4 Register (Offset = 14h) [reset = 0h]

CAP4 is shown in Figure 20-14 and described in Table 20-8.

Return to Summary Table.

Figure 20-14. CAP4 Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="32">CAP4</td></tr>
<tr><td colspan="32">R/W-0h</td></tr>
</table>

Table 20-8. CAP4 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CAP4	R/W	0h	In CMP mode, this is a time-stamp capture register. In APWM mode, this is the compare shadow (ACMP) register. Update the PWM compare value via this register. In this mode, CAP4 (ACMP) shadows CAP2.

20.5.7 ECCTL1 Register (Offset = 28h) [reset = 0h]

ECCTL1 is shown in Figure 20-15 and described in Table 20-9.

Return to Summary Table.

Figure 20-15. ECCTL1 Register

15	14	13	12	11	10	9	8
FREE_SOFT		PRESCALE					CAPLDEN
R/W-0h		R/W-0h					R/W-0h
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CTRRST4	CAP4POL	CTRRST3	CAP3POL	CTRRST2	CAP2POL	CTRRST1	CAP1POL
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

Table 20-9. ECCTL1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-14	FREE_SOFT	R/W	0h	Emulation Control. 0h = TSCTR counter stops immediately on emulation suspend. 1h = TSCTR counter runs until value equals 0. 2h = TSCTR counter is unaffected by emulation suspend (Run Free). 3h = TSCTR counter is unaffected by emulation suspend (Run Free).
13-9	PRESCALE	R/W	0h	Event Filter prescale select. Valid values: 0 to 1Fh. 0h = Divide by 1 (that is, no prescale, by-pass the prescaler) 1h = 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	R/W	0h	Enable Loading of CAP1-4 registers on a capture event. 0h = Disable CAP1-4 register loads at capture event time. 1h = Enable CAP1-4 register loads at capture event time.
7	CTRRST4	R/W	0h	Counter Reset on Capture Event 4. 0h = Do not reset counter on Capture Event 4 (absolute time stamp operation). 1h = Reset counter after Capture Event 4 time-stamp has been captured (used in difference mode operation).
6	CAP4POL	R/W	0h	Capture Event 4 Polarity select. 0h = Capture Event 4 triggered on a rising edge (RE). 1h = Capture Event 4 triggered on a falling edge (FE).
5	CTRRST3	R/W	0h	Counter Reset on Capture Event 3. 0h = Do not reset counter on Capture Event 3 (absolute time stamp). 1h = Reset counter after Event 3 time-stamp has been captured (used in difference mode operation).
4	CAP3POL	R/W	0h	Capture Event 3 Polarity select. 0h = Capture Event 3 triggered on a rising edge (RE). 1h = Capture Event 3 triggered on a falling edge (FE).
3	CTRRST2	R/W	0h	Counter Reset on Capture Event 2. 0h = Do not reset counter on Capture Event 2 (absolute time stamp). 1h = Reset counter after Event 2 time-stamp has been captured (used in difference mode operation).

Table 20-9. ECCTL1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	CAP2POL	R/W	0h	Capture Event 2 Polarity select. 0h = Capture Event 2 triggered on a rising edge (RE). 1h = Capture Event 2 triggered on a falling edge (FE).
1	CTRRST1	R/W	0h	Counter Reset on Capture Event 1. 0h = Do not reset counter on Capture Event 1 (absolute time stamp). 1h = Reset counter after Event 1 time-stamp has been captured (used in difference mode operation).
0	CAP1POL	R/W	0h	Capture Event 1 Polarity select. 0h = Capture Event 1 triggered on a rising edge (RE). 1h = Capture Event 1 triggered on a falling edge (FE).

20.5.8 ECCTL2 Register (Offset = 2Ah) [reset = 2h]

ECCTL2 is shown in Figure 20-16 and described in Table 20-10.

Return to Summary Table.

Figure 20-16. ECCTL2 Register

15	14	13	12	11	10	9	8
RESERVED					APWMPOL	CAP_APWM	SWSYNC
R-0h					R/W-0h	R/W-0h	R/W-0h

7	6	5	4	3	2	1	0
SYNCO_SEL		SYNCI_EN	TSCTRSTOP	REARM	STOP_WRAP		CONT_ONESHT
R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-1h		R/W-0h

Table 20-10. ECCTL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-11	RESERVED	R	0h	Reserved
10	APWMPOL	R/W	0h	APWM output polarity select. This is applicable only in APWM operating mode. 0h = Output is active high (that is, Compare value defines high time). 1h = Output is active low (that is, Compare value defines low time).
9	CAP_APWM	R/W	0h	CAP/APWM operating mode select. 0h = 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 CAP2 registers, permits user to enable CAP1 to CAP4 register load, CAPx and APWMx pin operates as a capture input. 1h = 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 CAP2 registers, disables loading of time-stamps into CAP1 to CAP4 registers, CAPx and APWMx pin operates as a APWM output.
8	SWSYNC	R/W	0h	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. Note: Selection CTR = PRD is meaningful only in APWM mode, however, you can choose it in CAP mode if you find doing so is useful. 0h = Writing a 0 has no effect. Reading always returns 0. 1h = Writing a 1 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 0.
7-6	SYNCO_SEL	R/W	0h	Sync-Out Select. 0h = 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	R/W	0h	Counter (TSCTR) Sync-In select mode. 0h = Disable sync-in option. 1h = Enable counter (TSCTR) to be loaded from CTRPHS register upon either a SYNCI signal or a software force event.
4	TSCTRSTOP	R/W	0h	Time Stamp (TSCTR) Counter Stop (freeze) Control. 0h = TSCTR is stopped. 1h = TSCTR is free-running.

Table 20-10. ECCTL2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	REARM	R/W	0h	One-Shot Re-Arming Control, that is, wait for stop trigger. Note: The re-arm function is valid in one shot or continuous mode. 0h = Writing a 0 has no effect. Reading always returns 0. 1h = Arms the one-shot sequence as follows: resets the Mod4 counter to 0, unfreezes the Mod4 counter, enables capture register loads.
2-1	STOP_WRAP	R/W	1h	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. Notes: STOP_WRAP is compared to Mod4 counter and, when equal, two actions occur: 1) Mod4 counter is stopped (frozen). 2) Capture register loads are inhibited. In one-shot mode, further interrupt events are blocked until re-armed. 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. 0h = 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.
0	CONT_ONESHT	R/W	0h	Continuous or one-shot mode control. This is applicable only in capture mode. 0h = Operate in continuous mode. 1h = Operate in one-shot mode.

20.5.9 ECEINT Register (Offset = 2Ch) [reset = 0h]

ECEINT is shown in Figure 20-17 and described in Table 20-11.

Return to Summary Table.

The interrupt enable bits (CEVT1, ...) block any of the selected events from generating an interrupt. Events are still latched into the flag bit (ECFLG register) and can be forced/cleared via the ECFRC/ECCLR registers.

The proper procedure for configuring peripheral modes and interrupts is as follows:

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.

Figure 20-17. ECEINT Register

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>CTR_CMP</td><td>CTR_PRD</td><td>CTROVF</td><td>CEVT4</td><td>CEVT3</td><td>CEVT2</td><td>CEVT1</td><td>RESERVED</td></tr>
<tr><td>R/W-0h</td><td>R/W-0h</td><td>R/W-0h</td><td>R/W-0h</td><td>R/W-0h</td><td>R/W-0h</td><td>R/W-0h</td><td>R-0h</td></tr>
</table>

Table 20-11. ECEINT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	CTR_CMP	R/W	0h	Counter Equal Compare Interrupt Enable. 0h = Disable Compare Equal as an Interrupt source. 1h = Enable Compare Equal as an Interrupt source.
6	CTR_PRD	R/W	0h	Counter Equal Period Interrupt Enable. 0h = Disable Period Equal as an Interrupt source. 1h = Enable Period Equal as an Interrupt source.
5	CTROVF	R/W	0h	Counter Overflow Interrupt Enable. 0h = Disabled Counter Overflow as an Interrupt source. 1h = Enable Counter Overflow as an Interrupt source.
4	CEVT4	R/W	0h	Capture Event 4 Interrupt Enable. 0h = Disable Capture Event 4 as an Interrupt source. 1h = Enable Capture Event 4 as an Interrupt source.
3	CEVT3	R/W	0h	Capture Event 3 Interrupt Enable. 0h = Disable Capture Event 3 as an Interrupt source. 1h = Enable Capture Event 3 as an Interrupt source.
2	CEVT2	R/W	0h	Capture Event 2 Interrupt Enable. 0h = Disable Capture Event 2 as an Interrupt source. 1h = Enable Capture Event 2 as an Interrupt source.
1	CEVT1	R/W	0h	Capture Event 1 Interrupt Enable. 0h = Disable Capture Event 1 as an Interrupt source. 1h = Enable Capture Event 1 as an Interrupt source.
0	RESERVED	R	0h	Reserved

20.5.10 ECFLG Register (Offset = 2Eh) [reset = 0h]

ECFLG is shown in Figure 20-18 and described in Table 20-12.

Return to Summary Table.

Figure 20-18. ECFLG Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CTR_CMP	CTR_PRD	CTROVF	CEVT4	CEVT3	CEVT2	CEVT1	INT
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

Table 20-12. ECFLG Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Reserved
7	CTR_CMP	R	0h	Compare Equal Compare Status Flag. This flag is active only in APWM mode. 0h = No event occurred. 1h = The counter (TSCTR) reached the compare register value (ACMP).
6	CTR_PRD	R	0h	Counter Equal Period Status Flag. This flag is only active in APWM mode. 0h = No event occurred. 1h = The counter (TSCTR) reached the period register value (APRD) and was reset.
5	CTROVF	R	0h	Counter Overflow Status Flag. This flag is active in CAP and APWM mode. 0h = No event occurred. 1h = The counter (TSCTR) has made the transition from FFFFFFFF to 00000000.
4	CEVT4	R	0h	Capture Event 4 Status Flag This flag is only active in CAP mode. 0h = No event occurred. 1h = The fourth event occurred at ECAPx pin.
3	CEVT3	R	0h	Capture Event 3 Status Flag. This flag is active only in CAP mode. 0h = No event occurred. 1h = The third event occurred at ECAPx pin.
2	CEVT2	R	0h	Capture Event 2 Status Flag. This flag is only active in CAP mode. 0h = No event occurred. 1h = The second event occurred at ECAPx pin.
1	CEVT1	R	0h	Capture Event 1 Status Flag. This flag is only active in CAP mode. 0h = No event occurred. 1h = The first event occurred at ECAPx pin.
0	INT	R	0h	Global Interrupt Status Flag. 0h = No interrupt is generated. 1h = An interrupt is generated.

20.5.11 ECCLR Register (Offset = 30h) [reset = 0h]

ECCLR is shown in Figure 20-19 and described in Table 20-13.

Return to Summary Table.

Figure 20-19. ECCLR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
CTR_CMP	CTR_PRD	CTROVF	CEVT4	CEVT3	CEVT2	CEVT1	INT
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

Table 20-13. ECCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Any writes to these bits must always have a value of 0.
7	CTR_CMP	R/W	0h	Counter Equal Compare Status Flag. 0h = Writing a 0 has no effect. Reading always returns 0. 1h = Writing a 1 clears the CTR_CMP flag condition.
6	CTR_PRD	R/W	0h	Counter Equal Period Status Flag. 0h = Writing a 0 has no effect. Reading always returns 0. 1h = Writing a 1 clears the CTR_PRD flag condition.
5	CTROVF	R/W	0h	Counter Overflow Status Flag. 0h = Writing a 0 has no effect. Reading always returns 0. 1h = Writing a 1 clears the CTROVF flag condition.
4	CEVT4	R/W	0h	Capture Event 4 Status Flag. 0h = Writing a 0 has no effect. Reading always returns 0. 1h = Writing a 1 clears the CEVT4 flag condition.
3	CEVT3	R/W	0h	Capture Event 3 Status Flag. 0h = Writing a 0 has no effect. Reading always returns 0. 1h = Writing a 1 clears the CEVT3 flag condition.
2	CEVT2	R/W	0h	Capture Event 2 Status Flag. 0h = Writing a 0 has no effect. Reading always returns 0. 1h = Writing a 1 clears the CEVT2 flag condition.
1	CEVT1	R/W	0h	Capture Event 1 Status Flag. 0h = Writing a 0 has no effect. Reading always returns 0. 1h = Writing a 1 clears the CEVT1 flag condition.
0	INT	R/W	0h	Global Interrupt Clear Flag. 0h = Writing a 0 has no effect. Reading always returns 0. 1h = Writing a 1 clears the INT flag and enable further interrupts to be generated if any of the event flags are set to 1.

20.5.12 ECFRC Register (Offset = 32h) [reset = 0h]

ECFRC is shown in Figure 20-20 and described in Table 20-14.

Return to Summary Table.

Figure 20-20. ECFRC Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							

7	6	5	4	3	2	1	0
CTR_CMP	CTR_PRD	CTROVF	CEVT4	CEVT3	CEVT2	CEVT1	RESERVED
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R-0h

Table 20-14. ECFRC Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Any writes to these bits must always have a value of 0.
7	CTR_CMP	R/W	0h	Force Counter Equal Compare Interrupt. 0h = Writing a 0 has no effect. Reading always returns 0. 1h = Writing a 1 sets the CTR_CMP flag bit.
6	CTR_PRD	R/W	0h	Force Counter Equal Period Interrupt. 0h = Writing a 0 has no effect. Reading always returns 0. 1h = Writing a 1 sets the CTR_PRD flag bit.
5	CTROVF	R/W	0h	Force Counter Overflow. 0h = Writing a 0 has no effect. Reading always returns 0. 1h = Writing a 1 to this bit sets the CTROVF flag bit.
4	CEVT4	R/W	0h	Force Capture Event 4. 0h = Writing a 0 has no effect. Reading always returns 0. 1h = Writing a 1 sets the CEVT4 flag bit
3	CEVT3	R/W	0h	Force Capture Event 3. 0h = Writing a 0 has no effect. Reading always returns 0. 1h = Writing a 1 sets the CEVT3 flag bit
2	CEVT2	R/W	0h	Force Capture Event 2. 0h = Writing a 0 has no effect. Reading always returns 0. 1h = Writing a 1 sets the CEVT2 flag bit.
1	CEVT1	R/W	0h	Force Capture Event 1. 0h = Writing a 0 has no effect. Reading always returns 0. 1h = Sets the CEVT1 flag bit.
0	RESERVED	R	0h	Any writes to these bits must always have a value of 0.

20.6 Application of the ECAP Module

The following sections will provide applications examples and code snippets to show how to configure and operate the eCAP module. For clarity and ease of use, the examples use the eCAP “C” header files. Below are useful #defines that will help in the understanding of the examples.

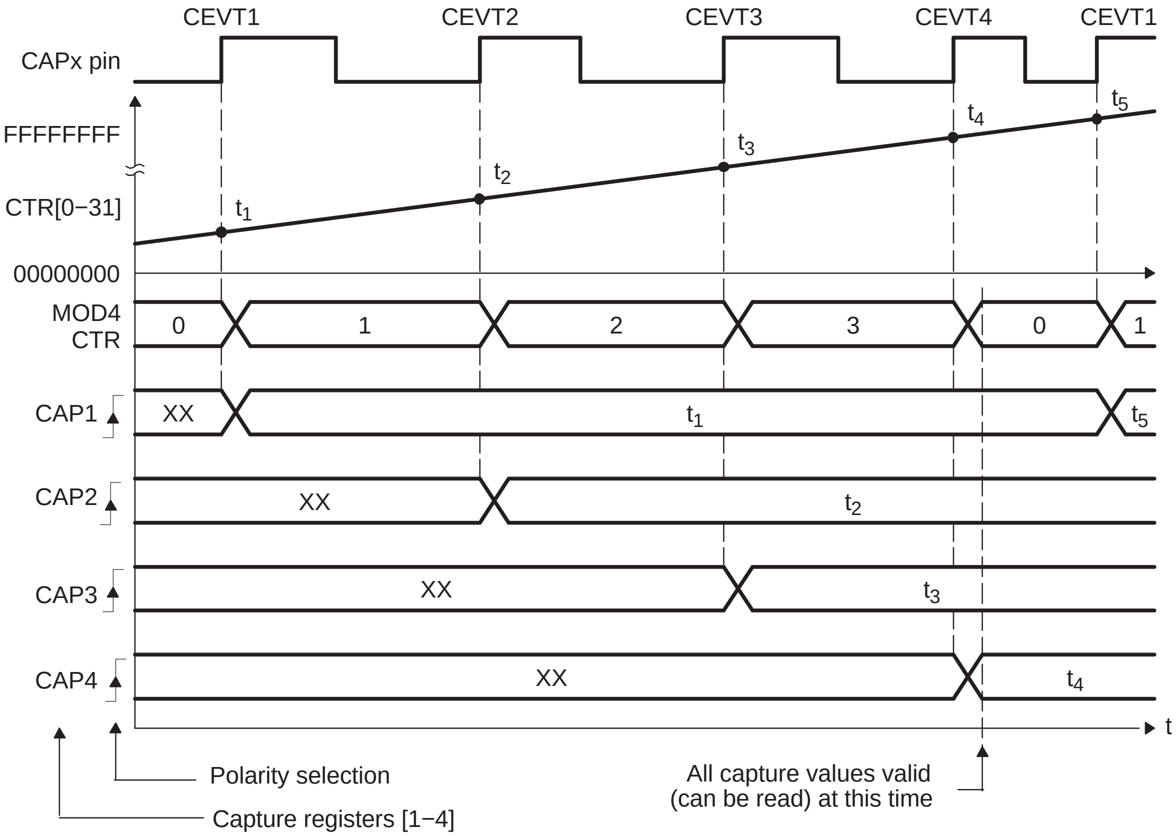
```
// ECCTL1 ( ECAP Control Reg 1)
 //==========================
 // CAPxPOL bits
 #define EC_RISING 0x0
 #define EC_FALLING 0x1
 // CTRRSTx bits
 #define EC_ABS_MODE 0x0
 #define EC_DELTA_MODE 0x1
 // PRESCALE bits
 #define EC_BYPASS 0x0
 #define EC_DIV1 0x0
 #define EC_DIV2 0x1
 #define EC_DIV4 0x2
 #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
```

20.6.1 Example 1 - Absolute Time-Stamp Operation Rising Edge Trigger

Figure 20-21 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 (that is, time-stamp) is first captured, then Mod4 counter is incremented to the next state. When the TSCTR reaches FFFF FFFFh (that is, maximum value), it wraps around to 0000 0000h (not shown in Figure 20-21), 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, that is, after the 4th event, hence event CEVT4 can conveniently be used to trigger an interrupt and the CPU can read data from the CAPx registers.

Figure 20-21. Capture Sequence for Absolute Time-stamp and Rising Edge Detect



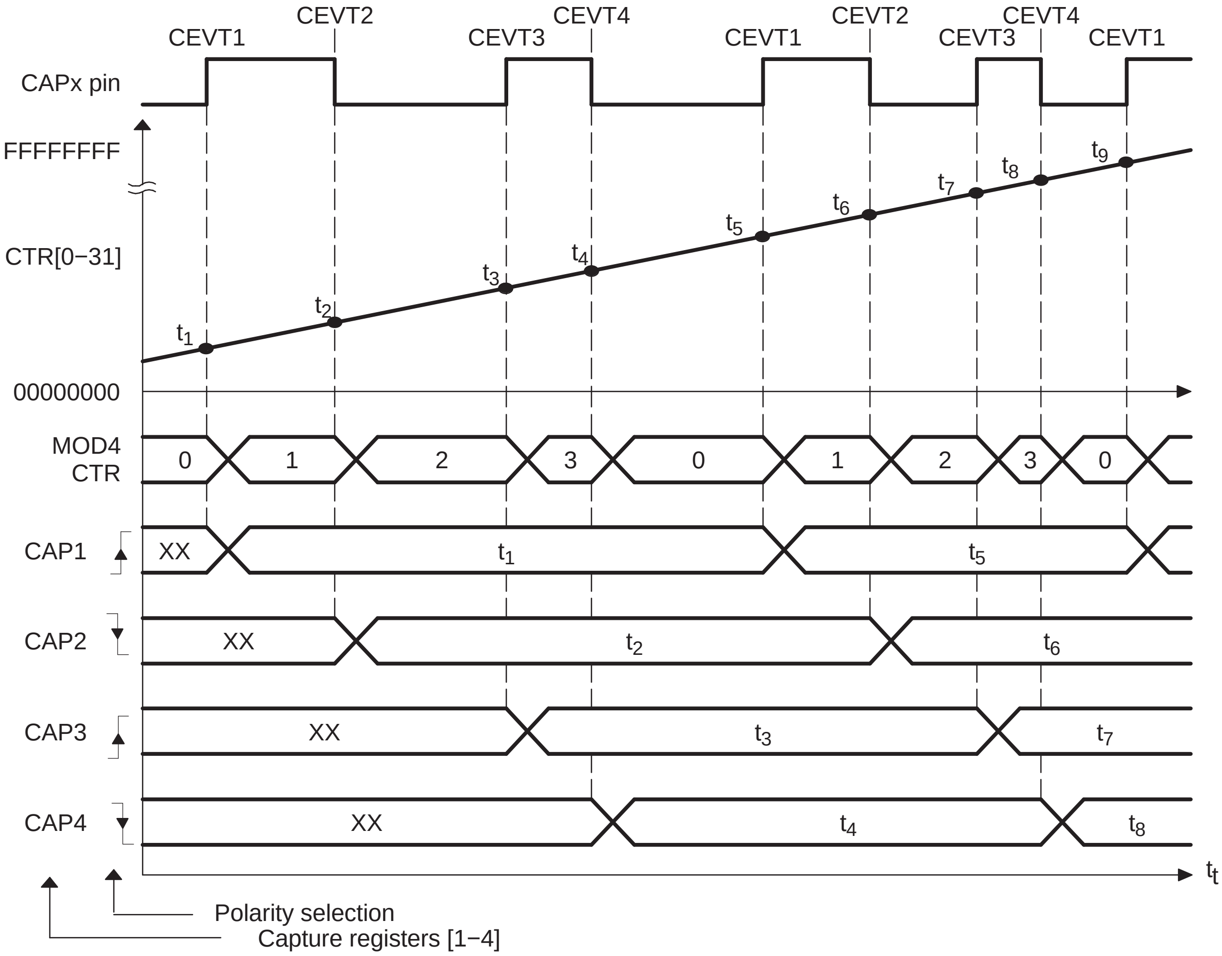
20.6.1.1 Code snippet for CAP mode Absolute Time, Rising Edge Trigger

```
// Code snippet for CAP mode Absolute Time, Rising edge trigger
 // Initialization Time
 //=======================
 // ECAP module 1 config
 ECap1Regs.ECCTL1.bit.CAP1POL = EC_RISING;
 ECap1Regs.ECCTL1.bit.CAP2POL = EC_RISING;
 ECap1Regs.ECCTL1.bit.CAP3POL = EC_RISING;
 ECap1Regs.ECCTL1.bit.CAP4POL = EC_RISING;
 ECap1Regs.ECCTL1.bit.CTRRST1 = EC_ABS_MODE;
 ECap1Regs.ECCTL1.bit.CTRRST2 = EC_ABS_MODE;
 ECap1Regs.ECCTL1.bit.CTRRST3 = EC_ABS_MODE;
 ECap1Regs.ECCTL1.bit.CTRRST4 = EC_ABS_MODE;
 ECap1Regs.ECCTL1.bit.CAPLDEN = EC_ENABLE;
 ECap1Regs.ECCTL1.bit.PRESCALE = EC_DIV1;
 ECap1Regs.ECCTL2.bit.CAP_APWM = EC_CAP_MODE;
 ECap1Regs.ECCTL2.bit.CONT_ONESHT = EC_CONTINUOUS;
 ECap1Regs.ECCTL2.bit.SYNCO_SEL = EC_SYNCO_DIS;
 ECap1Regs.ECCTL2.bit.SYNCI_EN = EC_DISABLE;
 ECap1Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN; // Allow TSCTR to run
 // Run Time ( e.g. CEVT4 triggered ISR call)
 //==========================================
 TSt1 = ECap1Regs.CAP1; // Fetch Time-Stamp captured at t1
 TSt2 = ECap1Regs.CAP2; // Fetch Time-Stamp captured at t2
 TSt3 = ECap1Regs.CAP3; // Fetch Time-Stamp captured at t3
 TSt4 = ECap1Regs.CAP4; // Fetch Time-Stamp captured at t4
 Period1 = TSt2-TSt1; // Calculate 1st period
 Period2 = TSt3-TSt2; // Calculate 2nd period
 Period3 = TSt4-TSt3; // Calculate 3rd period
```

20.6.2 Example 2 - Absolute Time-Stamp Operation Rising and Falling Edge Trigger

In Figure 20-22 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, that is, Period1 = $t_3 - t_1$, Period2 = $t_5 - t_3$, and so on; Duty Cycle1 (on-time %) = ($t_2 - t_1$) / Period1 × 100%, and so on; Duty Cycle1 (off-time %) = ($t_3 - t_2$) / Period1 × 100%, and so on.

Figure 20-22. Capture Sequence for Absolute Time-stamp With Rising and Falling Edge Detect



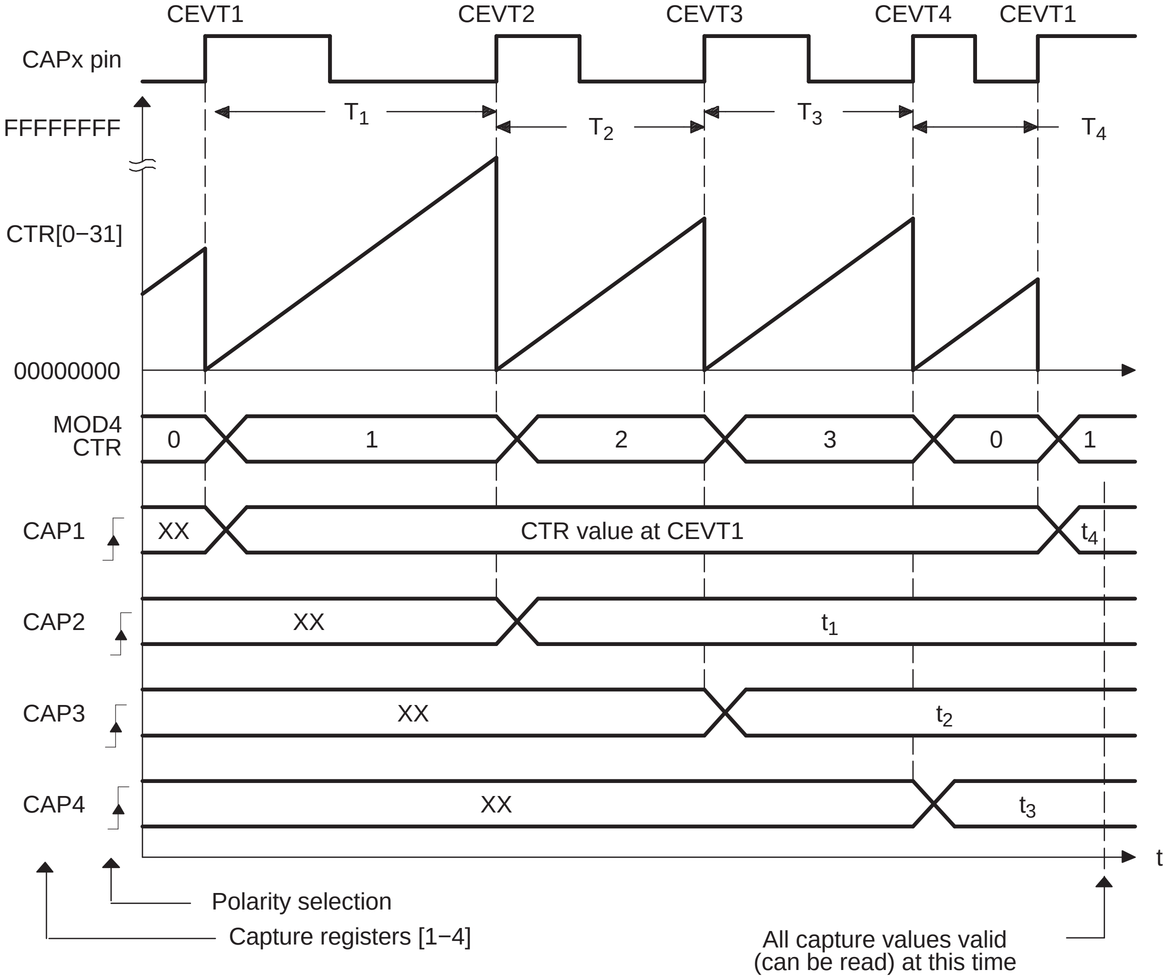
20.6.2.1 Code snippet for CAP mode Absolute Time, Rising & Falling Edge Triggers

```
// Code snippet for CAP mode Absolute Time, Rising & Falling edge triggers
 // Initialization Time
 //=======================
 // ECAP module 1 config
 ECap1Regs.ECCTL1.bit.CAP1POL = EC_RISING;
 ECap1Regs.ECCTL1.bit.CAP2POL = EC_FALLING;
 ECap1Regs.ECCTL1.bit.CAP3POL = EC_RISING;
 ECap1Regs.ECCTL1.bit.CAP4POL = EC_FALLING;
 ECap1Regs.ECCTL1.bit.CTRRST1 = EC_ABS_MODE;
 ECap1Regs.ECCTL1.bit.CTRRST2 = EC_ABS_MODE;
 ECap1Regs.ECCTL1.bit.CTRRST3 = EC_ABS_MODE;
 ECap1Regs.ECCTL1.bit.CTRRST4 = EC_ABS_MODE;
 ECap1Regs.ECCTL1.bit.CAPLDEN = EC_ENABLE;
 ECap1Regs.ECCTL1.bit.PRESCALE = EC_DIV1;
 ECap1Regs.ECCTL2.bit.CAP_APWM = EC_CAP_MODE;
 ECap1Regs.ECCTL2.bit.CONT_ONESHT = EC_CONTINUOUS;
 ECap1Regs.ECCTL2.bit.SYNCO_SEL = EC_SYNCO_DIS;
 ECap1Regs.ECCTL2.bit.SYNCI_EN = EC_DISABLE;
 ECap1Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN; // Allow TSCTR to run
 // Run Time ( e.g. CEVT4 triggered ISR call)
 //==========================================
 TSt1 = ECap1Regs.CAP1; // Fetch Time-Stamp captured at t1
 TSt2 = ECap1Regs.CAP2; // Fetch Time-Stamp captured at t2
 TSt3 = ECap1Regs.CAP3; // Fetch Time-Stamp captured at t3
 TSt4 = ECap1Regs.CAP4; // Fetch Time-Stamp captured at t4
 Period1 = TSt3-TSt1; // Calculate 1st period
 DutyOnTime1 = TSt2-TSt1; // Calculate On time
 DutyOffTime1 = TSt3-TSt2; // Calculate Off time
```

20.6.3 Example 3 - Time Difference (Delta) Operation Rising Edge Trigger

This example Figure 20-23 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 (that is, 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 (that is, 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 CAPx contents directly give timing data without the need for CPU calculations, that is, Period1 = T_1, Period2 = T_2, and so on. As shown in the diagram, the CEVT1 event is a good trigger point to read the timing data, T_1, T_2, T_3, T_4 are all valid here.

Figure 20-23. Capture Sequence for Delta Mode Time-stamp and Rising Edge Detect



20.6.3.1 Code snippet for CAP mode Delta Time, Rising Edge Trigger

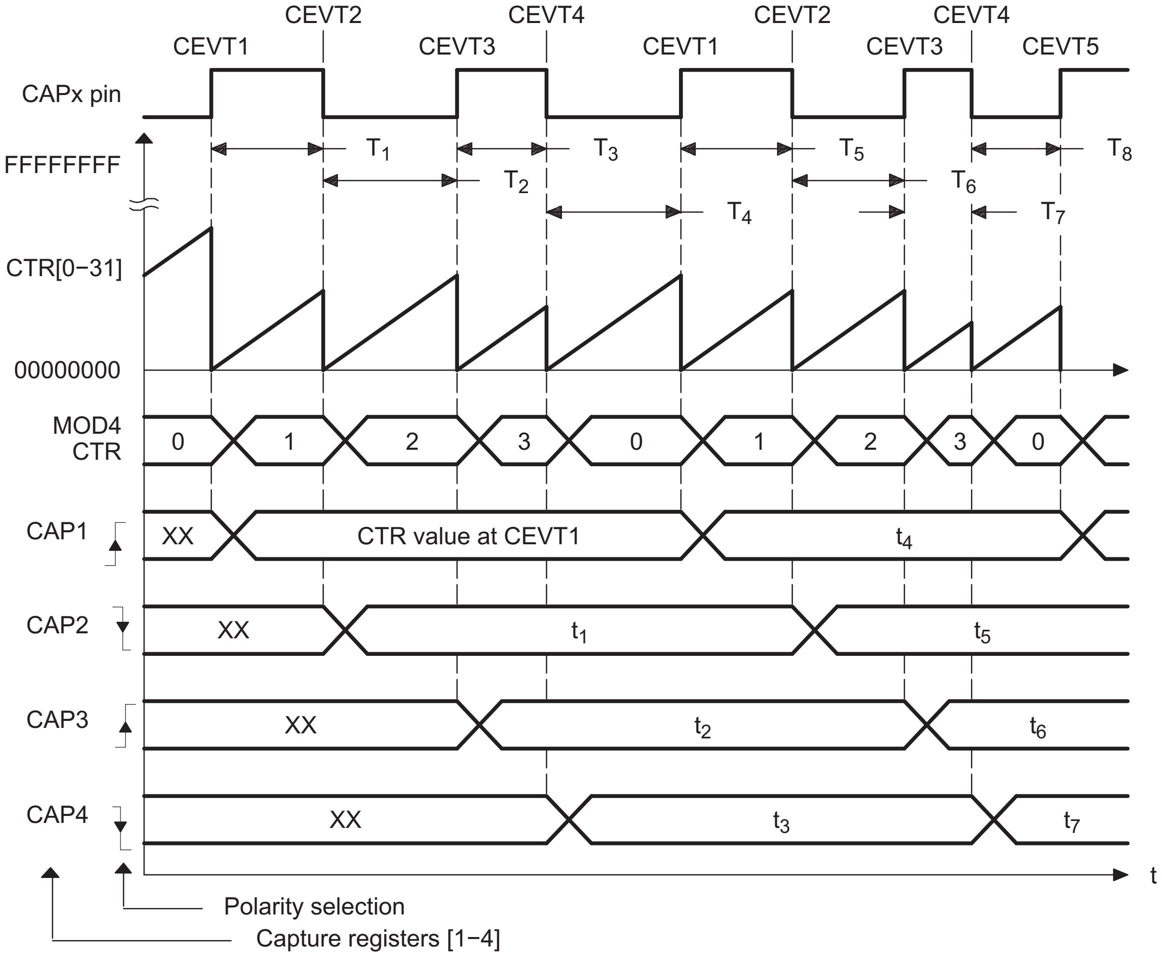
```
// Code snippet for CAP mode Delta Time, Rising edge trigger
 // Initialization Time
 //=======================
 // ECAP module 1 config
 ECap1Regs.ECCTL1.bit.CAP1POL = EC_RISING;
 ECap1Regs.ECCTL1.bit.CAP2POL = EC_RISING;
 ECap1Regs.ECCTL1.bit.CAP3POL = EC_RISING;
 ECap1Regs.ECCTL1.bit.CAP4POL = EC_RISING;
 ECap1Regs.ECCTL1.bit.CTRRST1 = EC_DELTA_MODE;
 ECap1Regs.ECCTL1.bit.CTRRST2 = EC_DELTA_MODE;
 ECap1Regs.ECCTL1.bit.CTRRST3 = EC_DELTA_MODE;
 ECap1Regs.ECCTL1.bit.CTRRST4 = EC_DELTA_MODE;
 ECap1Regs.ECCTL1.bit.CAPLDEN = EC_ENABLE;
 ECap1Regs.ECCTL1.bit.PRESCALE = EC_DIV1;
 ECap1Regs.ECCTL2.bit.CAP_APWM = EC_CAP_MODE;
 ECap1Regs.ECCTL2.bit.CONT_ONESHT = EC_CONTINUOUS;
 ECap1Regs.ECCTL2.bit.SYNCO_SEL = EC_SYNCO_DIS;
 ECap1Regs.ECCTL2.bit.SYNCI_EN = EC_DISABLE;
 ECap1Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN; // Allow TSCTR to run
 // Run Time ( e.g. CEVT1 triggered ISR call)
 //==========================================
 // Note: here Time-stamp directly represents the Period value.
 Period4 = ECap1Regs.CAP1; // Fetch Time-Stamp captured at T1
 Period1 = ECap1Regs.CAP2; // Fetch Time-Stamp captured at T2
 Period2 = ECap1Regs.CAP3; // Fetch Time-Stamp captured at T3
 Period3 = ECap1Regs.CAP4; // Fetch Time-Stamp captured at T4
```

20.6.4 Example 4 - Time Difference (Delta) Operation Rising and Falling Edge Trigger

In Figure 20-24 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, that is, Period1 = T_1+T_2, Period2 = T_3+T_4, and so on; Duty Cycle1 (on-time %) = T_1 / Period1 × 100%, and so on; Duty Cycle1 (off-time %) = T_2 / Period1 × 100%, and so on.

During initialization, you must write to the active registers for both period and compare. This will then automatically copy the init values into the shadow values. For subsequent compare updates, during run-time, only the shadow registers must be used.

Figure 20-24. Capture Sequence for Delta Mode Time-stamp With Rising and Falling Edge Detect



20.6.4.1 Code snippet for CAP mode Delta Time, Rising and Falling Edge Triggers

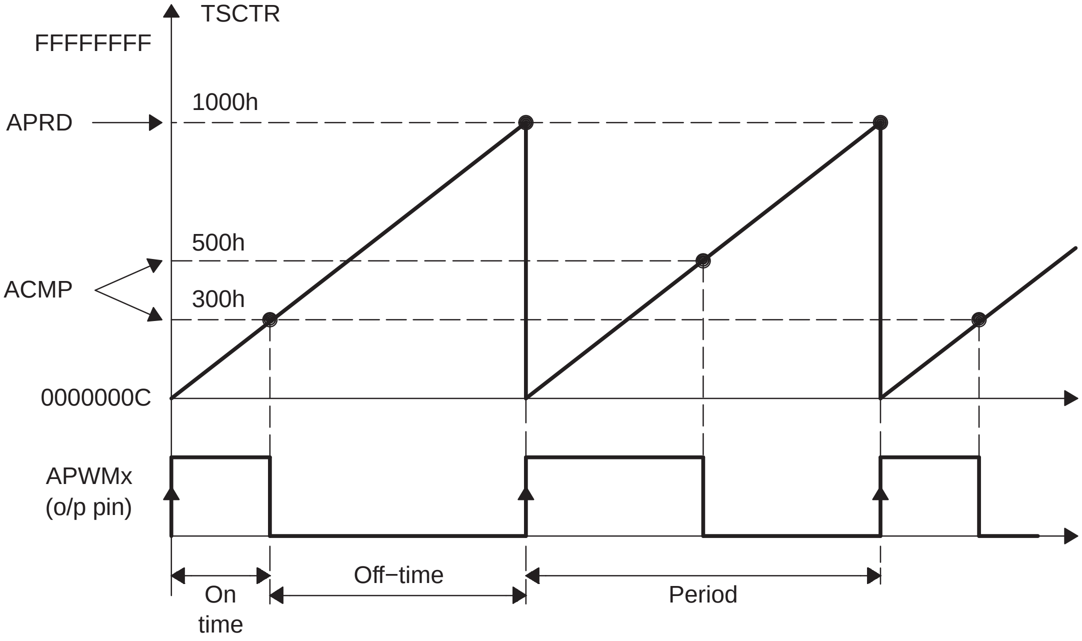
```
// Code snippet for CAP mode Delta Time, Rising and Falling edge triggers
 // Initialization Time
 //=======================
 // ECAP module 1 config
 ECap1Regs.ECCTL1.bit.CAP1POL = EC_RISING;
 ECap1Regs.ECCTL1.bit.CAP2POL = EC_FALLING;
 ECap1Regs.ECCTL1.bit.CAP3POL = EC_RISING;
 ECap1Regs.ECCTL1.bit.CAP4POL = EC_FALLING;
 ECap1Regs.ECCTL1.bit.CTRRST1 = EC_DELTA_MODE;
 ECap1Regs.ECCTL1.bit.CTRRST2 = EC_DELTA_MODE;
 ECap1Regs.ECCTL1.bit.CTRRST3 = EC_DELTA_MODE;
 ECap1Regs.ECCTL1.bit.CTRRST4 = EC_DELTA_MODE;
 ECap1Regs.ECCTL1.bit.CAPLDEN = EC_ENABLE;
 ECap1Regs.ECCTL1.bit.PRESCALE = EC_DIV1;
 ECap1Regs.ECCTL2.bit.CAP_APWM = EC_CAP_MODE;
 ECap1Regs.ECCTL2.bit.CONT_ONESHT = EC_CONTINUOUS;
 ECap1Regs.ECCTL2.bit.SYNCO_SEL = EC_SYNCO_DIS;
 ECap1Regs.ECCTL2.bit.SYNCI_EN = EC_DISABLE;
 ECap1Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN; // Allow TSCTR to run
 // Run Time ( e.g. CEVT1 triggered ISR call)
 //==========================================
 // Note: here Time-stamp directly represents the Duty cycle values.
 DutyOnTime1 = ECap1Regs.CAP2; // Fetch Time-Stamp captured at T2
 DutyOffTime1 = ECap1Regs.CAP3; // Fetch Time-Stamp captured at T3
 DutyOnTime2 = ECap1Regs.CAP4; // Fetch Time-Stamp captured at T4
 DutyOffTime2 = ECap1Regs.CAP1; // Fetch Time-Stamp captured at T1
 Period1 = DutyOnTime1 + DutyOffTime1;
 Period2 = DutyOnTime2 + DutyOffTime2;
```

20.7 Application of the APWM Mode

In this example, the eCAP module is configured to operate as a PWM generator. Here a very simple single channel PWM waveform is generated from output pin APWMx. The PWM polarity is active high, which means that the compare value (CAP2 reg is now a compare register) represents the on-time (high level) of the period. Alternatively, if the APWMPOL bit is configured for active low, then the compare value represents the off-time. Note here values are in hexadecimal (h) notation.

20.7.1 Simple PWM Generation (Independent Channel/s)

Figure 20-25. PWM Waveform Details of APWM Mode Operation



Example 20-1. Code Snippet for APWM Mode

```
// Code snippet for APWM mode Example 1
 // Initialization Time
 //=======================
 // ECAP module 1 config
 ECap1Regs.CAP1 = 0x1000; // Set period value
 ECap1Regs.CTRPHS = 0x0; // make phase zero
 ECap1Regs.ECCTL2.bit.CAP_APWM = EC_APWM_MODE;
 ECap1Regs.ECCTL2.bit.APWMPOL = EC_ACTV_HI; // Active high
 ECap1Regs.ECCTL2.bit.SYNCI_EN = EC_DISABLE; // Synch not used
 ECap1Regs.ECCTL2.bit.SYNCO_SEL = EC_SYNCO_DIS; // Synch not used
 ECap1Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN; // Allow TSCTR to run
 // Run Time (Instant 1, e.g. ISR call)
 //======================
 ECap1Regs.CAP2 = 0x300; // Set Duty cycle i.e. compare value
 // Run Time (Instant 2, e.g. another ISR call)
 //======================
 ECap1Regs.CAP2 = 0x500; // Set Duty cycle i.e. compare value
```

Chapter 21

SPNU514C – March 2018

Enhanced Quadrature Encoder Pulse (eQEP) Module

The enhanced quadrature encoder pulse (eQEP) module is used for direct interface with a linear or rotary incremental encoder to get position, direction, and speed information from a rotating machine for use in a high-performance motion and position-control system. This microcontroller implements 2 instances of the eQEP module.

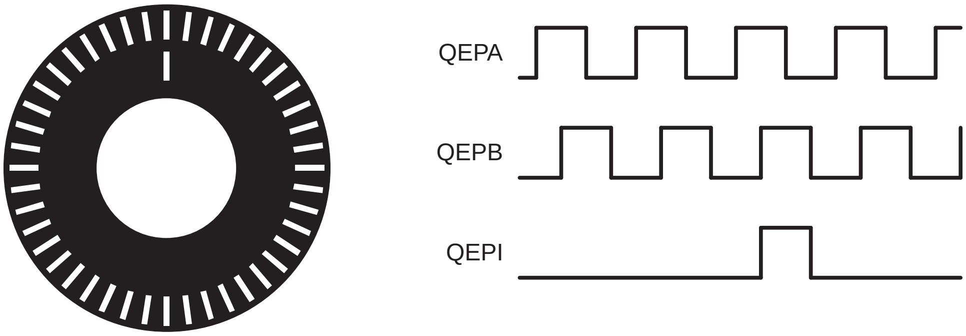
Topic **Page**

21.1 Introduction 886
21.2 Description 888
21.3 Quadrature Decoder Unit (QDU) 891
21.4 Position Counter and Control Unit (PCCU) 894
21.5 eQEP Edge Capture Unit 901
21.6 eQEP Watchdog 904
21.7 Unit Timer Base 904
21.8 eQEP Interrupt Structure 905
21.9 EQEP Registers 906

21.1 Introduction

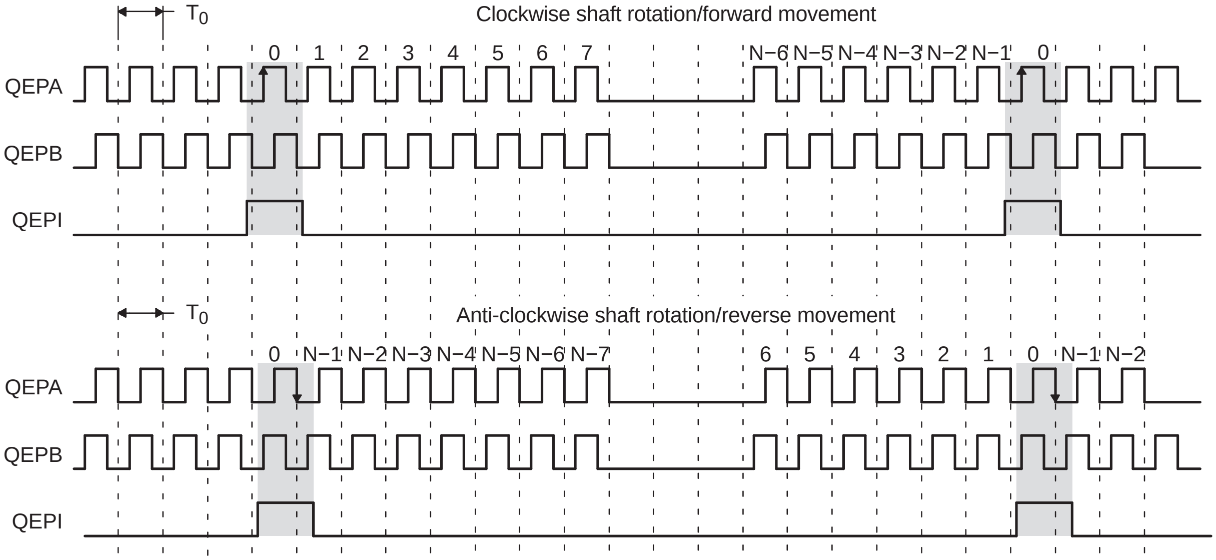
A single track of slots patterns the periphery of an incremental encoder disk, as shown in Figure 21-1. These slots create an alternating pattern of dark and light lines. The disk count is defined as the number of dark/light line pairs that occur per revolution (lines per revolution). As a rule, a second track is added to generate a signal that occurs once per revolution (index signal: QEPI), which can be used to indicate an absolute position. Encoder manufacturers identify the index pulse using different terms such as index, marker, home position, and zero reference

Figure 21-1. Optical Encoder Disk



To derive direction information, the lines on the disk are read out by two different photo-elements that "look" at the disk pattern with a mechanical shift of 1/4 the pitch of a line pair between them. This shift is realized with a reticle or mask that restricts the view of the photo-element to the desired part of the disk lines. As the disk rotates, the two photo-elements generate signals that are shifted 90° out of phase from each other. These are commonly called the quadrature QEPA and QEPB signals. The clockwise direction for most encoders is defined as the QEPA channel going positive before the QEPB channel and vise versa as shown in Figure 21-2.

Figure 21-2. QEP Encoder Output Signal for Forward/Reverse Movement

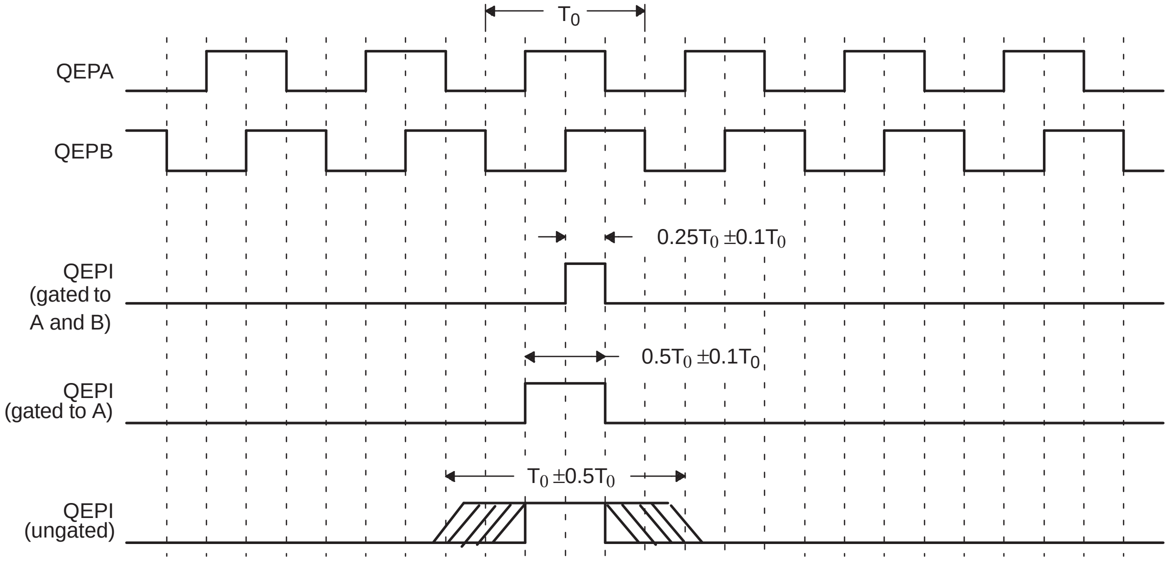

T0
Clockwise shaft rotation/forward movement
0 1 2 3 4 5 6 7 N−6 N−5 N−4 N−3 N−2 N−1 0
QEPA
QEPB
QEPI
T0
Anti-clockwise shaft rotation/reverse movement
0 N−1 N−2 N−3 N−4 N−5 N−6 N−7 6 5 4 3 2 1 0 N−1 N−2
QEPA
QEPB
QEPI

Legend: N = lines per revolution

The encoder wheel typically makes one revolution for every revolution of the motor or the wheel may be at a geared rotation ratio with respect to the motor. Therefore, the frequency of the digital signal coming from the QEPA and QEPB outputs varies proportionally with the velocity of the motor. For example, a 2000-line encoder directly coupled to a motor running at 5000 revolutions per minute (rpm) results in a frequency of 166.6 KHz, so by measuring the frequency of either the QEPA or QEPB output, the processor can determine the velocity of the motor.

Quadrature encoders from different manufacturers come with two forms of index pulse (gated index pulse or ungated index pulse) as shown in Figure 21-3. A nonstandard form of index pulse is ungated. In the ungated configuration, the index edges are not necessarily coincident with A and B signals. The gated index pulse is aligned to any of the four quadrature edges and width of the index pulse and can be equal to a quarter, half, or full period of the quadrature signal.

Figure 21-3. Index Pulse Example



Some typical applications of shaft encoders include robotics and even computer input in the form of a mouse. Inside your mouse you can see where the mouse ball spins a pair of axles (a left/right, and an up/down axle). These axles are connected to optical shaft encoders that effectively tell the computer how fast and in what direction the mouse is moving.

General Issues: Estimating velocity from a digital position sensor is a cost-effective strategy in motor control. Two different first order approximations for velocity may be written as:

$$v(k) \approx \frac{x(k) - x(k-1)}{T} = \frac{\Delta X}{T} \qquad (27)$$

$$v(k) \approx \frac{X}{t(k) - t(k-1)} = \frac{X}{\Delta T} \qquad (28)$$

where

v(k): Velocity at time instant k

x(k): Position at time instant k

x(k-1): Position at time instant k-1

T: Fixed unit time or inverse of velocity calculation rate

ΔX: Incremental position movement in unit time

t(k): Time instant "k"

t(k-1): Time instant "k-1"

X: Fixed unit position

ΔT: Incremental time elapsed for unit position movement.

Equation 27 is the conventional approach to velocity estimation and it requires a time base to provide unit time event for velocity calculation. Unit time is basically the inverse of the velocity calculation rate.

The encoder count (position) is read once during each unit time event. The quantity [x(k) - x(k-1)] is formed by subtracting the previous reading from the current reading. Then the velocity estimate is computed by multiplying by the known constant 1/T (where T is the constant time between unit time events and is known in advance).

Estimation based on Equation 27 has an inherent accuracy limit directly related to the resolution of the position sensor and the unit time period T. For example, consider a 500-line per revolution quadrature encoder with a velocity calculation rate of 400 Hz. When used for position the quadrature encoder gives a four-fold increase in resolution, in this case, 2000 counts per revolution. The minimum rotation that can be detected is therefore 0.0005 revolutions, which gives a velocity resolution of 12 rpm when sampled at 400 Hz. While this resolution may be satisfactory at moderate or high speeds, for example, 1% error at 1200 rpm, it would clearly prove inadequate at low speeds. In fact, at speeds below 12 rpm, the speed estimate would erroneously be zero much of the time.

At low speed, Equation 28 provides a more accurate approach. It requires a position sensor that outputs a fixed interval pulse train, such as the aforementioned quadrature encoder. The width of each pulse is defined by motor speed for a given sensor resolution. Equation 28 can be used to calculate motor speed by measuring the elapsed time between successive quadrature pulse edges. However, this method suffers from the opposite limitation, as does Equation 27. A combination of relatively large motor speeds and high sensor resolution makes the time interval ΔT small, and thus more greatly influenced by the timer resolution. This can introduce considerable error into high-speed estimates.

For systems with a large speed range (that is, speed estimation is needed at both low and high speeds), one approach is to use Equation 28 at low speed and have the DSP software switch over to Equation 27 when the motor speed rises above some specified threshold.

21.2 Description

This section provides the eQEP inputs, memory map, and functional description.

21.2.1 EQEP Inputs

The eQEP inputs include two pins for quadrature-clock mode or direction-count mode, an index (or 0 marker), and a strobe input.

- *QEPA/XCLK and QEPB/XDIR*

  These two pins can be used in quadrature-clock mode or direction-count mode.

  – *Quadrature-clock Mode*

    The eQEP encoders provide two square wave signals (A and B) 90 electrical degrees out of phase whose phase relationship is used to determine the direction of rotation of the input shaft and number of eQEP pulses from the index position to derive the relative position information. For forward or clockwise rotation, QEPA signal leads QEPB signal and vice versa. The quadrature decoder uses these two inputs to generate quadrature-clock and direction signals.

  – *Direction-count Mode*

    In direction-count mode, direction and clock signals are provided directly from the external source. Some position encoders have this type of output instead of quadrature output. The QEPA pin provides the clock input and the QEPB pin provides the direction input.

- *eQEPI: Index or Zero Marker*

  The eQEP encoder uses an index signal to assign an absolute start position from which position information is incrementally encoded using quadrature pulses. This pin is connected to the index output of the eQEP encoder to optionally reset the position counter for each revolution. This signal can be used to initialize or latch the position counter on the occurrence of a desired event on the index pin.

- QEPS: *Strobe Input*

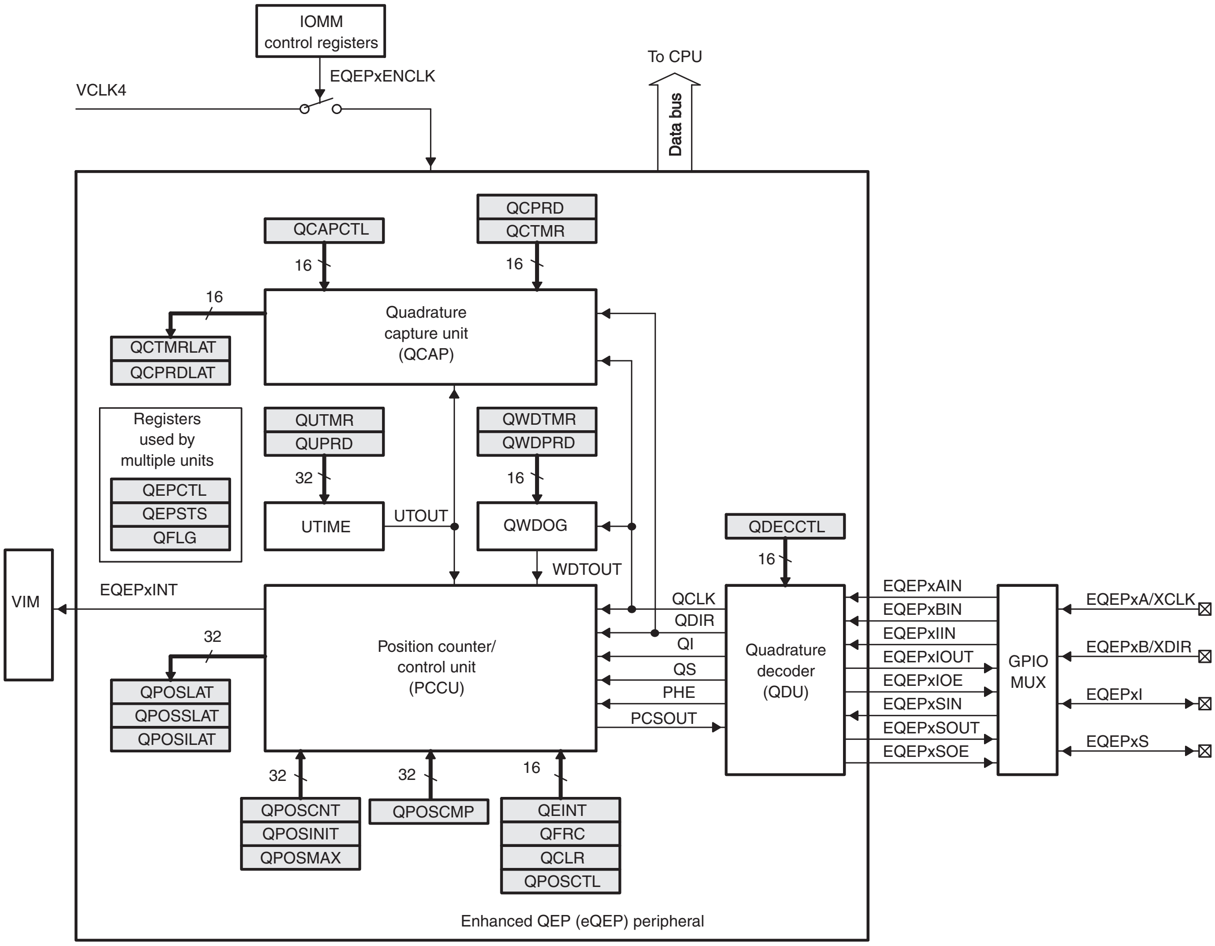
  This general-purpose strobe signal can initialize or latch the position counter on the occurrence of a desired event on the strobe pin. This signal is typically connected to a sensor or limit switch to notify that the motor has reached a defined position.

21.2.2 Functional Description

The eQEP peripheral contains the following major functional units (as shown in Figure 21-4):

- Programmable input qualification for each pin (part of the GPIO MUX)
- Quadrature decoder unit (QDU)
- Position counter and control unit for position measurement (PCCU)
- Quadrature edge-capture unit for low-speed measurement (QCAP)
- Unit time base for speed/frequency measurement (UTIME)
- Watchdog timer for detecting stalls (QWDOG)

Figure 21-4. Functional Block Diagram of the eQEP Peripheral



21.2.3 eQEP Memory Map

Table 21-1 lists the registers with their memory locations, sizes, and reset values.

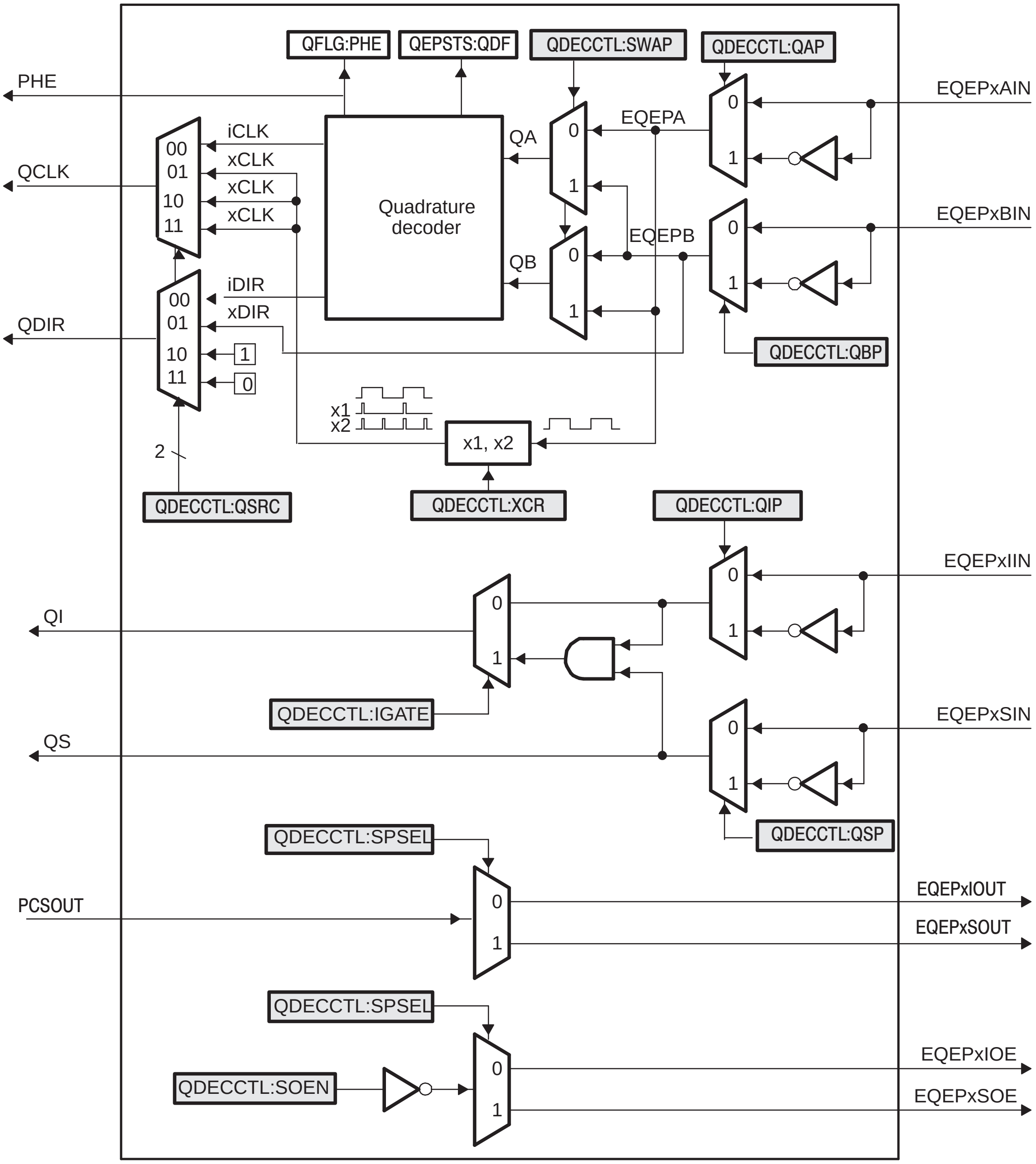
Table 21-1. EQEP Memory Map

Name	Address Offset	Size(x16)/ #shadow	Reset	Register Description
QPOSCNT	0x00	2/0	0x00000000	eQEP Position Counter
QPOSINIT	0x04	2/0	0x00000000	eQEP Initialization Position Count
QPOSMAX	0x08	2/0	0x00000000	eQEP Maximum Position Count
QPOSCMP	0x0C	2/1	0x00000000	eQEP Position-compare
QPOSILAT	0x10	2/0	0x00000000	eQEP Index Position Latch
QPOSSLAT	0x14	2/0	0x00000000	eQEP Strobe Position Latch
QPOSLAT	0x18	2/0	0x00000000	eQEP Position Latch
QUTMR	0x1C	2/0	0x00000000	eQEP Unit Timer
QUPRD	0x20	2/0	0x00000000	eQEP Unit Period Register
QWDTMR	0x24	1/0	0x0000	eQEP Watchdog Timer
QWDPRD	0x26	1/0	0x0000	eQEP Watchdog Period Register
QDECCTL	0x28	1/0	0x0000	eQEP Decoder Control Register
QEPCTL	0x2A	1/0	0x0000	eQEP Control Register
QCAPCTL	0x2C	1/0	0x0000	eQEP Capture Control Register
QPOSCTL	0x2E	1/0	0x00000	eQEP Position-compare Control Register
QEINT	0x30	1/0	0x0000	eQEP Interrupt Enable Register
QFLG	0x32	1/0	0x0000	eQEP Interrupt Flag Register
QCLR	0x34	1/0	0x0000	eQEP Interrupt Clear Register
QFRC	0x36	1/0	0x0000	eQEP Interrupt Force Register
QEPSTS	0x38	1/0	0x0000	eQEP Status Register
QCTMR	0x3A	1/0	0x0000	eQEP Capture Timer
QCPRD	0x3C	1/0	0x0000	eQEP Capture Period Register
QCTMRLAT	0x3E	1/0	0x0000	eQEP Capture Timer Latch
QCPRDLAT	0x40	1/0	0x0000	eQEP Capture Period Latch
Reserved	0x42	–	–	Reserved

21.3 Quadrature Decoder Unit (QDU)

Figure 21-5 shows a functional block diagram of the QDU.

Figure 21-5. Functional Block Diagram of Decoder Unit



21.3.1 Position Counter Input Modes

Clock and direction input to position counter is selected using QDECCTL[QSRC] bits, based on interface input requirement as follows:

- Quadrature-count mode
- Direction-count mode
- UP-count mode
- DOWN-count mode

21.3.1.1 Quadrature Count Mode

The quadrature decoder generates the direction and clock to the position counter in quadrature count mode.

Direction Decoding— The direction decoding logic of the eQEP circuit determines which one of the sequences (QEPA, QEPB) is the leading sequence and accordingly updates the direction information in QEPSTS[QDF] bit. Table 21-2 and Figure 21-6 show the direction decoding logic in truth table and state machine form. Both edges of the QEPA and QEPB signals are sensed to generate count pulses for the position counter. Therefore, the frequency of the clock generated by the eQEP logic is four times that of each input sequence. Figure 21-7 shows the direction decoding and clock generation from the eQEP input signals.

Phase Error Flag— In normal operating conditions, quadrature inputs QEPA and QEPB will be 90 degrees out of phase. The phase error flag (PHE) is set in the QFLG register when edge transition is detected simultaneously on the QEPA and QEPB signals to optionally generate interrupts. State transitions marked by dashed lines in Figure 21-6 are invalid transitions that generate a phase error.

Count Multiplication— The eQEP position counter provides 4x times the resolution of an input clock by generating a quadrature-clock (QCLK) on the rising/falling edges of both eQEP input clocks (QEPA and QEPB) as shown in Figure 21-7.

Reverse Count— In normal quadrature count operation, QEPA input is fed to the QA input of the quadrature decoder and the QEPB input is fed to the QB input of the quadrature decoder. Reverse counting is enabled by setting the SWAP bit in the QDECCTL register. This will swap the input to the quadrature decoder thereby reversing the counting direction.

Table 21-2. Quadrature Decoder Truth Table

Previous Edge	Present Edge	QDIR	QPOSCNT
QA↑	QB↑	UP	Increment
	QB↓	DOWN	Decrement
	QA↓	TOGGLE	Increment or Decrement
QA↓	QB↓	UP	Increment
	QB↑	DOWN	Decrement
	QA↑	TOGGLE	Increment or Decrement
QB↑	QA↑	DOWN	Increment
	QA↓	UP	Decrement
	QB↓	TOGGLE	Increment or Decrement
QB↓	QA↓	DOWN	Increment
	QA↑	UP	Decrement
	QB↑	TOGGLE	Increment or Decrement

Figure 21-6. Quadrature Decoder State Machine

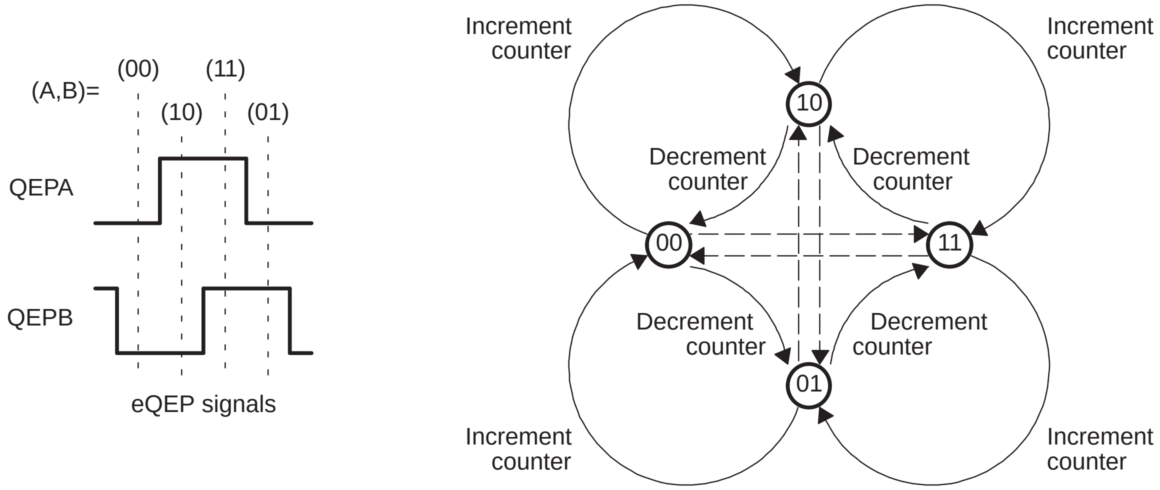
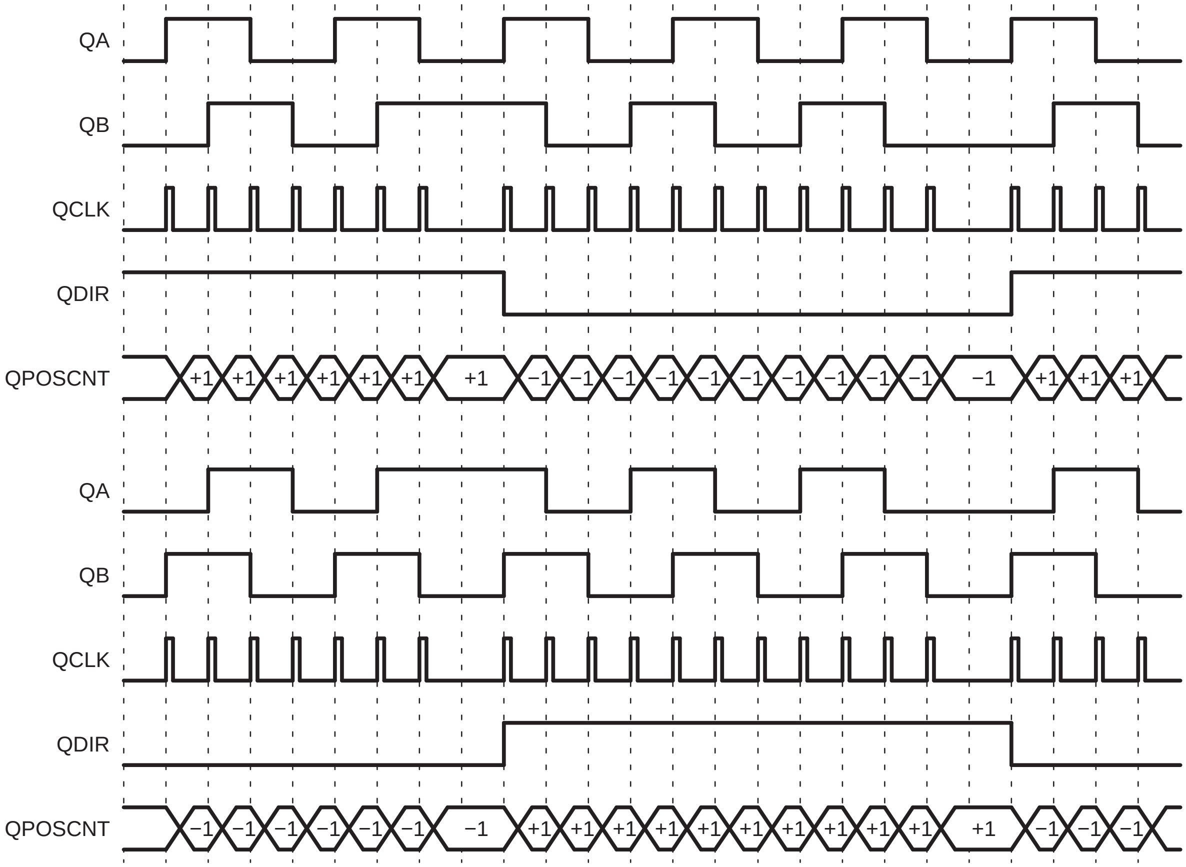


Figure 21-7. Quadrature-clock and Direction Decoding



21.3.1.2 Direction-Count Mode

Some position encoders provide direction and clock outputs, instead of quadrature outputs. In such cases, direction-count mode can be used. QEPA input will provide the clock for position counter and the QEPB input will have the direction information. The position counter is incremented on every rising edge of a QEPA input when the direction input is high and decremented when the direction input is low.

21.3.1.3 Up-Count Mode

The counter direction signal is hard-wired for up count and the position counter is used to measure the frequency of the QEPA input. Setting of the QDECCTL[XCR] bit enables clock generation to the position counter on both edges of the QEPA input, thereby increasing the measurement resolution by 2x factor.

21.3.1.4 Down-Count Mode

The counter direction signal is hardwired for a down count and the position counter is used to measure the frequency of the QEPA input. Setting of the QDECCTL[XCR] bit enables clock generation to the position counter on both edges of a QEPA input, thereby increasing the measurement resolution by 2x factor.

21.3.2 eQEP Input Polarity Selection

Each eQEP input can be inverted using QDECCTL[8:5] control bits. As an example, setting of QDECCTL[QIP] bit will invert the index input.

21.3.3 Position-Compare Sync Output

The enhanced eQEP peripheral includes a position-compare unit that is used to generate the position-compare sync signal on compare match between the position counter register (QPOSCNT) and the position-compare register (QPOSCMP). This sync signal can be output using an index pin or strobe pin of the EQEP peripheral.

Setting the QDECCTL[SOEN] bit enables the position-compare sync output and the QDECCTL[SPSEL] bit selects either an eQEP index pin or an eQEP strobe pin.

21.4 Position Counter and Control Unit (PCCU)

The position counter and control unit provides two configuration registers (QEPCTL and QPOSCTL) for setting up position counter operational modes, position counter initialization/latch modes and position-compare logic for sync signal generation.

21.4.1 Position Counter Operating Modes

Position counter data may be captured in different manners. In some systems, the position counter is accumulated continuously for multiple revolutions and the position counter value provides the position information with respect to the known reference. An example of this is the quadrature encoder mounted on the motor controlling the print head in the printer. Here the position counter is reset by moving the print head to the home position and then position counter provides absolute position information with respect to home position.

In other systems, the position counter is reset on every revolution using index pulse and position counter provides rotor angle with respect to index pulse position.

Position counter can be configured to operate in following four modes

- Position Counter Reset on Index Event
- Position Counter Reset on Maximum Position
- Position Counter Reset on the first Index Event
- Position Counter Reset on Unit Time Out Event (Frequency Measurement)

In all the above operating modes, position counter is reset to 0 on overflow and to QPOSMAX register value on underflow. Overflow occurs when the position counter counts up after QPOSMAX value. Underflow occurs when position counter counts down after "0". Interrupt flag is set to indicate overflow/underflow in QFLG register.

21.4.1.1 Position Counter Reset on Index Event (QEPCTL[PCRM] = 00)

If the index event occurs during the forward movement, then position counter is reset to 0 on the next eQEP clock. If the index event occurs during the reverse movement, then the position counter is reset to the value in the QPOSMAX register on the next eQEP clock.

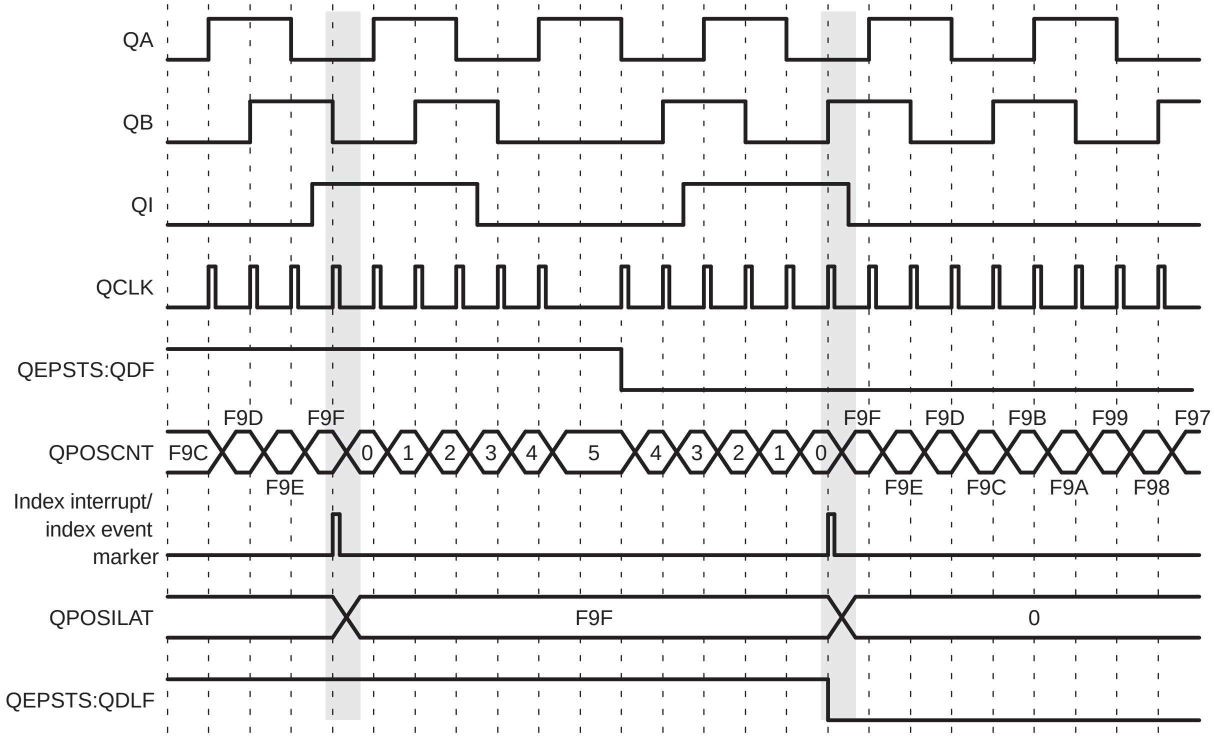
First index marker is defined as the quadrature edge following the first index edge. The eQEP peripheral records the occurrence of the first index marker (QEPSTS[FIMF]) and direction on the first index event marker (QEPSTS[FIDF]) in QEPSTS registers, it also remembers the quadrature edge on the first index marker so that same relative quadrature transition is used for index event reset operation.

For example, if the first reset operation occurs on the falling edge of QEPB during the forward direction, then all the subsequent reset must be aligned with the falling edge of QEPB for the forward rotation and on the rising edge of QEPB for the reverse rotation as shown in Figure 21-8.

The position-counter value is latched to the QPOSILAT register and direction information is recorded in the QEPSTS[QDLF] bit on every index event marker. The position-counter error flag (QEPSTS[PCEF]) and error interrupt flag (QFLG[PCE]) are set if the latched value is not equal to 0 or QPOSMAX. The position-counter error flag (QEPSTS[PCEF]) is updated on every index event marker and an interrupt flag (QFLG[PCE]) will be set on error that can be cleared only through software.

The index event latch configuration QEPCTL[IEL] bits are ignored in this mode and position counter error flag/interrupt flag are generated only in index event reset mode.

Figure 21-8. Position Counter Reset by Index Pulse for 1000 Line Encoder (QPOSMAX = 3999 or 0xF9F)

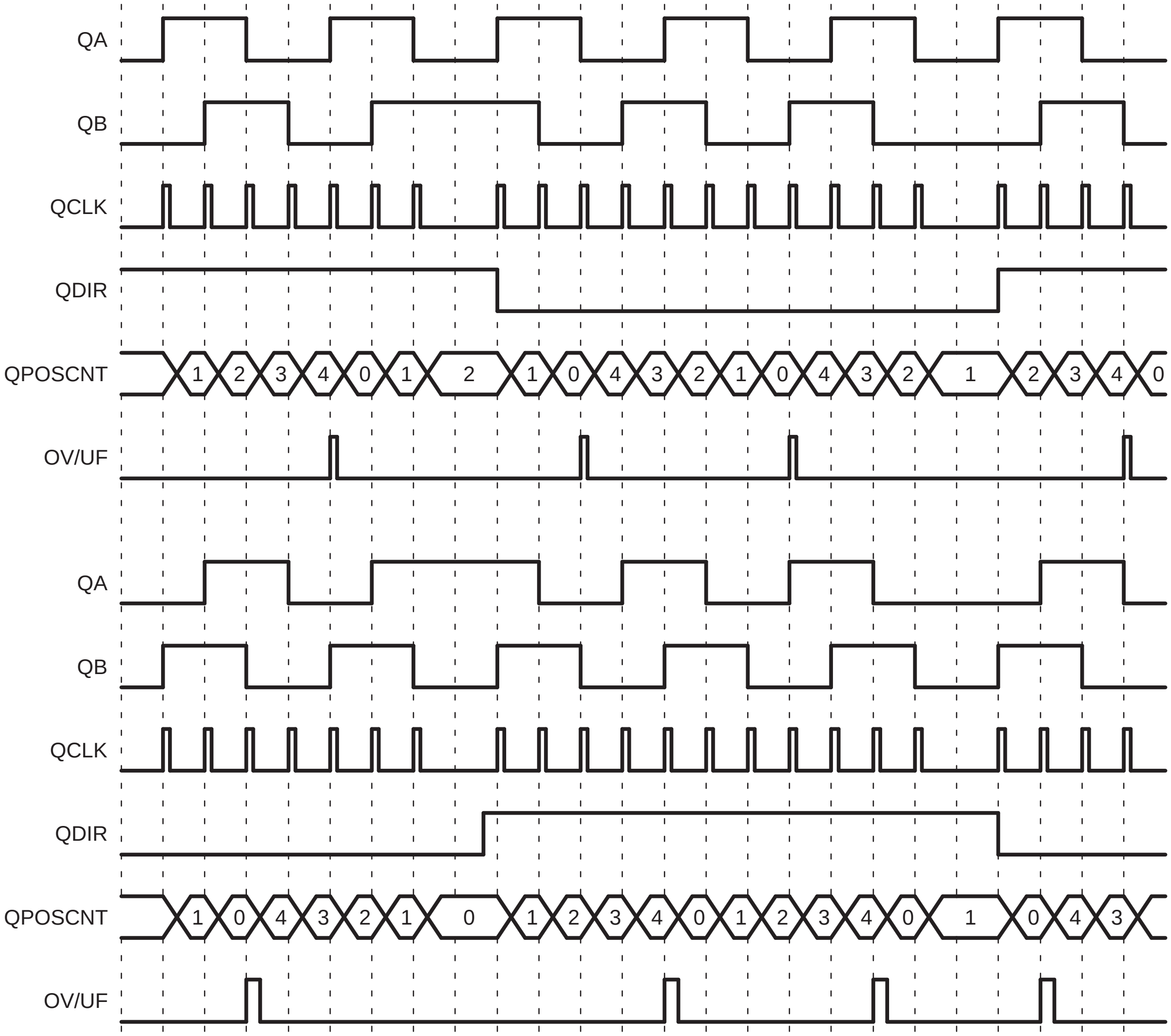


21.4.1.2 Position Counter Reset on Maximum Position (QEPCTL[PCRM] = 01)

If the position counter is equal to QPOSMAX, then the position counter is reset to 0 on the next eQEP clock for forward movement and position counter overflow flag is set. If the position counter is equal to ZERO, then the position counter is reset to QPOSMAX on the next QEP clock for reverse movement and position counter underflow flag is set. Figure 21-9 shows the position counter reset operation in this mode.

First index marker is defined as the quadrature edge following the first index edge. The eQEP peripheral records the occurrence of the first index marker (QEPSTS[FIMF]) and direction on the first index event marker (QEPSTS[FIDF]) in the QEPSTS registers; it also remembers the quadrature edge on the first index marker so that the same relative quadrature transition is used for the software index marker (QEPCTL[IEL]=11).

Figure 21-9. Position Counter Underflow/Overflow (QPOSMAX = 4)



21.4.1.3 Position Counter Reset on the First Index Event (QEPCTL[PCRM] = 10)

If the index event occurs during forward movement, then the position counter is reset to 0 on the next eQEP clock. If the index event occurs during the reverse movement, then the position counter is reset to the value in the QPOSMAX register on the next eQEP clock. Note that this is done only on the first occurrence and subsequently the position counter value is not reset on an index event; rather, it is reset based on maximum position as described in Section 21.4.1.2.

First index marker is defined as the quadrature edge following the first index edge. The eQEP peripheral records the occurrence of the first index marker (QEPSTS[FIMF]) and direction on the first index event marker (QEPSTS[FIDF]) in QEPSTS registers, it also remembers the quadrature edge on the first index marker so that same relative quadrature transition is used for software index marker (QEPCTL[IEL]=11).

21.4.1.4 Position Counter Reset on Unit Time out Event (QEPCTL[PCRM] = 11)

In this mode, the QPOSCNT value is latched to the QPOSLAT register and then the QPOSCNT is reset (to 0 or QPOSMAX, depending on the direction mode selected by QDECCTL[QSRC] bits on a unit time event). This is useful for frequency measurement.

21.4.2 Position Counter Latch

The eQEP index and strobe input can be configured to latch the position counter (QPOSCNT) into QPOSILAT and QPOSSLAT, respectively, on occurrence of a definite event on these pins.

21.4.2.1 Index Event Latch

In some applications, it may not be desirable to reset the position counter on every index event and instead it may be required to operate the position counter in full 32-bit mode (QEPCTL[PCRM] = 01 and QEPCTL[PCRM] = 10 modes).

In such cases, the eQEP position counter can be configured to latch on the following events and direction information is recorded in the QEPSTS[QDLF] bit on every index event marker.

- Latch on Rising edge (QEPCTL[IEL]=01)
- Latch on Falling edge (QEPCTL[IEL]=10)
- Latch on Index Event Marker (QEPCTL[IEL]=11)

This is particularly useful as an error checking mechanism to check if the position counter accumulated the correct number of counts between index events. As an example, the 1000-line encoder must count 4000 times when moving in the same direction between the index events.

The index event latch interrupt flag (QFLG[IEL]) is set when the position counter is latched to the QPOSILAT register. The index event latch configuration bits (QEPCTZ[IEL]) are ignored when QEPCTL[PCRM] = 00.

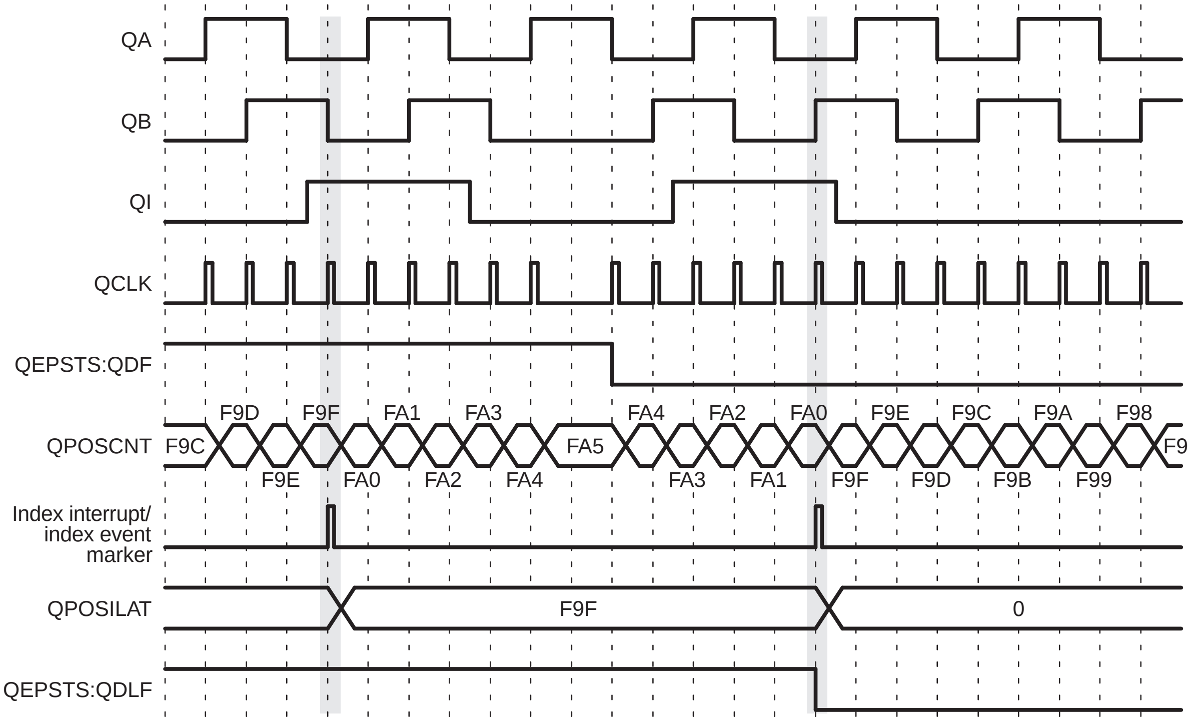
Latch on Rising Edge (QEPCTL[IEL]=01)— The position counter value (QPOSCNT) is latched to the QPOSILAT register on every rising edge of an index input.

Latch on Falling Edge (QEPCTL[IEL] = 10)— The position counter value (QPOSCNT) is latched to the QPOSILAT register on every falling edge of index input.

Latch on Index Event Marker/Software Index Marker (QEPCTL[IEL] = 11— The first index marker is defined as the quadrature edge following the first index edge. The eQEP peripheral records the occurrence of the first index marker (QEPSTS[FIMF]) and direction on the first index event marker (QEPSTS[FIDF]) in the QEPSTS registers. It also remembers the quadrature edge on the first index marker so that same relative quadrature transition is used for latching the position counter (QEPCTL[IEL]=11).

Figure 21-10 shows the position counter latch using an index event marker.

Figure 21-10. Software Index Marker for 1000-line Encoder (QEPCTL[IEL] = 1)



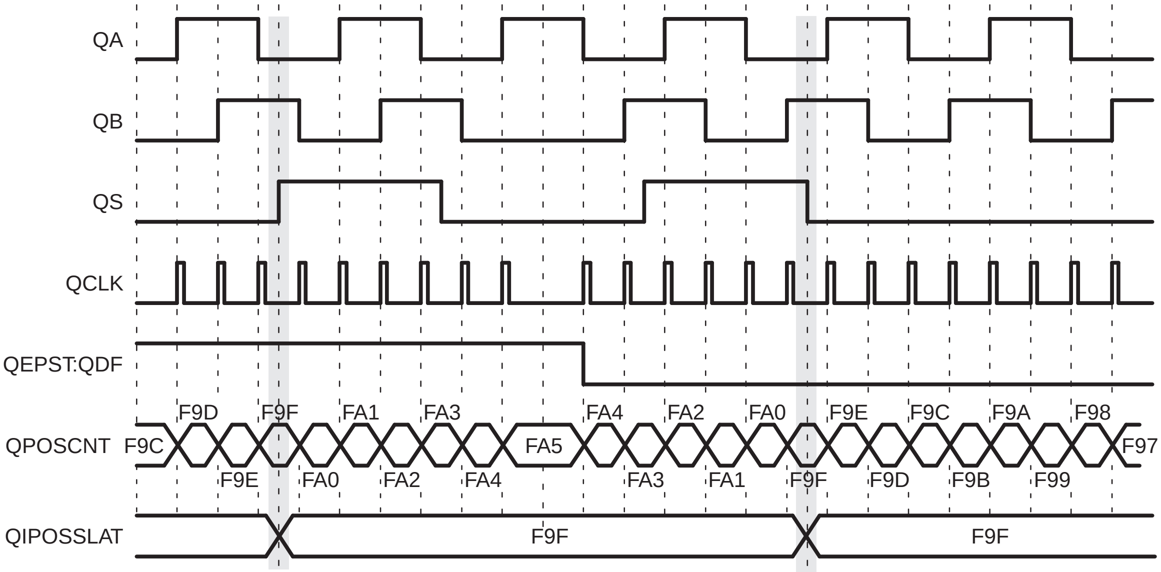
21.4.2.2 Strobe Event Latch

The position-counter value is latched to the QPOSSLAT register on the rising edge of the strobe input by clearing the QEPCTL[SEL] bit.

If the QEPCTL[SEL] bit is set, then the position counter value is latched to the QPOSSLAT register on the rising edge of the strobe input for forward direction and on the falling edge of the strobe input for reverse direction as shown in Figure 21-11.

The strobe event latch interrupt flag (QFLG[SEL) is set when the position counter is latched to the QPOSSLAT register.

Figure 21-11. Strobe Event Latch (QEPCTL[SEL] = 1)



21.4.3 Position Counter Initialization

The position counter can be initialized using following events:

- Index event
- Strobe event
- Software initialization

Index Event Initialization (IEI)— The QEPI index input can be used to trigger the initialization of the position counter at the rising or falling edge of the index input. If the QEPCTL[IEI] bits are 10, then the position counter (QPOSCNT) is initialized with a value in the QPOSINIT register on the rising edge of index input. Conversely, if the QEPCTL[IEI] bits are 11, initialization will be on the falling edge of the index input.

Strobe Event Initialization (SEI)— If the QEPCTL[SEI] bits are 10, then the position counter is initialized with a value in the QPOSINIT register on the rising edge of strobe input.

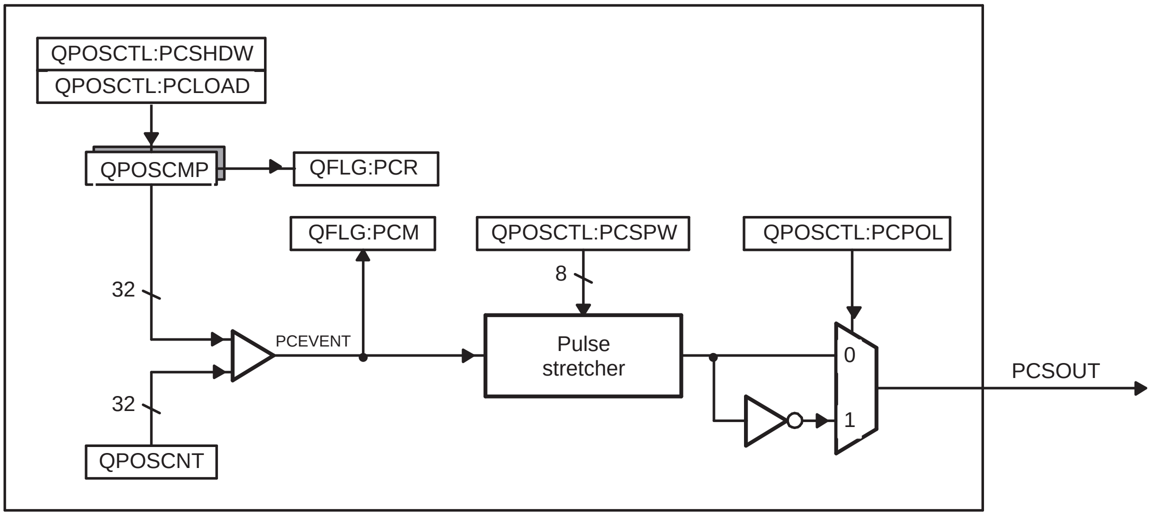
If QEPCTL[SEL] bits are 11, then the position counter is initialized with a value in the QPOSINIT register on the rising edge of strobe input for forward direction and on the falling edge of strobe input for reverse direction.

Software Initialization (SWI)— The position counter can be initialized in software by writing a 1 to the QEPCTL[SWI] bit. This bit is not automatically cleared. While the bit is still set, if a 1 is written to it again, the position counter will be re-initialized.

21.4.4 eQEP Position-compare Unit

The eQEP peripheral includes a position-compare unit that is used to generate a sync output and/or interrupt on a position-compare match. Figure 21-12 shows a diagram. The position-compare (QPOSCMP) register is shadowed and shadow mode can be enabled or disabled using the QPOSCTL[PSSHDW] bit. If the shadow mode is not enabled, the CPU writes directly to the active position compare register.

Figure 21-12. eQEP Position-compare Unit



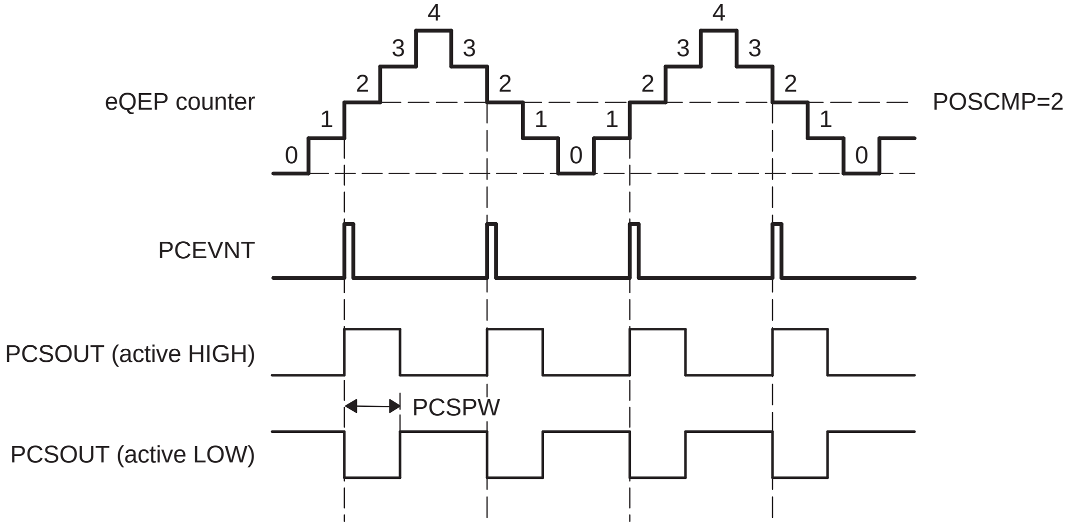
In shadow mode, you can configure the position-compare unit (QPOSCTL[PCLOAD]) to load the shadow register value into the active register on the following events and to generate the position-compare ready (QFLG[PCR]) interrupt after loading.

- Load on compare match
- Load on position-counter zero event

The position-compare match (QFLG[PCM]) is set when the position-counter value (QPOSCNT) matches with the active position-compare register (QPOSCMP) and the position-compare sync output of the programmable pulse width is generated on compare match to trigger an external device.

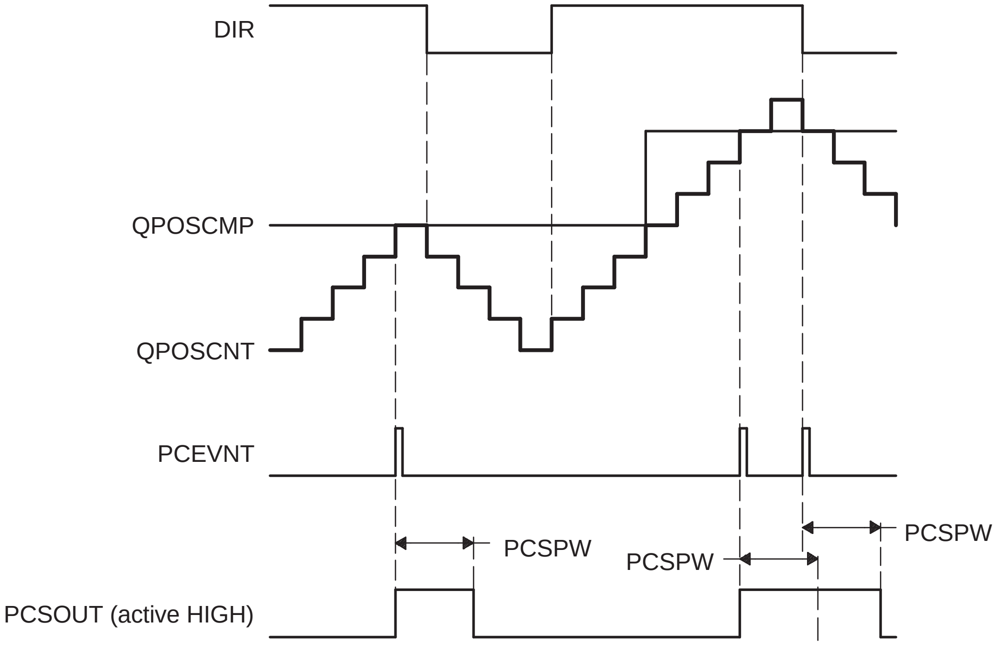
For example, if QPOSCMP = 2, the position-compare unit generates a position-compare event on 1 to 2 transitions of the eQEP position counter for forward counting direction and on 3 to 2 transitions of the eQEP position counter for reverse counting direction (see Figure 21-13).

Figure 21-13. eQEP Position-compare Event Generation Points



The pulse stretcher logic in the position-compare unit generates a programmable position-compare sync pulse output on the position-compare match. In the event of a new position-compare match while a previous position-compare pulse is still active, then the pulse stretcher generates a pulse of specified duration from the new position-compare event as shown in Figure 21-14.

Figure 21-14. eQEP Position-compare Sync Output Pulse Stretcher



21.5 eQEP Edge Capture Unit

The eQEP peripheral includes an integrated edge capture unit to measure the elapsed time between the unit position events as shown in Figure 21-15. This feature is typically used for low speed measurement using the following equation:

$$v(k) = \frac{X}{t(k) - t(k-1)} = \frac{X}{\Delta T} \tag{29}$$

where,

- X - Unit position is defined by integer multiple of quadrature edges (see Figure 21-16)
- ΔT - Elapsed time between unit position events
- v(k) - Velocity at time instant "k"

The eQEP capture timer (QCTMR) runs from prescaled VCLK4 and the prescaler is programmed by the QCAPCTL[CCPS] bits. The capture timer (QCTMR) value is latched into the capture period register (QCPRD) on every unit position event and then the capture timer is reset, a flag is set in QEPSTS:UPEVNT to indicate that new value is latched into the QCPRD register. Software can check this status flag before reading the period register for low speed measurement and clear the flag by writing 1.

Time measurement (ΔT) between unit position events will be correct if the following conditions are met:

- No more than 65,535 counts have occurred between unit position events.
- No direction change between unit position events.

The capture unit sets the eQEP overflow error flag (QEPSTS[COEF]) in the event of capture timer overflow between unit position events. If a direction change occurs between the unit position events, then an error flag is set in the status register (QEPSTS[CDEF]).

Capture Timer (QCTMR) and Capture period register (QCPRD) can be configured to latch on following events.

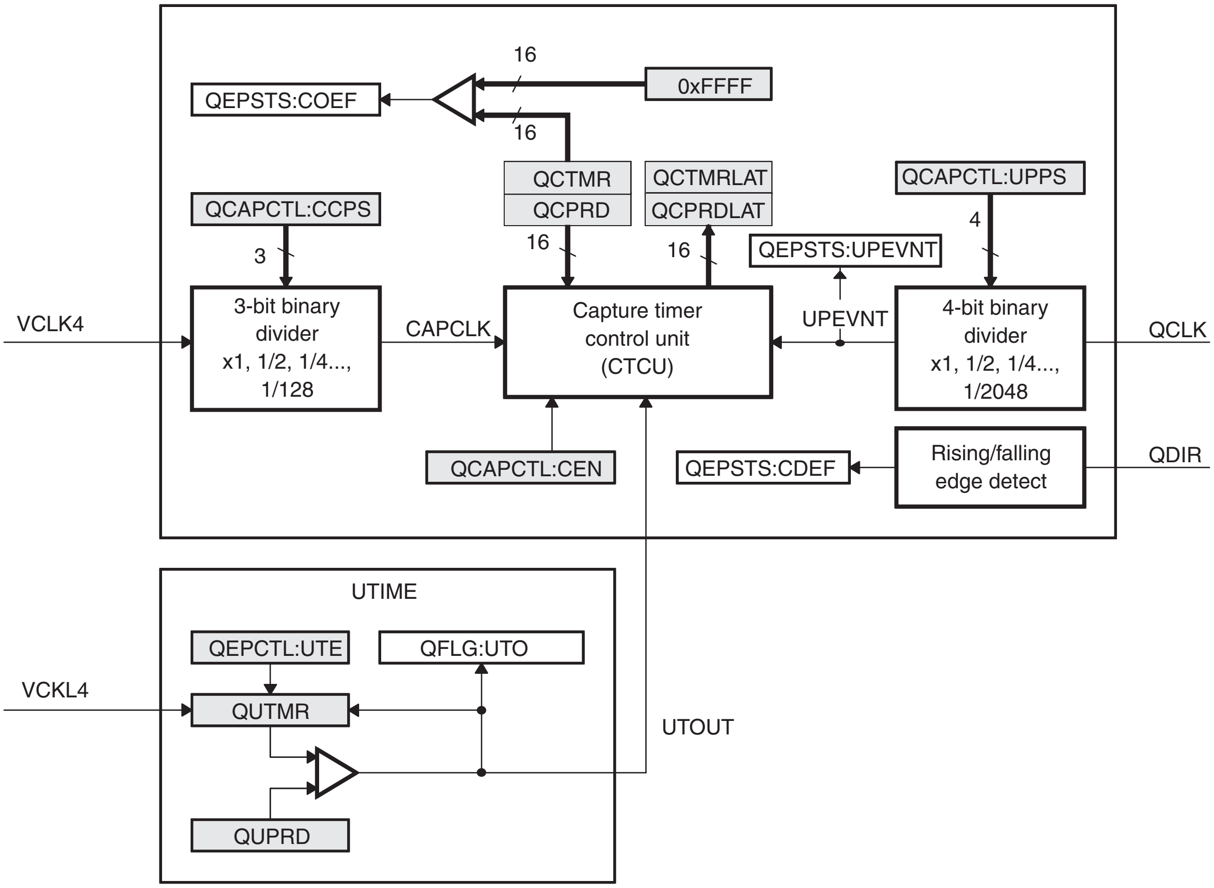
- CPU read of QPOSCNT register
- Unit time-out event

If the QEPCTL[QCLM] bit is cleared, then the capture timer and capture period values are latched into the QCTMRLAT and QCPRDLAT registers, respectively, when the CPU reads the position counter (QPOSCNT).

If the QEPCTL[QCLM] bit is set, then the position counter, capture timer, and capture period values are latched into the QPOSLAT, QCTMRLAT and QCPRDLAT registers, respectively, on unit time out.

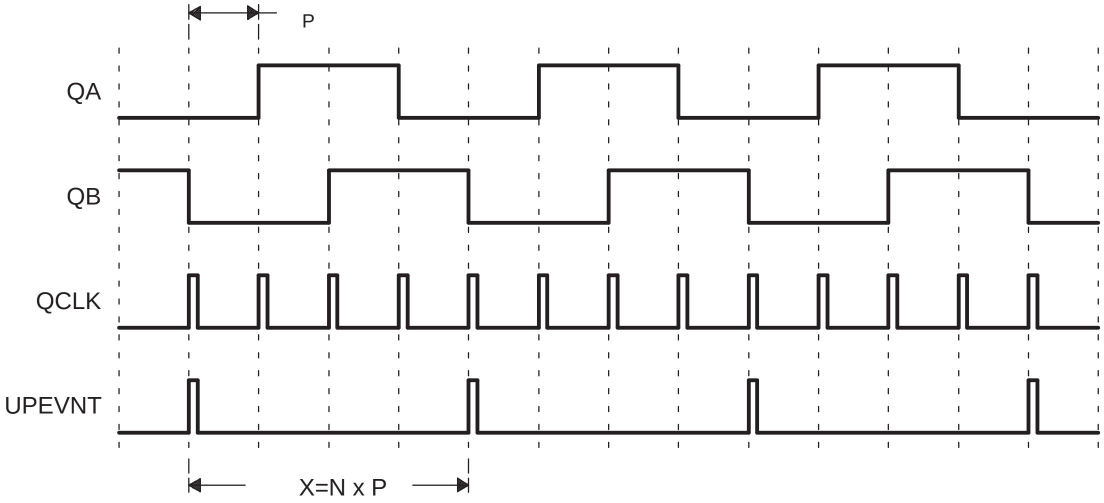
Figure 21-17 shows the capture unit operation along with the position counter.

Figure 21-15. eQEP Edge Capture Unit



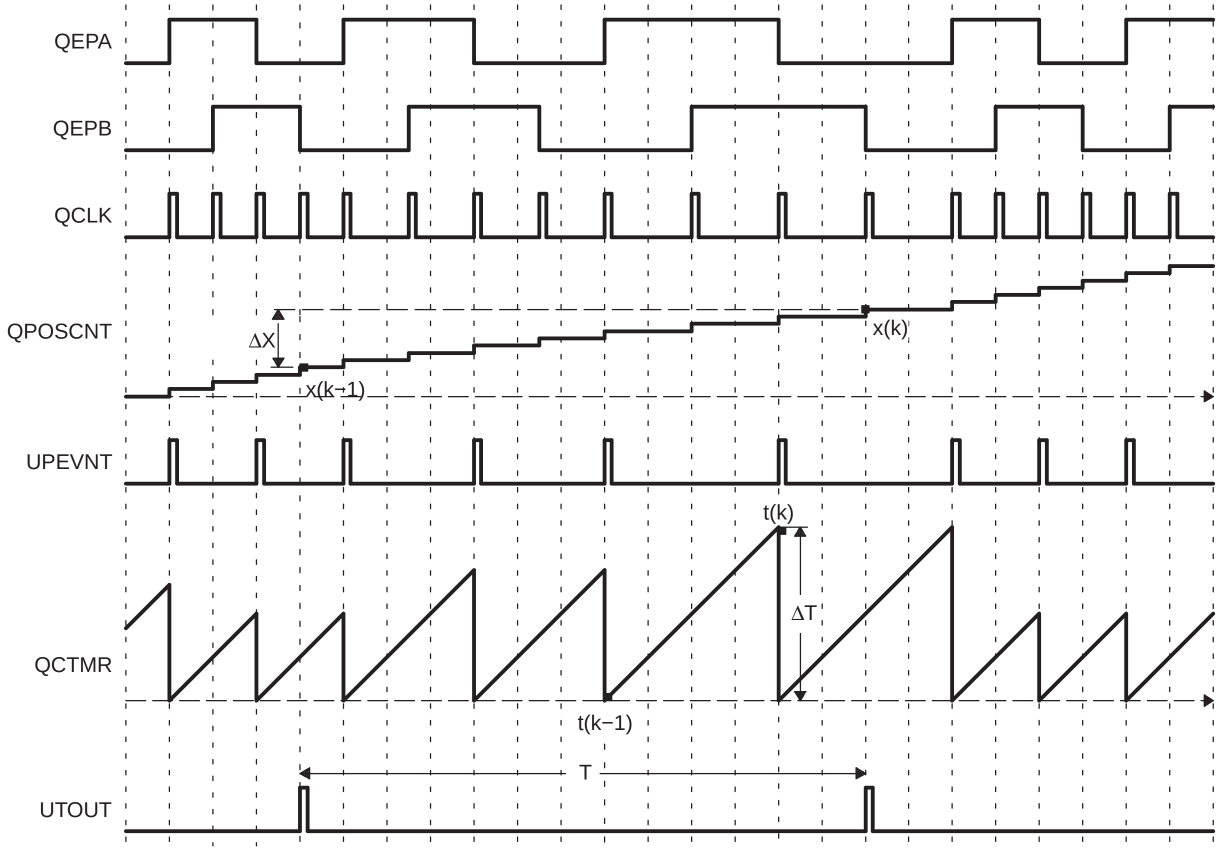
NOTE: The QCAPCTL[UPPS] prescaler should not be modified dynamically (such as switching the unit event prescaler from QCLK/4 to QCLK/8). Doing so may result in undefined behavior. The QCAPCTL[CPPS] prescaler can be modified dynamically (such as switching CAPCLK prescaling mode from SYSCLK/4 to SYSCLK/8) only after the capture unit is disabled.

Figure 21-16. Unit Position Event for Low Speed Measurement (QCAPCTL[UPPS] = 0010)



N = Number of quadrature periods selected using QCAPCTL[UPPS] bits

Figure 21-17. eQEP Edge Capture Unit - Timing Details



Velocity Calculation Equations:

$$v(k) = \frac{x(k) - x(k-1)}{T} = \frac{\Delta X}{T} \text{ o} \tag{30}$$

where

v(k): Velocity at time instant k

x(k): Position at time instant k

x(k-1): Position at time instant k-1

T: Fixed unit time or inverse of velocity calculation rate

ΔX: Incremental position movement in unit time

X: Fixed unit position

ΔT: Incremental time elapsed for unit position movement

t(k): Time instant "k"

t(k-1): Time instant "k-1"

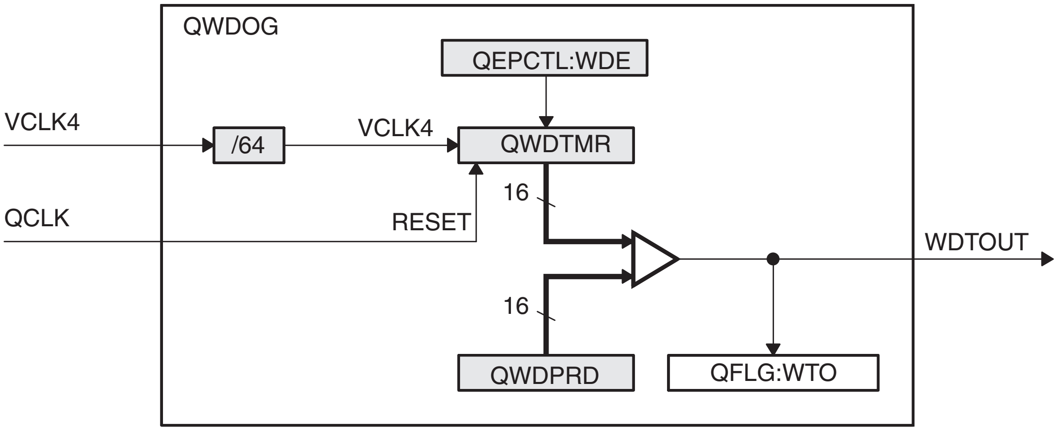
Unit time (T) and unit period(X) are configured using the QUPRD and QCAPCTL[UPPS] registers. Incremental position output and incremental time output is available in the QPOSLAT and QCPRDLAT registers.

Parameter	Relevant Register to Configure or Read the Information
T	Unit Period Register (QUPRD)
ΔX	Incremental Position = QPOSLAT(k) - QPOSLAT(K-1)
X	Fixed unit position defined by sensor resolution and ZCAPCTL[UPPS] bits
ΔT	Capture Period Latch (QCPRDLAT)

21.6 eQEP Watchdog

The eQEP peripheral contains a 16-bit watchdog timer that monitors the quadrature-clock to indicate proper operation of the motion-control system. The eQEP watchdog timer is clocked from VCLK4/64 and the quadrate clock event (pulse) resets the watchdog timer. If no quadrature-clock event is detected until a period match (QWDPRD = QWDTMR), then the watchdog timer will time out and the watchdog interrupt flag will be set (QFLG[WTO]). The time-out value is programmable through the watchdog period register (QWDPRD).

Figure 21-18. eQEP Watchdog Timer

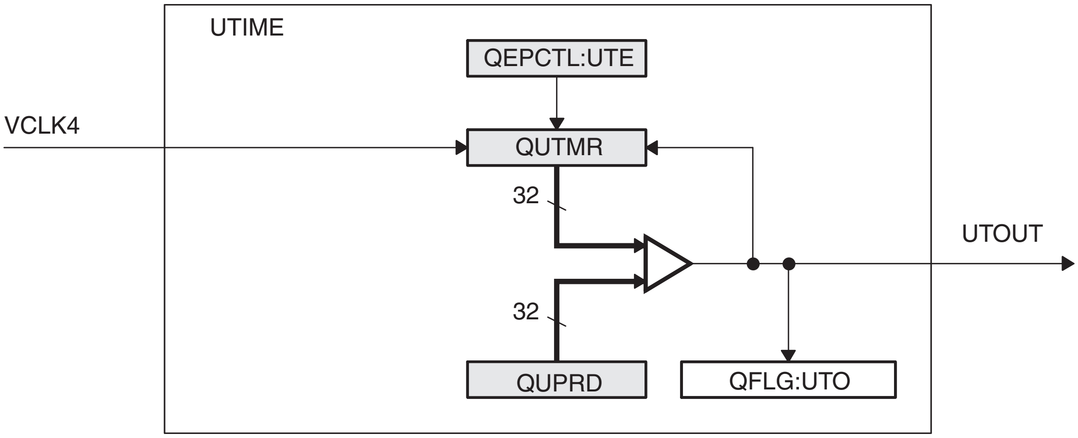


21.7 Unit Timer Base

The eQEP peripheral includes a 32-bit timer (QUTMR) that is clocked by VCLK4 to generate periodic interrupts for velocity calculations. The unit time out interrupt is set (QFLG[UTO]) when the unit timer (QUTMR) matches the unit period register (QUPRD).

The eQEP peripheral can be configured to latch the position counter, capture timer, and capture period values on a unit time out event so that latched values are used for velocity calculation as described in Section 21.5.

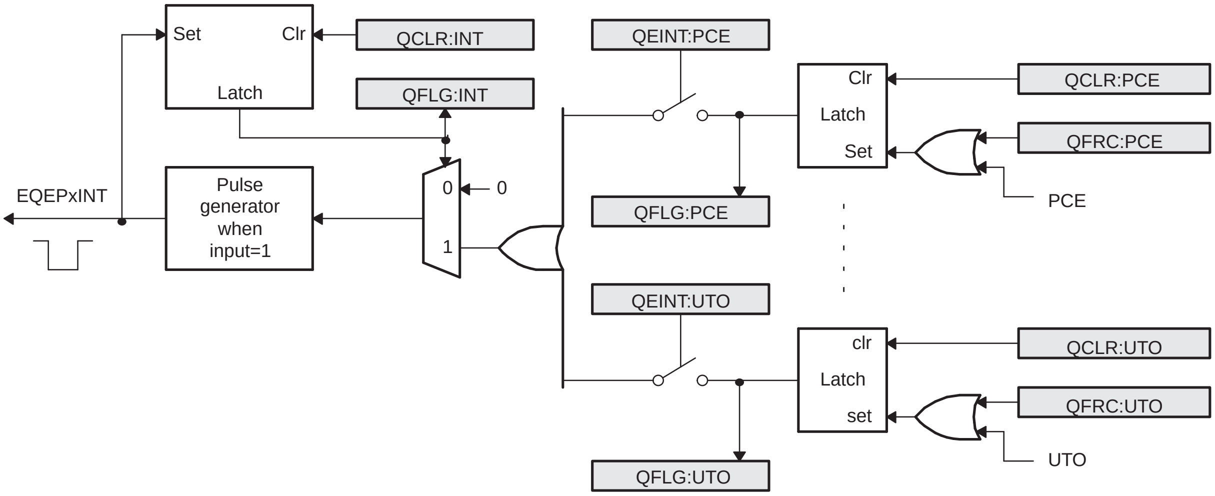
Figure 21-19. eQEP Unit Time Base



21.8 eQEP Interrupt Structure

Figure 21-20 shows how the interrupt mechanism works in the EQEP module.

Figure 21-20. EQEP Interrupt Generation



Eleven interrupt events (PCE, PHE, QDC, WTO, PCU, PCO, PCR, PCM, SEL, IEL and UTO) can be generated. The interrupt control register (QEINT) is used to enable/disable individual interrupt event sources. The interrupt flag register (QFLG) indicates if any interrupt event has been latched and contains the global interrupt flag bit (INT). An interrupt pulse is generated only to the PIE if any of the interrupt events is enabled, the flag bit is 1 and the INT flag bit is 0. The interrupt service routine will need to clear the global interrupt flag bit and the serviced event, via the interrupt clear register (QCLR), before any other interrupt pulses are generated. You can force an interrupt event by way of the interrupt force register (QFRC), which is useful for test purposes.

21.9 EQEP Registers

Table 21-3 lists the memory-mapped registers for the EQEP. All register offset addresses not listed in Table 21-3 should be considered as reserved locations and the register contents should not be modified.

The base address for the control registers is FCF7 9900h for eQEP1 and FCF7 9A00h for eQEP2.

Table 21-3. EQEP Registers

Offset	Acronym	Register Name	Section
0h	QPOSCNT	eQEP Position Counter Register	Section 21.9.1
4h	QPOSINIT	eQEP Position Counter Initialization Register	Section 21.9.2
8h	QPOSMAX	eQEP Maximum Position Count Register	Section 21.9.3
Ch	QPOSCMP	eQEP Position-compare Register	Section 21.9.4
10h	QPOSILAT	eQEP Index Position Latch Register	Section 21.9.5
14h	QPOSSLAT	eQEP Strobe Position Latch Register	Section 21.9.6
18h	QPOSLAT	eQEP Position Counter Latch Register	Section 21.9.7
1Ch	QUTMR	eQEP Unit Timer Register	Section 21.9.8
20h	QUPRD	eQEP Register Unit Period Register	Section 21.9.9
24h	QWDTMR	eQEP Watchdog Timer Register	Section 21.9.10
26h	QWDPRD	eQEP Watchdog Period Register	Section 21.9.11
28h	QDECCTL	eQEP Decoder Control Register	Section 21.9.12
2Ah	QEPCTL	eQEP Control Register	Section 21.9.13
2Ch	QCAPCTL	eQEP Capture Control Register	Section 21.9.14
2Eh	QPOSCTL	eQEP Position-compare Control Register	Section 21.9.15
30h	QEINT	eQEP Interrupt Enable Register	Section 21.9.16
32h	QFLG	eQEP Interrupt Flag Register	Section 21.9.17
34h	QCLR	eQEP Interrupt Clear Register	Section 21.9.18
36h	QFRC	eQEP Interrupt Force Register	Section 21.9.19
38h	QEPSTS	eQEP Status Register	Section 21.9.20
3Ah	QCTMR	eQEP Capture Timer Register	Section 21.9.21
3Ch	QCPRD	eQEP Capture Period Register	Section 21.9.22
3Eh	QCTMRLAT	eQEP Capture Timer Latch Register	Section 21.9.23
40h	QCPRDLAT	eQEP Capture Period Latch Register	Section 21.9.24

Complex bit access types are encoded to fit into small table cells. Table 21-4 shows the codes that are used for access types in this section.

Table 21-4. EQEP Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

21.9.1 QPOSCNT Register (Offset = 0h) [reset = 0h]

QPOSCNT is shown in Figure 21-21 and described in Table 21-5.

Return to Summary Table.

Figure 21-21. QPOSCNT Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="32">QPOSCNT</td></tr>
<tr><td colspan="32">R/W-0h</td></tr>
</table>

Table 21-5. QPOSCNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	QPOSCNT	R/W	0h	This 32-bit position counter register counts up/down on every eQEP pulse based on direction input. This counter acts as a position integrator whose count value is proportional to position from a give reference point.

21.9.2 QPOSINIT Register (Offset = 4h) [reset = 0h]

QPOSINIT is shown in Figure 21-22 and described in Table 21-6.

Return to Summary Table.

Figure 21-22. QPOSINIT Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="32">QPOSINIT</td></tr>
<tr><td colspan="32">R/W-0h</td></tr>
</table>

Table 21-6. QPOSINIT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	QPOSINIT	R/W	0h	This register contains the position value that is used to initialize the position counter based on external strobe or index event. The position counter is initialized through software.

21.9.3 QPOSMAX Register (Offset = 8h) [reset = 0h]

QPOSMAX is shown in Figure 21-23 and described in Table 21-7.

Return to Summary Table.

Figure 21-23. QPOSMAX Register

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

Table 21-7. QPOSMAX Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	QPOSMAX	R/W	0h	This register contains the maximum position counter value.

21.9.4 QPOSCMP Register (Offset = Ch) [reset = 0h]

QPOSCMP is shown in Figure 21-24 and described in Table 21-8.

Return to Summary Table.

Figure 21-24. QPOSCMP Register

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

Table 21-8. QPOSCMP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	QPOSCMP	R/W	0h	The position-compare value in this register is compared with the position counter (QPOSCNT) to generate sync output and/or interrupt on compare match.

21.9.5 QPOSILAT Register (Offset = 10h) [reset = 0h]

QPOSILAT is shown in Figure 21-25 and described in Table 21-9.

Return to Summary Table.

Figure 21-25. QPOSILAT Register

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

Table 21-9. QPOSILAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	QPOSILAT	R	0h	The position-counter value is latched into this register on an index event as defined by the QEPCTL[IEL] bits.

21.9.6 QPOSSLAT Register (Offset = 14h) [reset = 0h]

QPOSSLAT is shown in Figure 21-26 and described in Table 21-10.

Return to Summary Table.

Figure 21-26. QPOSSLAT Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="32">QPOSSLAT</td></tr>
<tr><td colspan="32">R-0h</td></tr>
</table>

Table 21-10. QPOSSLAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	QPOSSLAT	R	0h	The position-counter value is latched into this register on strobe event as defined by the QEPCTL[SEL] bits.

21.9.7 QPOSLAT Register (Offset = 18h) [reset = 0h]

QPOSLAT is shown in Figure 21-27 and described in Table 21-11.

Return to Summary Table.

Figure 21-27. QPOSLAT Register

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

Table 21-11. QPOSLAT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	QPOSLAT	R	0h	The position-counter value is latched into this register on unit time out event.

21.9.8 QUTMR Register (Offset = 1Ch) [reset = 0h]

QUTMR is shown in Figure 21-28 and described in Table 21-12.

Return to Summary Table.

Figure 21-28. QUTMR Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="32">QUTMR</td></tr>
<tr><td colspan="32">R/W-0h</td></tr>
</table>

Table 21-12. QUTMR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	QUTMR	R/W	0h	This register acts as time base for unit time event generation. When this timer value matches with unit time period value, unit time event is generated.

21.9.9 QUPRD Register (Offset = 20h) [reset = 0h]

QUPRD is shown in Figure 21-29 and described in Table 21-13.

Return to Summary Table.

Figure 21-29. QUPRD Register

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

Table 21-13. QUPRD Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	QUPRD	R/W	0h	This register contains the period count for unit timer to generate periodic unit time events to latch the eQEP position information at periodic interval and optionally to generate interrupt.

21.9.10 QWDTMR Register (Offset = 24h) [reset = 0h]

QWDTMR is shown in Figure 21-30 and described in Table 21-14.

Return to Summary Table.

Figure 21-30. QWDTMR Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QWDTMR															
R/W-0h															

Table 21-14. QWDTMR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	QWDTMR	R/W	0h	This register acts as time base for watchdog to detect motor stalls. When this timer value matches with watchdog period value, watchdog timeout interrupt is generated. This register is reset upon edge transition in quadrature-clock indicating the motion.

21.9.11 QWDPRD Register (Offset = 26h) [reset = 0h]

QWDPRD is shown in Figure 21-31 and described in Table 21-15.

Return to Summary Table.

Figure 21-31. QWDPRD Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QWDPRD															
R/W-0h															

Table 21-15. QWDPRD Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	QWDPRD	R/W	0h	This register contains the time-out count for the eQEP peripheral watchdog timer. When the watchdog timer value matches the watchdog period value, a watchdog timeout interrupt is generated.

21.9.12 QDECCTL Register (Offset = 28h) [reset = 0h]

QDECCTL is shown in Figure 21-32 and described in Table 21-16.

Return to Summary Table.

Figure 21-32. QDECCTL Register

15	14	13	12	11	10	9	8
QSRC		SOEN	SPSEL	XCR	SWAP	IGATE	QAP
R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
QBP	QIP	QSP	RESERVED				
R/W-0h	R/W-0h	R/W-0h	R-0h				

Table 21-16. QDECCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-14	QSRC	R/W	0h	Position-counter source selection. 0h = Quadrature count mode (QCLK = iCLK, QDIR = iDIR) 1h = Direction-count mode (QCLK = xCLK, QDIR = xDIR) 2h = UP count mode for frequency measurement (QCLK = xCLK, QDIR = 1) 3h = DOWN count mode for frequency measurement (QCLK = xCLK, QDIR = 0)
13	SOEN	R/W	0h	Sync output-enable. 0h = Disable position-compare sync output. 1h = Enable position-compare sync output.
12	SPSEL	R/W	0h	Sync output pin selection. 0h = Index pin is used for sync output. 1h = Strobe pin is used for sync output.
11	XCR	R/W	0h	External clock rate. 0h = 2x resolution: Count the rising/falling edge. 1h = 1x resolution: Count the rising edge only.
10	SWAP	R/W	0h	Swap quadrature clock inputs. This swaps the input to the quadrature decoder, reversing the counting direction. 0h = Quadrature-clock inputs are not swapped. 1h = Quadrature-clock inputs are swapped.
9	IGATE	R/W	0h	Index pulse gating option. 0h = Disable gating of Index pulse. 1h = Gate the index pin with strobe.
8	QAP	R/W	0h	QEPA input polarity. 0h = No effect. 1h = Negates QEPA input.
7	QBP	R/W	0h	QEPB input polarity. 0h = No effect. 1h = Negates QEPB input.
6	QIP	R/W	0h	QEPI input polarity. 0h = No effect. 1h = Negates QEPI input.
5	QSP	R/W	0h	QEPS input polarity. 0h = No effect. 1h = Negates QEPS input.
4-0	RESERVED	R	0h	Always write as 0.

21.9.13 QEPCTL Register (Offset = 2Ah) [reset = 0h]

QEPCTL is shown in Figure 21-33 and described in Table 21-17.

Return to Summary Table.

Figure 21-33. QEPCTL Register

15	14	13	12	11	10	9	8
FREE_SOFT		PCRM		SEI		IEI	
R/W-0h		R/W-0h		R/W-0h		R/W-0h	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SWI	SEL	IEL		QPEN	QCLM	UTE	WDE
R/W-0h	R/W-0h	R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h

Table 21-17. QEPCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-14	FREE_SOFT	R/W	0h	Emulation control bits. QPOSCNT behavior: 0x0 = Position counter stops immediately on emulation suspend. 0x1 = Position counter continues to count until the rollover. 0x2 = Position counter is unaffected by emulation suspend. 0x3 = Position counter is unaffected by emulation suspend. QWDTMR behavior: 0x0 = Watchdog counter stops immediately. 0x1 = Watchdog counter counts until WD period match roll over. 0x2 = Watchdog counter is unaffected by emulation suspend. 0x3 = Watchdog counter is unaffected by emulation suspend. QUTMR behavior: 0x0 = Unit timer stops immediately. 0x1 = Unit timer counts until period rollover. 0x2 = Unit timer is unaffected by emulation suspend. 0x3 = Unit timer is unaffected by emulation suspend. QCTMR behavior: 0x0 = Capture Timer stops immediately. 0x1 = Capture Timer counts until next unit period event. 0x2 = Capture Timer is unaffected by emulation suspend. 0x3 = Capture Timer is unaffected by emulation suspend.
13-12	PCRM	R/W	0h	Position counter reset mode. 0h = Position counter reset on an index event. 1h = Position counter reset on the maximum position. 2h = Position counter reset on the first index event. 3h = Position counter reset on a unit time event.
11-10	SEI	R/W	0h	Strobe event initialization of position counter. 0h = Does nothing (action is disabled). 1h = Does nothing (action is disabled). 2h = Initializes the position counter on rising edge of the QEPS signal . 3h = Clockwise Direction: Initializes the position counter on the rising edge of QEPS strobe. Counter Clockwise Direction: Initializes the position counter on the falling edge of QEPS strobe.
9-8	IEI	R/W	0h	Index event initialization of position counter. 0h = Do nothing (action is disabled). 1h = Do nothing (action is disabled). 2h = Initializes the position counter on the rising edge of the QEPI signal (QPOSCNT = QPOSINIT). 3h = Initializes the position counter on the falling edge of QEPI signal (QPOSCNT = QPOSINIT).

Table 21-17. QEPCTL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7	SWI	R/W	0h	Software initialization of position counter. 0h = Do nothing (action is disabled). 1h = Initialize position counter (QPOSCNT = QPOSINIT). This bit is not cleared automatically.
6	SEL	R/W	0h	Strobe event latch of position counter. 0h = The position counter is latched on the rising edge of QEPS strobe (QPOSSLAT = POSCCNT). Latching on the falling edge is done by inverting the strobe input using the QSP bit in the QDECCTL register. 1h = Clockwise Direction: Position counter is latched on rising edge of QEPS strobe. Counter Clockwise Direction: Position counter is latched on falling edge of QEPS strobe.
5-4	IEL	R/W	0h	Index event latch of position counter (software index marker). 0h = Reserved 1h = Latches position counter on rising edge of the index signal. 2h = Latches position counter on falling edge of the index signal. 3h = Software index marker. Latches the position counter and quadrature direction flag on index event marker. The position counter is latched to the QPOSILAT register and the direction flag is latched in the QEPSTS[QDLF] bit. This mode is useful for software index marking.
3	QPEN	R/W	0h	Quadrature position counter enable/software reset. 0h = Reset the eQEP peripheral internal operating flags/read-only registers. Control/configuration registers are not disturbed by a software reset. 1h = eQEP position counter is enabled.
2	QCLM	R/W	0h	eQEP capture latch mode. 0h = Latch on position counter read by CPU. Capture timer and capture period values are latched into QCTMRLAT and QCPRDLAT registers when CPU reads the QPOSCNT register. 1h = Latch on unit time out. Position counter, capture timer and capture period values are latched into QPOSLAT, QCTMRLAT, and QCPRDLAT registers on unit time out.
1	UTE	R/W	0h	eQEP unit timer enable. 0h = Disable eQEP unit timer. 1h = Enable eQEP unit timer.
0	WDE	R/W	0h	eQEP watchdog enable. 0h = Disable eQEP watchdog timer. 1h = Enable eQEP watchdog timer.

21.9.14 QCAPCTL Register (Offset = 2Ch) [reset = 0h]

QCAPCTL is shown in Figure 21-34 and described in Table 21-18.

Return to Summary Table.

Figure 21-34. QCAPCTL Register

15	14	13	12	11	10	9	8
CEN	RESERVED						
R/W-0h	R-0h						
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RESERVED	CCPS			UPPS			
R-0h	R/W-0h			R/W-0h			

Table 21-18. QCAPCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	CEN	R/W	0h	eQEP capture enable. 0h = eQEP capture unit is disabled. 1h = eQEP capture unit is enabled.
14-7	RESERVED	R	0h	Always write as 0.
6-4	CCPS	R/W	0h	eQEP capture timer clock prescaler. 0h = CAPCLK = VCLK4/1 1h = CAPCLK = VCLK4/2 2h = CAPCLK = VCLK4/4 3h = CAPCLK = VCLK4/8 4h = CAPCLK = VCLK4/16 5h = CAPCLK = VCLK4/32 6h = CAPCLK = VCLK4/64 7h = CAPCLK = VCLK4/128
3-0	UPPS	R/W	0h	Unit position event prescaler. 0h = UPEVNT = QCLK/1 1h = UPEVNT = QCLK/2 2h = UPEVNT = QCLK/4 3h = UPEVNT = QCLK/8 4h = UPEVNT = QCLK/16 5h = UPEVNT = QCLK/32 6h = UPEVNT = QCLK/64 7h = UPEVNT = QCLK/128 8h = UPEVNT = QCLK/256 9h = UPEVNT = QCLK/512 Ah = UPEVNT = QCLK/1024 Bh = UPEVNT = QCLK/2048 Ch = Reserved Dh = Reserved Eh = Reserved Fh = Reserved

21.9.15 QPOSCTL Register (Offset = 2Eh) [reset = 0h]

QPOSCTL is shown in Figure 21-35 and described in Table 21-19.

Return to Summary Table.

Figure 21-35. QPOSCTL Register

15	14	13	12	11	10	9	8
PCSHDW	PCLOAD	PCPOL	PCE	PCSPW			
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PCSPW							
R/W-0h							

Table 21-19. QPOSCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
15	PCSHDW	R/W	0h	Position-compare shadow enable. 0h = Shadow is disabled, load Immediate. 1h = Shadow is enabled.
14	PCLOAD	R/W	0h	Position-compare shadow load mode. 0h = Load on QPOSCNT = 0. 1h = Load when QPOSCNT = QPOSCMP.
13	PCPOL	R/W	0h	Polarity of sync output. 0h = Active HIGH pulse output. 1h = Active LOW pulse output.
12	PCE	R/W	0h	Position-compare enable. 0h = Disable position compare unit. 1h = Enable position compare unit.
11-0	PCSPW	R/W	0h	Select-position-compare sync output pulse width. Valid values: 0 to FFFh. 0h = 1 x 4 x VCLK4 cycles 1h = 2 x 4 x VCLK4 cycles FFFh = 4096 x 4 x VCLK4 cycles

21.9.16 QEINT Register (Offset = 30h) [reset = 0h]

QEINT is shown in Figure 21-36 and described in Table 21-20.

Return to Summary Table.

Figure 21-36. QEINT Register

15	14	13	12	11	10	9	8
RESERVED				UTO	IEL	SEL	PCM
R-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PCR	PCO	PCU	WTO	QDC	QPE	PCE	RESERVED
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R-0h

Table 21-20. QEINT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Always write as 0
11	UTO	R/W	0h	Unit time out interrupt enable. 0h = Interrupt is disabled. 1h = Interrupt is enabled.
10	IEL	R/W	0h	Index event latch interrupt enable. 0h = Interrupt is disabled. 1h = Interrupt is enabled.
9	SEL	R/W	0h	Strobe event latch interrupt enable. 0h = Interrupt is disabled. 1h = Interrupt is enabled.
8	PCM	R/W	0h	Position-compare match interrupt enable. 0h = Interrupt is disabled. 1h = Interrupt is enabled.
7	PCR	R/W	0h	Position-compare ready interrupt enable. 0h = Interrupt is disabled. 1h = Interrupt is enabled.
6	PCO	R/W	0h	Position counter overflow interrupt enable. 0h = Interrupt is disabled. 1h = Interrupt is enabled.
5	PCU	R/W	0h	Position counter underflow interrupt enable. 0h = Interrupt is disabled. 1h = Interrupt is enabled.
4	WTO	R/W	0h	Watchdog time out interrupt enable. 0h = Interrupt is disabled. 1h = Interrupt is enabled.
3	QDC	R/W	0h	Quadrature direction change interrupt enable. 0h = Interrupt is disabled. 1h = Interrupt is enabled.
2	QPE	R/W	0h	Quadrature phase error interrupt enable. 0h = Interrupt is disabled. 1h = Interrupt is enabled.
1	PCE	R/W	0h	Position counter error interrupt enable. 0h = Interrupt is disabled. 1h = Interrupt is enabled.
0	RESERVED	R	0h	Reserved

21.9.17 QFLG Register (Offset = 32h) [reset = 0h]

QFLG is shown in Figure 21-37 and described in Table 21-21.

Return to Summary Table.

Figure 21-37. QFLG Register

15	14	13	12	11	10	9	8
RESERVED				UTO	IEL	SEL	PCM
R-0h				R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
PCR	PCO	PCU	WTO	QDC	PHE	PCE	INT
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

Table 21-21. QFLG Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Always write as 0.
11	UTO	R	0h	Unit time out interrupt flag. 0h = No interrupt is generated. 1h = Set by eQEP unit timer period match.
10	IEL	R	0h	Index event latch interrupt flag. 0h = No interrupt is generated. 1h = Set after latching the QPOSCNT to QPOSILAT.
9	SEL	R	0h	Strobe event latch interrupt flag. 0h = No interrupt is generated. 1h = Set after latching the QPOSCNT to QPOSSLAT.
8	PCM	R	0h	eQEP compare match event interrupt flag. 0h = No interrupt is generated. 1h = Set on position-compare match.
7	PCR	R	0h	Position-compare ready interrupt flag. 0h = No interrupt is generated. 1h = Set after transferring the shadow register value to the active position compare register.
6	PCO	R	0h	Position counter overflow interrupt flag. 0h = No interrupt is generated. 1h = Set on position counter overflow.
5	PCU	R	0h	Position counter underflow interrupt flag. 0h = No interrupt is generated. 1h = Set on position counter underflow.
4	WTO	R	0h	Watchdog timeout interrupt flag. 0h = No interrupt is generated. 1h = Set by watchdog timeout.
3	QDC	R	0h	Quadrature direction change interrupt flag. 0h = No interrupt is generated. 1h = Set during change of direction.
2	PHE	R	0h	Quadrature phase error interrupt flag. 0h = No interrupt is generated. 1h = Set on simultaneous transition of QEPA and QEPB.
1	PCE	R	0h	Position counter error interrupt flag. 0h = No interrupt is generated. 1h = Position counter error.

Table 21-21. QFLG Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	INT	R	0h	Global interrupt status flag. 0h = No interrupt is generated. 1h = Interrupt is generated.

21.9.18 QCLR Register (Offset = 34h) [reset = 0h]

QCLR is shown in Figure 21-38 and described in Table 21-22.

Return to Summary Table.

Figure 21-38. QCLR Register

15	14	13	12	11	10	9	8
RESERVED				UTO	IEL	SEL	PCM
R-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

7	6	5	4	3	2	1	0
PCR	PCO	PCU	WTO	QDC	PHE	PCE	INT
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

Table 21-22. QCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Always write as 0.
11	UTO	R/W	0h	Clear unit time out interrupt flag. 0h = No effect. 1h = Clears the interrupt flag.
10	IEL	R/W	0h	Clear index event latch interrupt flag. 0h = No effect. 1h = Clears the interrupt flag.
9	SEL	R/W	0h	Clear strobe event latch interrupt flag. 0h = No effect. 1h = Clears the interrupt flag.
8	PCM	R/W	0h	Clear eQEP compare match event interrupt flag. 0h = No effect. 1h = Clears the interrupt flag.
7	PCR	R/W	0h	Clear position-compare ready interrupt flag. 0h = No effect. 1h = Clears the interrupt flag.
6	PCO	R/W	0h	Clear position counter overflow interrupt flag. 0h = No effect. 1h = Clears the interrupt flag.
5	PCU	R/W	0h	Clear position counter underflow interrupt flag. 0h = No effect. 1h = Clears the interrupt flag.
4	WTO	R/W	0h	Clear watchdog timeout interrupt flag. 0h = No effect. 1h = Clears the interrupt flag.
3	QDC	R/W	0h	Clear quadrature direction change interrupt flag. 0h = No effect. 1h = Clears the interrupt flag.
2	PHE	R/W	0h	Clear quadrature phase error interrupt flag. 0h = No effect. 1h = Clears the interrupt flag.
1	PCE	R/W	0h	Clear position counter error interrupt flag. 0h = No effect. 1h = Clears the interrupt flag.

Table 21-22. QCLR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	INT	R/W	0h	Global interrupt clear flag. 0h = No effect. 1h = Clears the interrupt flag and enables further interrupts to be generated if an event flags is set to 1.

21.9.19 QFRC Register (Offset = 36h) [reset = 0h]

QFRC is shown in Figure 21-39 and described in Table 21-23.

Return to Summary Table.

Figure 21-39. QFRC Register

15	14	13	12	11	10	9	8
RESERVED				UTO	IEL	SEL	PCM
R-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PCR	PCO	PCU	WTO	QDC	PHE	PCE	RESERVED
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R-0h

Table 21-23. QFRC Register Field Descriptions

Bit	Field	Type	Reset	Description
15-12	RESERVED	R	0h	Always write as 0s
11	UTO	R/W	0h	Force unit time out interrupt. 0h = No effect. 1h = Force the interrupt.
10	IEL	R/W	0h	Force index event latch interrupt. 0h = No effect. 1h = Force the interrupt.
9	SEL	R/W	0h	Force strobe event latch interrupt. 0h = No effect. 1h = Force the interrupt.
8	PCM	R/W	0h	Force position-compare match interrupt. 0h = No effect. 1h = Force the interrupt.
7	PCR	R/W	0h	Force position-compare ready interrupt. 0h = No effect. 1h = Force the interrupt.
6	PCO	R/W	0h	Force position counter overflow interrupt. 0h = No effect. 1h = Force the interrupt.
5	PCU	R/W	0h	Force position counter underflow interrupt. 0h = No effect. 1h = Force the interrupt.
4	WTO	R/W	0h	Force watchdog time out interrupt. 0h = No effect. 1h = Force the interrupt.
3	QDC	R/W	0h	Force quadrature direction change interrupt. 0h = No effect. 1h = Force the interrupt.
2	PHE	R/W	0h	Force quadrature phase error interrupt. 0h = No effect. 1h = Force the interrupt.
1	PCE	R/W	0h	Force position counter error interrupt. 0h = No effect. 1h = Force the interrupt.
0	RESERVED	R	0h	Always write as 0.

21.9.20 QEPSTS Register (Offset = 38h) [reset = 8Eh]

QEPSTS is shown in Figure 21-40 and described in Table 21-24.

Return to Summary Table.

Figure 21-40. QEPSTS Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
UPEVNT	FIDF	QDF	QDLF	COEF	CDEF	FIMF	PCEF
R-1h	R-0h	R-0h	R-0h	R/W-1h	R/W-1h	R/W-1h	R-0h

Table 21-24. QEPSTS Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	Always write as 0.
7	UPEVNT	R	1h	Unit position event flag. 0h = No unit position event is detected. 1h = Unit position event is detected. Write 1 to clear.
6	FIDF	R	0h	Direction on the first index marker. Status of the direction is latched on the first index event marker. 0h = Counter-clockwise rotation (or reverse movement) on the first index event. 1h = Clockwise rotation (or forward movement) on the first index event.
5	QDF	R	0h	Quadrature direction flag. 0h = Counter-clockwise rotation (or reverse movement). 1h = Clockwise rotation (or forward movement).
4	QDLF	R	0h	eQEP direction latch flag. Status of direction is latched on every index event marker. 0h = Counter-clockwise rotation (or reverse movement) on index event marker. 1h = Clockwise rotation (or forward movement) on index event marker.
3	COEF	R/W	1h	Capture overflow error flag. 0h = Sticky bit, cleared by writing 1. 1h = Overflow occurred in eQEP Capture timer (QCTMR).
2	CDEF	R/W	1h	Capture direction error flag. 0h = Sticky bit, cleared by writing 1. 1h = Direction change occurred between the capture position event.
1	FIMF	R/W	1h	First index marker flag. 0h = Sticky bit, cleared by writing 1. 1h = Set by first occurrence of index pulse.
0	PCEF	R	0h	Position counter error flag. This bit is not sticky and it is updated for every index event. 0h = No error occurred during the last index transition. 1h = Position counter error.

21.9.21 QCTMR Register (Offset = 3Ah) [reset = 0h]

QCTMR is shown in Figure 21-41 and described in Table 21-25.

Return to Summary Table.

Figure 21-41. QCTMR Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QCTMR															
R/W-0h															

Table 21-25. QCTMR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	QCTMR	R/W	0h	This register provides time base for edge capture unit.

21.9.22 QCPRD Register (Offset = 3Ch) [reset = 0h]

QCPRD is shown in Figure 21-42 and described in Table 21-26.

Return to Summary Table.

Figure 21-42. QCPRD Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QCPRD															
R/W-0h															

Table 21-26. QCPRD Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	QCPRD	R/W	0h	This register holds the period count value between the last successive eQEP position events.

21.9.23 QCTMRLAT Register (Offset = 3Eh) [reset = 0h]

QCTMRLAT is shown in Figure 21-43 and described in Table 21-27.

Return to Summary Table.

Figure 21-43. QCTMRLAT Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QCTMRLAT															
R-0h															

Table 21-27. QCTMRLAT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	QCTMRLAT	R	0h	The eQEP capture timer value is latched into this register on two events viz., unit timeout event, reading the eQEP position counter.

21.9.24 QCPRDLAT Register (Offset = 40h) [reset = 0h]

QCPRDLAT is shown in Figure 21-44 and described in Table 21-28.

Return to Summary Table.

Figure 21-44. QCPRDLAT Register

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">QCPRDLAT</td></tr>
<tr><td colspan="16">R/W-0h</td></tr>
</table>

Table 21-28. QCPRDLAT Register Field Descriptions

Bit	Field	Type	Reset	Description
15-0	QCPRDLAT	R/W	0h	eQEP capture period value is latched into this register on two events, unit timeout event, reading the eQEP position counter.

Chapter 22
SPNU514C–March 2018

Analog To Digital Converter (ADC) Module

This chapter describes the analog to digital converter (ADC) interface module.

Topic **Page**

22.1 Overview 935
22.2 Introduction 936
22.3 Basic Features and Usage of the ADC........ 939
22.4 Advanced Conversion Group Configuration Options 946
22.5 ADC Module Basic Interrupts 954
22.6 ADC Module DMA Requests........ 955
22.7 ADC Magnitude Threshold Interrupts 956
22.8 ADC Special Modes 957
22.9 ADC Results' RAM Special Features 964
22.10 ADEVT Pin General Purpose I/O Functionality 965
22.11 ADC_CONTROL Registers........ 967

22.1 Overview

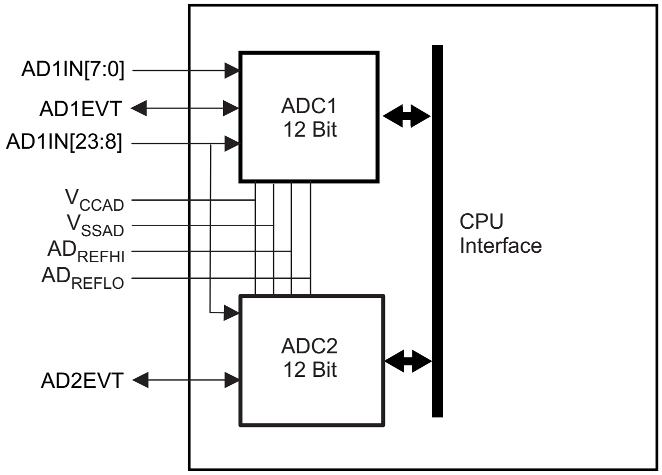
This microcontroller implements up to two instances of the ADC module. The main features of the ADC module are:

- Selectable 10-bit or 12-bit resolution, 10-bit mode is the default
- Successive-approximation-register (SAR) architecture
- Three conversion groups – Group1, Group2 and Event Group
- All three conversion groups can be configured to be hardware-triggered; group1 and group2 can also be triggered by software
- Conversion results are stored in a 64-word memory (SRAM)
  - These 64 words are divided between the three conversion groups and are configurable by software
  - Accesses to the conversion result RAM are protected by parity
- Flexible options for generating DMA requests for transferring conversion results
- Selectable channel conversion order:
  - Sequential conversions in ascending order of channel number, OR
  - User-defined channel conversion order with the Enhanced Channel Selection Mode
- Single or continuous conversion modes
- Embedded self-test logic for input channel failure detection (open / short to power / short to ground)
- Embedded calibration logic for offset error correction
- Enhanced Power-down mode
- External event pin (ADxEVT) to trigger conversions
  - ADxEVT is also programmable as general-purpose I/O
- Eight hardware events to trigger conversions

The two instances of the 12-bit ADC modules on the microcontroller share 16 analog input channels. The connections are shown in Figure 22-1.

- ADC1 supports 24 channels.
- ADC2 supports 16 channels, all of which are shared with ADC1.
- When using both ADC1 and ADC2 on a shared channel, the sample windows must be identical such that the sample windows completely match each other or non-overlapping with a minimum of 2 ADC cycles buffer between the end of one ADC’s sample window and the start of the other ADC’s sample window.
- The reference voltages, as well as operating supply and ground, are shared between the two ADC cores.

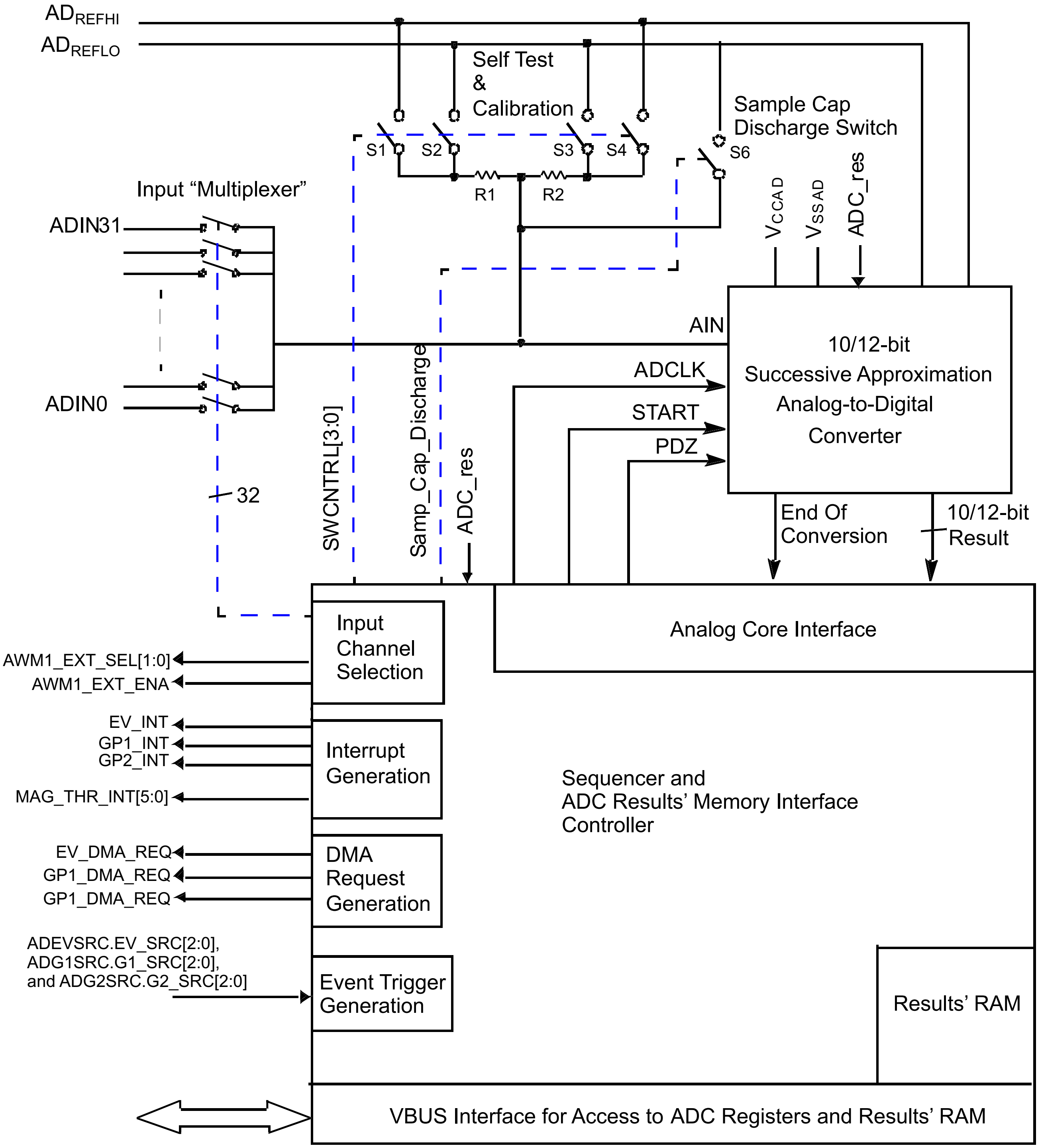
Figure 22-1. Channel Assignments of Two ADC Cores



22.2 Introduction

This section presents a brief functional description of the analog-to-digital converter (ADC) module. Figure 22-2 illustrates the components of the ADC module.

Figure 22-2. ADC Block Diagram



22.2.1 Input Multiplexor

The input multiplexor (MUX) connects the selected input channel to the AIN input of the ADC core. The ADC1 module supports up to 24 inputs as shown in Figure 22-2. The ADC2 module supports up to 16 inputs. The sequencer selects the channel to be converted. Enabling the enhanced channel selection mode also allows one or more of the analog input channels to be connected to the output of an external analog switch or multiplexor.

22.2.2 Self-Test and Calibration Cell

The ADC includes specific hardware that allows a software algorithm to detect open/short on an ADC analog input. It also allows the application program to calibrate the ADC. Also see Section 22.8.1 and Section 22.8.2.

22.2.3 Analog-to-Digital Converter Core

The ADC core is a combination voltage scaling, charge redistribution Successive Approximation Register (SAR) based analog-to-digital converter. The core can be configured for operation in 10-bit resolution (default) or 12-bit resolution. This is controlled by the sequencer logic. This selection applies to all conversions performed by the ADC module. It is not possible to convert some channels with a 12-bit resolution and some with a 10-bit resolution.

A single conversion from an analog input to a digital conversion result occurs in two distinct periods:

- Sampling Period:
  - The sequencer generates a START signal to the ADC core to signal the start of the sampling period.
  - The analog input signal is sampled directly on to the switched capacitor array during this period, providing an inherent sample-and-hold function.
  - The sampling period ends one full ADCLK after the falling edge of the START signal.
  - The sequencer can control the sampling period duration by configuring the conversion group’s sample time control register (ADEVSAMP, ADG1SAMP, ADG2SAMP). This register controls the time for which the START signal stays high.
- Conversion Period:
  - The conversion period starts one full ADCLK after the falling edge of START.
  - One bit of the conversion result is output on each rising edge of ADCLK in the conversion period, starting with the most-significant bit first.
  - The conversion period is 12 ADCLK cycles in case of a 12-bit ADC, and is 10 ADCLK cycles in case of a 10-bit ADC.
  - The ADC core generates an End-Of-Conversion (EOC) signal to the sequencer at the end of the conversion period. At this time the complete 12-, or 10-bit conversion result is available.
  - The sequencer captures the ADC core conversion result output as soon as EOC is driven High.

The analog conversion range is determined by the reference voltages: AD_{REFHI} and AD_{REFLO}. AD_{REFHI} is the top reference voltage and is the maximum analog voltage that can be converted. An analog input voltage equal to AD_{REFHI} or higher results in an output code of 0x3FF for 10-bit resolution and 0xFFF for 12-bit resolution. AD_{REFLO} is the bottom reference voltage and is the minimum analog voltage that can be converted. Applying an input voltage equal to AD_{REFLO} or lower results in an output code of 0x000. Both AD_{REFHI} and AD_{REFLO} must be chosen not to exceed the analog power supplies: V_{CCAD} and V_{SSAD}, respectively. Input voltages between AD_{REFHI} and AD_{REFLO} produce a conversion result given by Equation 31 for 10-bit resolution and by Equation 32 for 12-bit resolution.

$$DigitalResult = \frac{1024 \ x \ (InputVoltage - AD_{REFLO})}{AD_{REFHI} - AD_{REFLO}} - 0.5 \tag{31}$$

$$DigitalResult = \frac{4096 \ x (InputVoltage - AD_{REFLO})}{(AD_{REFHI} - AD_{REFLO})} - 0.5 \tag{32}$$

22.2.4 Sequencer

The sequencer coordinates the operations of the ADC, including the input multiplexor, the ADC core, and the result memory. In addition, the logic of the sequencer sets the status register flags when the conversion is ongoing, stopped, or finished.

All the features of the sequencer are discussed in detail in the following sections of this document.

22.2.5 Conversion Groups

Several applications require groups of channels to be converted using a single trigger source for example. There could also be some groups of channels identified which require a specific setting of the acquisition time. The ADC module supports three conversion groups for this purpose – Group1, Group2 and the Event Group.

Any of the available analog input channels can be assigned to any of the conversion groups. This also allows a particular channel to be repeatedly sampled by selecting it in multiple groups. There is an inherent priority scheme used when multiple conversion groups are triggered at once. The Event Group is the highest-priority, followed by the Group1 and then the Group2.

The Event Group is always hardware event-triggered. Group1 and Group2 are software-triggered by default and can be configured to be hardware-, or event-triggered as well. The triggering of conversions in each group is discussed in Section 22.3.6.

Each conversion group has a separate set of control registers to:

- Select the input channels to be converted
- Configure the mode of conversion: single conversion sequence or continuous conversions
- Configure the input channel sampling time
- Configure the interrupt and/or DMA request generation conditions

22.3 Basic Features and Usage of the ADC

This section describes the usage of the basic features of the ADC module.

22.3.1 How to Select Between 12-bit and 10-bit Resolutions

The 10_12_BIT field of the ADC Operating Mode Control Register (ADOPMODECR) configures the ADC to be in 10-bit or 12-bit resolution mode.

- If 10_12_BIT = 0, the module is in 10-bit resolution mode. This is the default mode of operation.
- If 10_12_BIT = 1, the module is in 12-bit resolution mode.

22.3.2 How to Set Up the ADCLK Speed

The ADC sequencer generates the clock for the ADC core, ADCLK. The ADC core uses the ADCLK signal for its timing. The ADCLK is generated by dividing down the input clock to the ADC module, which is the VBUSP interface clock, VCLK. A 5-bit field (PS) in the ADC Clock Control Register (ADCLOCKCR) is used to divide down the VCLK by 1 up to 32. The ADCLK valid frequency range is specified in the device datasheet.

$f_{ADCLK} = f_{VCLK} / (PS + 1)$

The maximum frequency for ADCLK is specified in the device datasheet.

22.3.3 How to Set Up the Input Channel Acquisition Time

The signal acquisition time for each group is separately configurable using the ADG1SAMP[11:0], ADG2SAMP[11:0], and ADEVSAMP[11:0] registers.

The acquisition time is specified in terms of ADCLK cycles and ranges from a minimum of 2 ADCLK cycles to a maximum of 4098 ADCLK cycles.

For example, Group1 acquisition time, t_{ACQG1} = G1SAMP[11:0] + 2, in ADCLK cycles.

The minimum acquisition time is specified in the device datasheet. This time also depends on the impedance of the circuit connected to the analog input channel being converted. See the *ADC Source Impedance for Hercules™ ARM® Safety MCUs Application Report* (SPNA118).

22.3.4 How to Select an Input Channel for Conversion

The ADC module needs to be enabled first before selecting an input channel for conversion. The ADC module can be enabled by setting the ADC EN bit in the ADC Operating Mode Control Register (ADOPMODECR). Multiple input channels can be selected for conversion in each group. Only one input channel is converted at a time. The channels to be converted are configured in one or more of the three conversion groups' channel selection registers. Channels to be converted in Group1 are configured in the Group1 Channel-Select Register (ADG1SEL), those to be converted in Group2 are configured in the Group2 Channel-Select Register (ADG2SEL), and those to be converted in the Event Group are configured in the Event Group Channel-Select Register (ADEVSEL).

The description in this section only refers to the case when the enhanced channel selection mode is not enabled. Input channel selection in the enhanced channel selection mode is defined in Section 22.4.

22.3.5 How to Select Between Single Conversion Sequence or Continuous Conversions

Each group has its own mode control register. The MODE field of these control registers allow the application to select between a single conversion sequence or continuous conversion mode.

NOTE: Selecting continuous conversion mode for all three groups

All three conversion groups cannot be configured to be in a continuous conversion mode. If the application configures the group mode control registers to enable continuous conversion mode for all three groups, then the Group2 will be automatically be configured to be in a single conversion sequence mode.

With conversions ongoing in continuous conversion mode, if the MODE field of a group is cleared, then that group switches to the single conversion sequence mode. Conversions for this group will stop once all channels selected for that group have been converted.

22.3.6 How to Start a Conversion

The conversion groups Group1 and Group2 are software-triggered by default. A conversion in these groups can be started just by writing the desired channels to the respective Channel-Select Registers. For example, in order to convert channels 0, 1, 2, and 3 in Group1 and channels 8, 9, 10, and 11 in Group2, the application just has to write 0x0000000F to ADG1SEL and 0x00000F00 to ADG2SEL. The ADC module will start by servicing the group that was triggered first, Group1 in this example.

The conversions for all groups are performed in ascending order of the channel number. For the Group1 the conversions will be performed in the order: channel 0 first, followed by channel 1, then channel 2, and then channel 3. The Group2 conversions will be performed in the order: channels 8, 9, 10, and 11.

The Event Group is only hardware-triggered. There are up to eight hardware event trigger sources defined for the ADC module. Check the device datasheet for a complete listing of these eight hardware trigger options.

The trigger source to be used needs to be configured in the ADEVSRC register. Similar registers also exist for the Group1 and Group2 as these can also be configured to be event-triggered.

The polarity of the event trigger is also configurable, with a falling edge being the default.

An Event Group conversion starts when at least one channel is selected for conversion in this group, and when the defined event trigger occurs.

If any conversion group is configured to be in a continuous conversion mode, then it needs to only be triggered once. All the channels selected for conversion in that group will be converted repeatedly.

22.3.7 How to Know When the Group Conversion is Completed

Each conversion group has a status flag to indicate when its conversion has ended. See ADEVSR, ADG1SR, and ADG2SR. This bit is set when a conversion sequence for a group ends. This bit does is always set if a group is configured for continuous conversions.

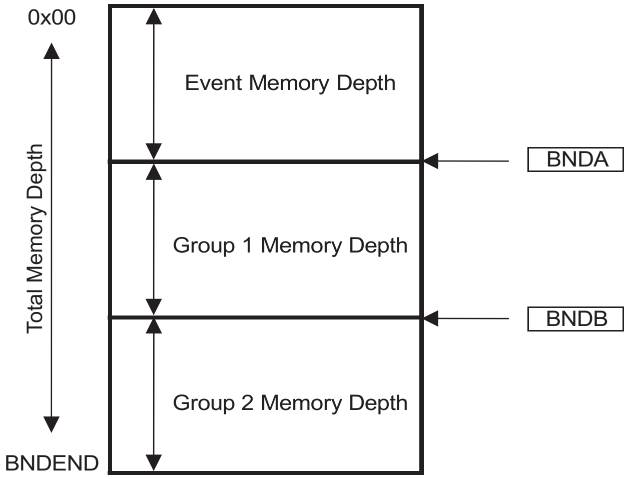
22.3.8 How Results are Stored in the Results' Memory

The ADC stores the conversion results in three separate memory regions in the ADC Results' RAM, one region for each group. Each memory region is a stack of buffers, with each buffer capable of holding one conversion result. The number of buffers allocated for each group is programmed by configuring the ADC module registers ADBNDCR and ADBNDEND.

ADBNDCR contains two 9-bit pointers BNDA and BNDB. BNDA, BNDB, and BNDEND are used to partition the total memory available into three memory regions as shown in Figure 22-3. Both BNDA and BNDB are pointers referenced from the start of the results' memory. BNDA specifies the number of buffers allocated for the Event Group conversion results in units of two buffers; BNDB specifies the number of buffers allocated for the Event Group plus Group1 in units of two buffers. Refer to for more details on configuring the ADC results' memory.

ADBNDEND contains a 3-bit field called BNDEND that configures the total memory available. The ADC module can support up to 1024 buffers. The device supports a maximum of 64 buffers for both the ADC modules.

Figure 22-3. FIFO Implementation



- Number of buffers for Event Group = 2 × BNDA
- Number of buffers for Group1 = 2 × (BNDB – BNDA)
- Number of buffers for Group2 = Total number of buffers – 2 × BNDB

22.3.9 How to Read the Results from the Results' Memory

The CPU can read the conversion results in one of two ways:

1. By using the conversion results memory as a FIFO queue
2. By accessing the conversion results memory directly

22.3.9.1 Reading Conversion Results from a FIFO

The conversion results for each group can be accessed via a range of addresses provided to facilitate the use of the ARM Cortex-R4 CPU's Load-Multiple (LDM) instruction. A single read performed using the LDR instruction can also be used to read out a single conversion result. The results are read out from the group's memory region as a FIFO queue by reading from any location inside this address range. The conversion result that got stored first gets read first. A result that is read from the memory in this method is removed from the memory. For example, a read from any address in the range ADEVBUFFER (offset 90h to AFh) pulls out one conversion result from the Event Group memory.

Figure 22-4. Format of Conversion Result Read from FIFO, 12-bit ADC

Offset Address Register	31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
0x90 to 0xAF ADEVBUFFER	EV_ EMPTY	Reserved										EV_CHID				
	Reserved				EV_DR											
0xB0 to 0xCF ADG1BUFFER	G1_ EMPTY	Reserved										G1_CHID				
	Reserved				G1_DR											
0xD0 to 0xEF ADG2BUFFER	G2_ EMPTY	Reserved										G2_CHID				
	Reserved				G2_DR											

Figure 22-5. Format of Conversion Result Read from FIFO, 10-bit ADC

Offset Address Register	31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
0x90 to 0xAF ADEVBUFFER	Reserved															
	EV_ EMPTY	EV_CHID					EV_DR									
0xB0 to 0xCF ADG1BUFFER	Reserved															
	G1_ EMPTY	G1_CHID					G1_DR									
0xD0 to 0xEF ADG2BUFFER	Reserved															
	G2_ EMPTY	G2_CHID					G2_DR									

Option to read channel id along with conversion result:

The application has an option to read the channel id along with the conversion result. This is controlled by the CHID field of the group's mode control register. If the option to read the channel id is not selected, the channel id field of the conversion result reads as zeros.

Protection against reading from empty FIFO:

There is also a hardware mechanism to protect the application from reading past the number of new conversion results held in the FIFO. Once all available conversion results have been read out of the FIFO by the application, a subsequent read from the FIFO causes the mechanism to indicate that the FIFO is empty by setting the EMPTY field.

Debug / Emulation Support:

For debug purposes, each conversion group also provides an address that the application can read from for extracting the group's conversion results. However, no status flags for a conversion group are affected by reading from these emulation buffer addresses. For example, reading from ADEVEMUBUFFER (offset 0xF0) returns the next result in the Event Group buffer but does not actually remove that result from the buffer or change the amount of data held in the buffer.

22.3.9.2 Reading Conversion Results Directly from the Conversion Results' Memory

The conversion result memory is part of the device's memory map. The base address for the ADC1 result memory is 0xFF3E0000, while that for the ADC2 result memory is 0xFF3A0000.

Figure 22-6. ADC Memory Mapping

ADC1	ADC2	
0xFF3E0000	0xFF3A0000	Conversion word 0
0xFF3E0004	0xFF3A0004	Conversion word 1
0xFF3E0008	0xFF3A0008	Conversion word 2
0xFF3E01F8	0xFF3A01F8	Conversion word 62
0xFF3E00FC	0xFF3A00FC	Conversion word 63

The application can identify the address ranges for each of the three memory regions for the three conversion groups after performing the segmentation as described in Section 22.3.8. It is up to the application to read the desired results from the three conversion groups. The formats of the conversion results when reading from RAM directly are shown in Figure 22-7 and Figure 22-8.

Figure 22-7. Format of Conversion Result Directly Read from ADC RAM, 12-bit ADC

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC RAM address	Reserved															channel id [4]
	channel id [3-0]				12-bit conversion result											

Figure 22-8. Format of Conversion Result Directly Read from ADC RAM, 10-bit ADC

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC RAM address	Reserved															
	Rsvd	channel id [4-0]					10-bit conversion result									

Note that there is no EMPTY field to protect the application from reading data that has been previously read.

Each group does have a separate register which holds the address in the group's result memory where the ADC will write the next conversion result. These are the ADEVRAMWRADDR, ADG1RAMWRADDR, and ADG2RAMWRADDR registers. The application can use this information to calculate how many valid conversion results are available to be read.

Benefit of reading conversion results directly from ADC RAM:

The application does not have to read out conversion results sequentially as in the case of reading from a FIFO. As a result, the application can selectively read the conversion results for any particular input channel of interest without having to read other channels' conversion results.

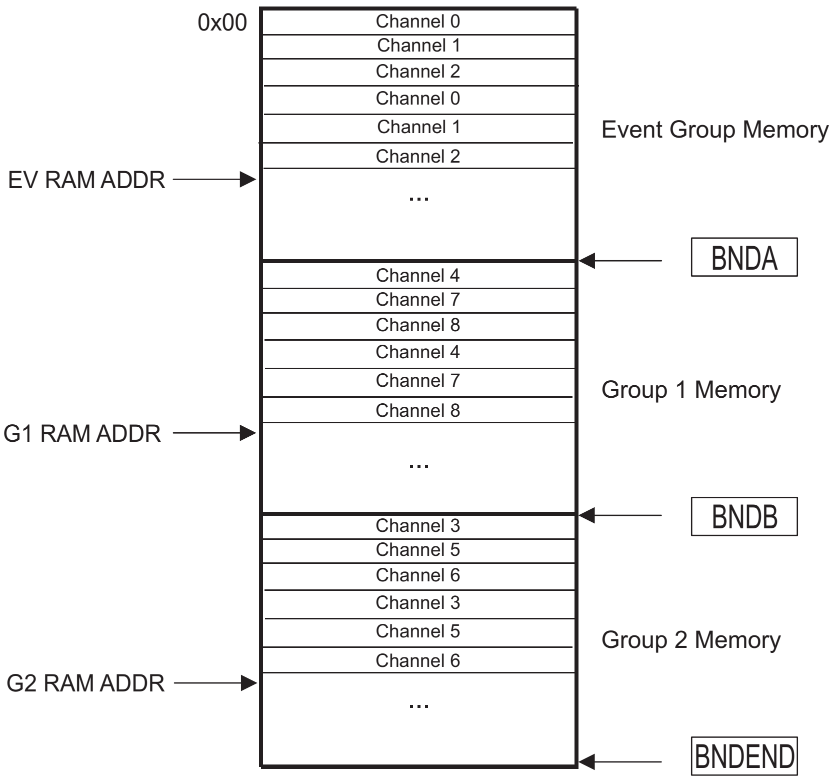
22.3.9.3 Example

Suppose that channels 0, 1, and 2 are selected for conversion in the Event Group, channels 4, 7, and 8 are selected for conversion in group 1, and channels 3, 5, and 6 are selected for conversion in group 2. The conversion results will get stored in the three memory regions as shown in Figure 22-9.

Suppose that the CPU wants to read out the results for the Event Group from a FIFO queue. The CPU needs to read from any address in the range ADEVBUFFER (offset 0x90 to 0xAF) multiple times, or do a "load multiple" from this range of addresses. This will cause the ADC to return the results for channel 0, then channel 1, then channel 2, then channel 0, and so on for each read access to this address range.

Now suppose that the application wants to read out the results for the group 1 from the RAM directly. The conversion results for the group 1 are accessible starting from address ADC RAM Base Address + BNDA. Also, it is known that the first result at this address is for the input channel 4, the next one is for input channel 7, and so on. So the application can selectively read the conversion results for only one channel if so desired.

Figure 22-9. Conversion Results Storage



22.3.10 How to Stop a Conversion

A group's conversion can be stopped by clearing the group's channel select register.

22.3.11 Example Sequence for Basic Configuration of ADC Module

The following sequence is necessary to configure the ADC to convert channels 0, 2, 4, and 8 in single-conversion mode using Group1:

1. Write 0 to the Reset Control Register (ADRSTCR) to release the module from the reset state
2. Write 1 to the ADC_EN bit of the Operating Mode Control Register (ADOPMODECR) to enable the ADC state machine
3. Configure the ADCLK frequency by programming the desired divider into the Clock Control Register (ADCLOCKCR)
4. Configure the acquisition time for the group that is to be used. For example, configure the Group1 Sampling Time Control Register (ADG1SAMP) to set the acquisition time for Group1.
5. Select the channels that need to be converted in Group1 by writing to the Group1 Channel Select Register (ADG1SEL). In this example, a value of 0x115 needs to be written to ADG1SEL in order to select channels 0, 2, 4, and 8 for conversion in Group1.
   - The ADC sequencer will start the Group1 conversions as soon as the write to the ADG1SEL register is completed.
6. Wait for the G1_END bit to be set in the Group1 Conversion Status Register (ADG1SR). This bit gets set when all the channels selected for conversion in Group1 are converted and the results are stored in the Group1 memory.
7. Read the conversion results by reading from the Group1 FIFO access location (ADG1BUFFER) or by reading directly from the Group1 results' memory.

22.4 Advanced Conversion Group Configuration Options

Figure 22-10 shows the operating mode control registers and the status registers for each of the three conversion groups. The register addresses shown are offsets from the base address. The ADC1 register frame base address is FFF7 C000h and the ADC2 register frame base address is FFF7 C200h.

Figure 22-10. ADC Groups' Operating Mode Control and Status Registers

Offset Address Register	31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
0x010 ADEVMODECR	Reserved															No Reset On ChnSel
	Reserved						EV_ DATA_FMT		Reserved		EV_ CHID	OVR_ EV_ RAM_ IGN	Rsvd	EV_ 8BIT	EV_ MODE	FRZ_ EV
0x014 ADG1MODECR	Reserved															No Reset On ChnSel
	Reserved						G1_ DATA_FMT		Reserved		G1_ CHID	OVR_ G1_ RAM_ IGN	Rsvd	G1_ 8BIT	G1_ MODE	FRZ_ G1
0x018 ADG2MODECR	Reserved															No Reset On ChnSel
	Reserved						G2_ DATA_FMT		Reserved		G2_ CHID	OVR_ G2_ RAM_ IGN	Rsvd	G2_ 8BIT	G2_ MODE	FRZ_ G2
0x06C ADEVSR	Reserved															
	Reserved												EV_ MEM_ EMPTY	EV_ BUSY	EV_ STOP	EV_ END
0x070 ADG1SR	Reserved															
	Reserved												G1_ MEM_ EMPTY	G1_ BUSY	G1_ STOP	G1_ END
0x074 ADG2SR	Reserved															
	Reserved												G2_ MEM_ EMPTY	G2_ BUSY	G2_ STOP	G2_ END
0x19C ADEVCURRCOUNT	Reserved															
	Reserved											EV_CURRENT_COUNT				
0x1A0 ADEVMAXCOUNT	Reserved															
	Reserved											EV_MAX_COUNT				
0x1A4 ADG1CURRCOUNT	Reserved															
	Reserved											G1_CURRENT_COUNT				
0x1A8 ADG1MAXCOUNT	Reserved															
	Reserved											G1_MAX_COUNT				
0x1AC ADG2CURRCOUNT	Reserved															
	Reserved											G2_CURRENT_COUNT				
0x1B0 ADG2MAXCOUNT	Reserved															
	Reserved											G2_MAX_COUNT				

22.4.1 Group Trigger Options

The Group1 and Group2 operating mode control registers have an extra control bit: HW_TRIG. This bit configures the group to be hardware event-triggered instead of software-triggered, which is the default.

When a group is configured to be event-triggered, the group conversion starts when at least one channel is selected for conversion in this group, and when the defined event trigger occurs. The event trigger source is defined for each group in the ADEVSRC, ADG1SRC, and the ADG2SRC registers. The actual connections used as the event trigger sources are defined in the device datasheet for both the ADC modules.

22.4.2 Analog Input Channel Selection Mode Options

The ADC modules on this device support two different modes for selecting the analog input channel to be converted. These are the sequential channel selection mode (default), and the enhanced channel selection mode. These two modes are now described:

22.4.2.1 Sequential Channel Selection Mode

This is the default mode and allows the ADC module to be used in a backwards compatible mode to the ADC module on other Hercules ARM Safety MCUs from Texas Instruments. As discussed in Section 22.3.4, an analog input channel can be selected for conversion in one or more conversion groups by setting the bit corresponding to that channel number in the group's channel select register.

22.4.2.2 Enhanced Channel Selection Mode

There are some important concepts related to the enhanced channel selection mode. These are defined first:

- Look-Up Table

  This is a 32-word deep memory-mapped region used to define the analog input channel number to be converted. The LUTs for the three groups are stacked together so that the entire LUT occupies 96 words. Each word is aligned on a 32-bit boundary. The LUTs for ADC1 start at 0xFF3E2000 and the LUTs for ADC2 start at 0xFF3A2000.

- Conversion Group Sub-Sequence

  A group sub-sequence is defined as the conversion for a set of channels that is converted on each conversion trigger. The number of channels selected for conversion in a group sub-sequence is defined by the number of bits that are set in the group's channel select register. For example, setting bits 0, 1, 29, and 31 in ADG1SEL means that each Group1 conversion sub-sequence consists of 4 conversions.

- LUT Index

  A "CURRENT_COUNT" register for each group is maintained as an index into that group's LUT. This register increments each time a channel conversion is completed. Therefore, as its name suggests, a read from this register returns the number of conversions completed since the last write to the group's channel select register. The CURRENT_COUNT register resets to all zeros under any of the following conditions:

  1. The ADC peripheral is reset via a global peripheral reset
  2. The ADC peripheral is reset via the ADC Reset Control Register,
  3. The CURRENT_COUNT becomes equal to the MAX_COUNT defined for that conversion group
  4. The application writes zeros to the CURRENT_COUNT register
  5. The conversion group's result RAM is reset

- Maximum Number of Conversions

  A "MAX_COUNT" register for each conversion group stores the maximum number of conversions to be performed before the index into a group's LUT is reset to zero. This register can be programmed to a value between 0 and 31. It is recommended to program the MAX_COUNT register with a value that is one less than a multiple of the number of channels in that group's conversion sub-sequence (number of bits that are set in the group's channel select register).

22.4.2.2.1 Look-Up Table Details

As described earlier, each conversion group has a look-up table (LUT) that is used when the enhanced channel selection mode is enabled. This look-up table starts at an offset of 8kB from the base of the ADC results RAM. The LUT holds 32 entries for each of the three conversion groups. The first 32 entries are for the event group, the next 32 entries are for Group1 and the last 32 entries are for Group2. Figure 22-11 shows an example LUT entry for the Event group.

Figure 22-11. Example Look-Up Table Entry

31 16
Reserved
R-0

15 13	12 8	7 5	4 0
Reserved	EV_EXT_CHN_MUX_SEL	Reserved	EV_INT_CHN_MUX_SEL
R-0	R/W-0	R-0	R/W-0

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

Table 22-1. ADC Look-Up Table Field Descriptions

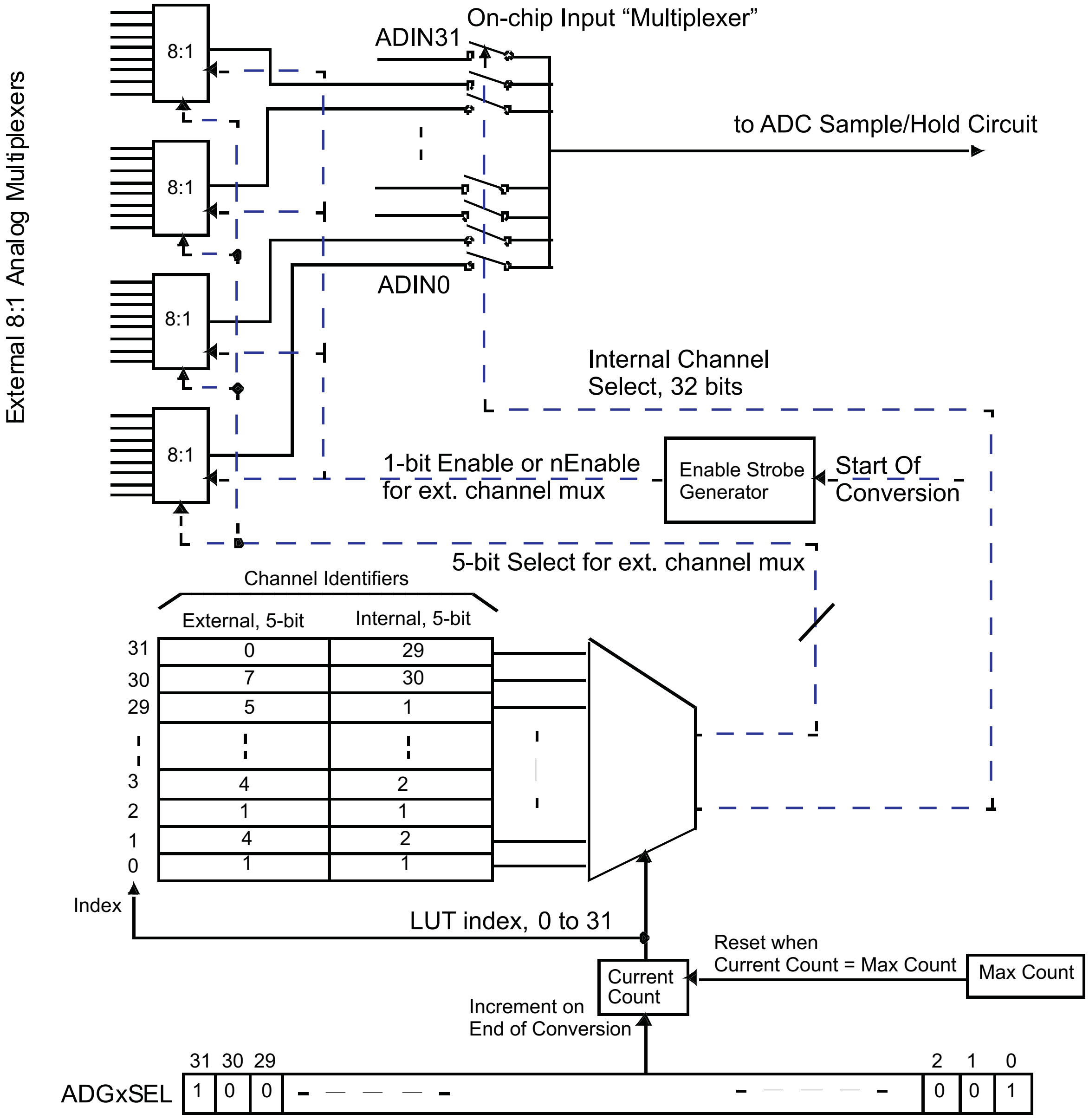
Bit	Field	Value	Description
31–13	Reserved	0	Reads return zeros, writes have no effect.
12–8	EV_EXT_CHN_MUX_SEL		This field defines the external analog mux select that is output from the ADC module when the Event group CURRENT_COUNT register points to this LUT entry, and when the Event group conversion is triggered with the enhanced channel selection mode enabled.
7–5	Reserved	0	Reads return zeros, writes have no effect.
4–0	EV_INT_CHN_MUX_SEL		This field defines the internal analog mux select that is output from the ADC module when the Event group CURRENT_COUNT register points to this LUT entry, and when the Event group conversion is triggered with the enhanced channel selection mode enabled. This can be a value between 0 and 31, which corresponds to the internal analog input channel number between 0 and 31. Note that this device only supports 24 input channels for ADC1 and 16 input channels for ADC2. If the application configures an unavailable channel number in the EV_INT_CHN_MUX_SEL field, the ADC will still perform the conversion and the result will be indeterminate.

22.4.2.2.2 Example ADC Conversion Sequence Using Enhanced Channel Selection Mode

Consider the example conversion Group1 configuration shown in Figure 22-12. Only bits 0 and 31 of ADG1SEL are set. Assume that all other bits in this register are zeros.

In case of the default sequential channel selection mode, the write to the ADG1SEL register would cause the Group1 conversions to start with channel 0 followed by channel 31. The conversions would then stop or repeat in this order depending on whether Group1 is in single or continuous conversion mode.

Figure 22-12. Group1 Enhanced Channel Selection Mode Example



	Channel Identifiers	
	External, 5-bit	Internal, 5-bit
31	0	29
30	7	30
29	5	1
⋮	⋮	⋮
3	4	2
2	1	1
1	4	2
0	1	1

Now suppose that the application has enabled the enhanced channel selection mode for Group1 with the G1_MAX_COUNT register configured to be 3. Also suppose that the application has programmed the Group1 LUT as shown in Figure 22-12. Now suppose that the application triggers Group1 conversions by writing 0x80000001 to ADG1SEL, that is, bits 0 and 31 are set and all others are zeros. The ADC conversions will proceed in the following sequence:

- Input Channel Selection

  The initial value of G1_CURRENT_COUNT is 0, which is used as the index into the Group1 LUT. The row 0 of Group1's LUT has values of 1 for the G1_EXT_CHN_MUX_SEL and a value of 1 for the G1_INT_CHN_MUX_SEL. The 5-bit external channel id of 0b00001 is driven out on the AWM1_EXT_SEL terminals. This selects channel 1 for all the connected external analog multiplexors, as shown in Figure 22-12. The ADC module also outputs an enable signal to the external analog multiplexors via the AWM1_EXT_ENA terminal.

  Now consider the fact that the internal channel id is also configured to be 1 in row 0 of the Group1 LUT. This causes the switch for ADC's internal channel 1 (ADIN1) to be closed. All other internal ADC input switches (ADIN0, ADIN2, ADIN3, ..., ADIN31) will be open. Note that the ADIN1 input channel is actually connected to the output of an 8:1 analog multiplexor.

  In effect, the ADC will convert channel 1 of the 8:1 analog multiplexor connected to the ADIN1 terminal of the microcontroller.

- After Completion of Conversion

  Once the first conversion is completed, the CURRENT_COUNT value of 0 is stored in the "channel id" field of the conversion result RAM of Group1 along with the actual conversion result from the ADC core. Then the G1_CURRENT_COUNT value of 0 is compared against the G1_MAX_COUNT value of 3. The values do not match, so that G1_CURRENT_COUNT is incremented from 0 to 1.

- Next Channel Selection

  There are two bits set in the ADG1SEL register, so that the ADC module now uses the G1_CURRENT_COUNT value of 1 to index the Group1 LUT. As shown in Figure 22-12, this row in Group1 LUT contains 4 as the G1_EXT_CHN_MUX_SEL and 2 as the G1_INT_CHN_MUX_SEL.

  ADC input channel ADIN2 is not connected to any external analog multiplexor and is connected directly to the analog signal to be converted. Note that the ADC module still drives the AWM1_EXT_ENA and the AWM1_EXT_SEL (value of 4, that is, 0b00100) to all the external analog multiplexors connected to the microcontroller.

- End of Conversion Sub-Sequence

  Once the conversion of the internal channel ADIN2 is completed, the G1_CURRENT_COUNT of 1 is stored in the "channel id" field of the Group1 result RAM along with the actual conversion result. This value of 1 is compared against the G1_MAX_COUNT value of 3. The values do not match, so that G1_CURRENT_COUNT is incremented from 1 to 2.

  There are no more conversions required in this sub-sequence as only two bits are set in ADG1SEL.

- Continuation on Next Group1 Trigger

  When the ADC Group1 is triggered again or if Group1 is in continuous conversion mode, the G1_CURRENT_COUNT of 2 is again used to index the Group1 LUT. Following the same reasoning as before, this will cause the channel 1 of the 8:1 analog multiplexor connected to ADIN1 to be converted.

  Once this conversion is done, the G1_CURRENT_COUNT value of 2 is stored in the "channel id" field of the result RAM along with the conversion result. This still does not match the G1_MAX_COUNT of 3, so that G1_CURRENT_COUNT is now incremented from 2 to 3.

  This index value of 3 is used to again convert channel ADIN2, following the same reasoning as before.

  When this conversion is completed, the G1_CURRENT_COUNT of 3 is stored as the "channel id" field of the result RAM along with the conversion result.

  Also, now this G1_CURRENT_COUNT value of 3 matches the G1_MAX_COUNT. This resets the G1_CURRENT_COUNT to zero.

The above sequence proceeds as described whenever Group1 is next triggered, or if Group1 is configured to be in a continuous conversion mode.

22.4.3 Single or Continuous Conversion Modes

The EV_MODE, G1_MODE, and G2_MODE bits are used to select between either single or continuous conversion mode for each of the three groups.

22.4.3.1 Single Conversion Mode

A conversion group configured to be in single-conversion mode gets serviced only once by the ADC for each group trigger. The trigger can be a software trigger as in the case of Group1 and Group2 by default, or it could be a hardware event trigger as in the case of the Event Group or Group1 or Group2.

The entire conversion sequence, from the acceptance of the group conversion request to the end of the last channel's conversion, is flagged for each group by the corresponding BUSY bit in that group's status register. After single-conversion mode is started, the BUSY bit is read as 1 until the conversion of the last channel is complete. The END bit for the group is set once all the channels in that group are converted.

For example, say channels 0, 2, 4, and 6 are selected for conversion in Group1 in single-conversion mode. When the Group1 gets serviced, the ADC will start conversion for channel 0, then channel 2, then channel 4, and then channel 6. It will then stop servicing the Group1, set the G1_END status bit, and look to service the Event Group or the Group2, if required.

22.4.3.2 Continuous Conversion Mode

A conversion group configured to be in continuous-conversion mode gets serviced by the ADC continuously. The group still needs to be triggered appropriately for the first conversion to start. The conversions are performed continuously thereafter.

The entire conversion sequence, from the acceptance of the group conversion request to the end of the last channel's conversion, is flagged for each group by the corresponding BUSY bit in that group's status register. After continuous-conversion mode is started, the BUSY bit is read as 1 as long as the continuous-conversion mode for this group is selected.

As an example, say the channels 0, 2, 4, and 6 are selected for conversion in Group1, now in continuous-conversion mode. When the Group1 gets serviced, the ADC will complete conversions for channels 0, 2, 4 and 6, and then look to service the Event Group or the Group2. Once it is done servicing the Event Group or the Group2, it will return to service the Group1 again. The Group1 does not need to be triggered again for the repeated conversion.

NOTE: Configuring all conversion groups in continuous conversion mode

All the three groups cannot operate in continuous-conversion mode at the same time. If the application program configures all three groups to be in continuous-conversion mode, the Group2 is automatically reset to single-conversion mode, and the G2_MODE bit in the ADG2MODECR register is cleared to reflect the single-conversion mode of Group2.

22.4.4 Conversion Group Freeze Capability

The ADC module has an inherent priority order between the three conversion groups. This group priority determines the order of conversion in case multiple groups are triggered. The priority of conversions between the three groups in descending order is:

1. Event Group
2. Group1
3. Group2

Examples of conversion group priority:

- If an Event Group conversion is ongoing in single conversion sequence mode and Group2 and Group1 conversions are requested, then the ADC will finish conversion of channels selected in Event Group, then switch over to converting channels selected in Group1, and then convert channels selected in Group2.
- If Group1 conversions are ongoing in continuous conversion mode and Group2 conversion is requested, then the ADC will complete converting the current channel for Group1 and switch over to converting channels selected in Group2. The new conversion request for Group2 has a higher priority than the pending continuous conversion request for Group1.

The conversion group freeze capability allows the application to override this default priority between the conversion groups. Enabling the freeze capability allows the ADC to freeze a higher-priority conversion group's conversions whenever there is a request for conversion in another (lower-priority) group.

For example, setting the FRZ_EV bit in the ADEVMODECR register will allow the ADC to freeze ongoing Event Group conversions whenever there is a pending request, or a new request for a Group1 or Group2 conversion. The conversions for the Event Group will be frozen as long as the Group1 or Group2 conversions are active. Once the Group1 or Group2 conversions are completed, the Event Group conversions start from where they were frozen.

While a group's conversions are frozen, the group's STOP status bit is set. This bit is cleared once the group's conversions are restarted.

22.4.5 Conversion Group Memory Overrun Option

An overrun condition occurs when the ADC module tries to store more conversion results to a group's results' memory which is already full. In this case, the ADC allows two options.

If the OVR_RAM_IGN bit in the group's operating mode control register (ADEVMODECR, ADG1MODECR, ADG2MODECR) is set, then the ADC module ignores the contents of the group's results' memory and wraps around to overwrite the memory with the results of new conversions.

If the OVR_RAM_IGN bit is not set, then the application program has to read out the group's results' memory upon an overrun condition; only then can the ADC continue to write new results to the memory.

22.4.6 Response on Writing Non-Zero Value to Conversion Group's Channel Select Register

If the application writes a non-zero value to a group's channel select register while that group's conversions are already being serviced, then that group's conversions will be restarted with the new configuration programmed in the channel select registers.

The following rules apply in terms of the effect on the ADC conversion sequence:

- If the new conversion request comes from the same group as the ongoing conversion, then the ongoing conversion will be stopped in whichever stage it is in, and the new sequence of conversions will be started.
- If the new conversion request comes from a separate group, then the ongoing channel's conversion will be completed before starting the new sequence of conversions.

The following rules apply in terms of the effect on the group's results memory:

- If a group conversion is ongoing or is frozen, writing a non-zero value to the group's channel select register will also reset its results FIFO. This does not clear the contents of the results FIFO; only the ADC module is allowed to overwrite the FIFO's contents with new conversion results starting from the first location.
- If the group conversion is completed (<GRP>_END flag is set), or the group is not being used, then writing a non-zero value to the group's channel select register will either be reset or not depending on the value of the No_Reset_on_ChnSel bit for that group (ADEVMODECR, ADG1MODECR, ADG2MODECR).
  - If the No_Reset_on_ChnSel bit is 0, then the group's FIFO will be reset.
  - If the No_Reset_on_ChnSel bit is 1, then the group's FIFO will not be reset.

22.4.7 Conversion Result Size on Reading: 8-bit, 10-bit, or 12-bit

Some applications do not need the full 12-bit resolution of the ADC modules on the device and can work with 8-bit or 10-bit conversion results.

22.4.7.1 ADC Configured in 12-bit Resolution

The mode control register for each conversion group contains a field called DATA_FMT, which defines the format of the conversion result read out of the result RAM, when accessed as a FIFO.

The DATA_FMT field is encoded as follows:

- If DATA_FMT = 00, the complete 12-bit conversion result is read out of the FIFO.
- If DATA_FMT = 01, the 12-bit conversion result is right-shifted by 2 and the resulting 10-bit result is read out of the FIFO.
- If DATA_FMT = 10, the 12-bit conversion result is right-shifted by 4 and the resulting 8-bit result is read out of the FIFO.

This control field is not effective when the application chooses to access the conversion result memory directly. In that case, the application can choose to mask off the number of bits as required.

22.4.7.2 ADC Configured in 10-bit Resolution

The DATA_FMT field is not effective in this mode and the application has the choice to read either the full 10-bit conversion result or an 8-bit conversion result. This is controlled by the 8BIT field of the group’s operating mode control register.

- If 8BIT = 0, the complete 10-bit conversion result is read out of the FIFO.
- If 8BIT = 1, the 10-bit conversion result is right-shifted by 2 and the resulting 8-bit result is read out of the FIFO.

22.4.8 Option to Read Group Channel Id Along With Conversion Result

The ADC module allows the application program to also read out the analog input channel number along with its conversion result. This capability is enabled by setting the CHID bit in the group’s operating mode control register.

- If CHID = 0, the bits [14-10] are forced to 00000 when the conversion results are read out from the group’s results’ FIFO.
- If CHID = 1, the bits [14-10] in the group’s results’ memory contain the input channel number to which the conversion result belongs.

NOTE: Actual Storage of Channel ID

Regardless of whether the CHID bit is set or not, the channel number is **always stored** in the memory along with the conversion result. The CHID bit only affects whether the channel number is available with the conversion result **when the group’s memory is read**. Therefore, the CHID bit for a group can be changed dynamically without affecting that group’s ongoing conversions.

22.5 ADC Module Basic Interrupts

This section describes the basic interrupts generated by the ADC module.

22.5.1 Group Conversion End Interrupt

The ADC module sets the group’s conversion end flag (EV_END, G1_END, or G2_END) in that group’s interrupt flag register (ADEVINTFLG, ADG1INTFLG, ADG2INTFLG) when all the channels selected for conversion in that group are converted. This causes a group conversion end interrupt to be generated if this interrupt is enabled by setting the group’s END_INT_EN control bit (EV_END_INT_EN, G1_END_INT_EN, or G2_END_INT_EN).

This interrupt can be easily used for conversion groups configured to be in the single-conversion mode. The application program can read out the conversion results, change the group’s configuration if necessary, and restart the conversions by triggering the group from within the interrupt service routine.

For groups configured to be in continuous conversion mode, this interrupt condition is not practical as the conversions are always in progress. In this case, the Group Memory Threshold Interrupt is more practical as the application can allow a programmable number of conversion results to accumulate before interrupting the CPU.

22.5.2 Group Memory Threshold Interrupt

The ADC module has the ability to generate an interrupt for a fixed number of conversions for each group. A group memory threshold register determines how many conversion results must be in a group’s memory region before the CPU is interrupted. This feature can be used to significantly reduce the CPU load when using interrupts for reading the conversion results.

The group’s threshold register needs to be configured before the group conversions are triggered. This threshold register value behaves like a down-counter, which decrements each time the ADC writes a conversion result to this group’s memory. This counter is incremented each time the application program reads a conversion result from the results’ memory by accessing the FIFO queue. Simultaneous read (by application program) and write (by ADC module) operations from the group’s results’ memory leave the threshold counter unchanged.

The threshold counter can decrement past 0 and become negative. It always increments back to its original value when the memory region is emptied. To determine how many samples are in the memory region at a given moment, the threshold counter can be subtracted from the originally configured threshold count.

Whenever the threshold counter transitions from +1 to 0, it sets the group’s threshold interrupt flag, and the CPU is interrupted if the group’s threshold interrupt is enabled. The CPU is expected to clear the interrupt flag after reading the conversion results from the memory.

The interrupt flag is not set when the threshold counter stays at 0 or transitions from -1 to 0.

22.5.3 Group Memory Overrun Interrupt

An interrupt can be generated for each group if the number of ADC conversions for that group exceed the number of buffers allocated for that conversion group. The application program can choose to read out all the conversion results using the CPU or the DMA. Alternatively, the application program can set the group’s OVR_RAM_IGN bit and allow the ADC module to overwrite the group’s results’ memory contents with new conversion results.

22.6 ADC Module DMA Requests

This section describes the capabilities of the ADC module to take advantage of the Platform DMA controller module. The ADC module can generate a DMA request under two conditions:

22.6.1 DMA Request for Each Conversion Result Written to the Results' Memory

In this mode, the ADC module will generate the first DMA request as soon as a conversion result gets written to the group's results' memory. Subsequent writes to the results' memory will cause DMA requests to be generated. This mode allows a smaller amount of ADC results' memory to suffice for an application.

This DMA request generation is enabled by setting the group's DMA_EN bit in the group's DMA control register. The BLK_XFER bit in this register must be left cleared (default) if a DMA request is desired to be generated for new results getting written to the results' memory.

22.6.2 DMA Request for a Fixed Number of Conversion Results

This mode is enabled by setting both the group's DMA_EN and the group's BLK_XFER bits in the group's DMA control registers.

In this mode, a DMA request will be generated for a specified number of conversion results being available in the group's results' memory. The number of conversion results desired are configured using the group's BLOCKS field in the control registers.

For example, if the BLOCK count is configured for 10, then ADC module will generate a DMA request at the end of 10th conversion. DMA controller should complete reading out 10 data before next set of 10 conversions complete.

NOTE: Usage of Block DMA transfers with Threshold Interrupts

It is not recommended to enable the block DMA transfers for a group at the same time as the group threshold interrupt. The group's BLOCKS field is essentially the same as the group's THRESHOLD field in the group's interrupt control register described in Section 22.5.2.

22.7 ADC Magnitude Threshold Interrupts

The ADC allows up to three magnitude threshold interrupts to be generated. The comparison parameters are programmed via the Magnitude Compare Interrupt Control Register (ADMAGINTxCR).

22.7.1 Magnitude Threshold Interrupt Configuration

The following fields are configurable for each of the three available magnitude threshold interrupts:

1. CHN_THR_COMP: Specifies whether to compare two channels' conversion results, or to compare a channel's conversion result to a programmable threshold value. A value of 0 will select the programmable threshold to be compared, and a value of 1 will select the conversion result of the channel identified by the COMP_CHID field to be compared.
2. MAG_CHID: Specifies the channel number from 0 to 23 whose conversion result needs to be monitored.
3. COMP_CHID: Specifies the channel number from 0 to 23 whose last conversion result is used for the comparison with the conversion result of the channel being monitored.
4. MAG_THR: Specifies the value for comparison with the conversion result of the channel identified by the MAG_CHID field.
5. CMP_GE_LT: Specifies whether the conversion result of the channel identified by MAG_CHID is compared to be "greater than or equal to", or "less than" the reference value. The reference value can be the conversion result of another channel identified by the COMP_CHID field, or it could be a threshold value specified in the MAG_THR field. A value of 0 in the CMP_GE_LT field indicates a "less than" comparison and a value of 1 indicates a "greater than or equal to" comparison.

22.7.2 Magnitude Threshold Interrupt Comparison Mask Configuration

There is also a separate comparison mask register (ADMAGINTxMASK) for each of the three magnitude threshold interrupts. This register is used to specify the bits that are masked off for the sake of the comparison. For example, the lower 4 bits of the conversion result can be masked off by writing 0xf to the interrupt comparison mask register, allowing a gross comparison to be made. By default, the full 10/12-bit conversion results are compared.

22.7.3 Magnitude Threshold Interrupt Enable / Disable Control

Each of the three magnitude interrupts also have separate interrupt enable set (ADMAGINTENASET) and clear (ADMAGINTENACLR) registers. These are used to respectively enable and disable that particular magnitude threshold interrupt from being generated. To enable a magnitude threshold interrupt, write a 1 to the corresponding bit of the interrupt enable set register. Conversely, to disable a magnitude threshold interrupt, write a 1 to the corresponding bit of the interrupt enable clear register.

22.7.4 Magnitude Threshold Interrupt Flags

There is a separate Magnitude Interrupt Flag register (ADMAGINTFLG) that holds the flags for these three interrupts. This flag gets set whenever the comparison condition for the corresponding interrupt is met. A magnitude threshold interrupt is generated if the corresponding flag is set inside the flag register, and the interrupt generation is enabled. This flag can be cleared by writing a 1 to the flag or by reading from the interrupt offset register in case of this interrupt being the current highest-priority pending interrupt.

22.7.5 Magnitude Threshold Interrupt Offset Register

It is possible to have multiple magnitude threshold interrupts pending at the same time. The magnitude threshold interrupt offset register (ADMAGINTOFF) holds the index of the currently pending highest priority magnitude threshold interrupt. The magnitude threshold interrupt 1 has the highest priority while the magnitude threshold interrupt 3 has the lowest priority. This is a read-only register and returns zeros if none of the magnitude threshold interrupts are pending. Writes to this register have no effect.

A read from this register updates the register to the next highest-priority pending magnitude threshold interrupt. This read also clears the corresponding flag from the magnitude threshold interrupt flag register. However, a read from the magnitude threshold interrupt offset register in emulation mode does not affect the interrupt flag register or the interrupt offset register.

22.8 ADC Special Modes

The ADC module supports some special modes for diagnostics and power saving purposes.

22.8.1 ADC Error Calibration Mode

The application program can activate a calibration sequence any time self-test mode is disabled (SELF_TEST = 0). This calibration sequence includes the conversion of an embedded calibration reference voltage followed by the calculation of an offset error correction value.

NOTE: Disable Self-Test Mode Before Calibration

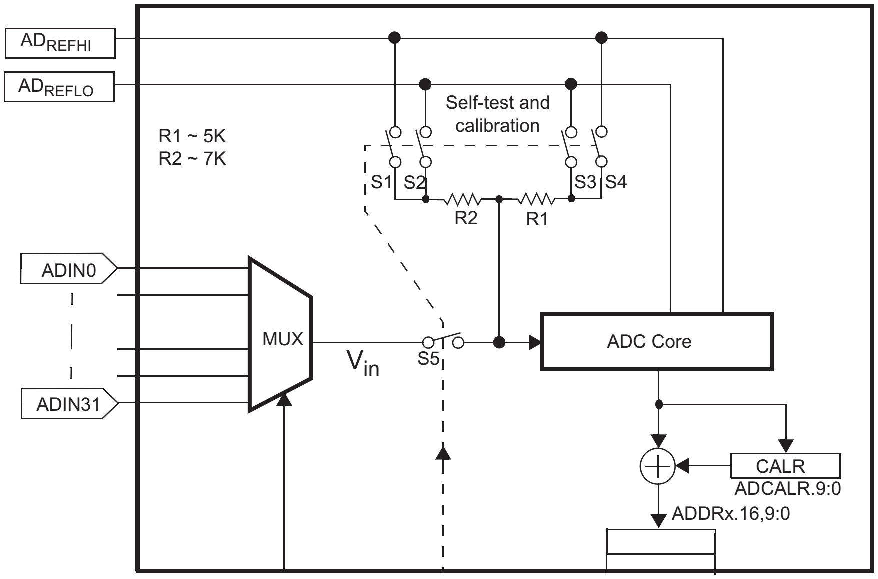
To avoid errors during the calibration operation, self-test mode must *not* be enabled during a calibration sequence. In addition, to ensure accurate results, calibrate the ADC in an environment with minimum noise.

Calibration mode is enabled by setting the CAL_EN bit (ADCALCR.0). The application needs to ensure that no conversion group is being serviced when the calibration mode is enabled.

The input multiplexor gets disabled and only the reference voltage is connected to the ADC core input. Switch S5 of Figure 22-13 is opened. In addition, the digital result issued from a conversion is output from the ADC core to the calibration and offset error correction register, ADCALR. The ADC results' memory is not affected by the calibration conversion.

When calibration mode is disabled, the ADC can be configured for normal conversions.

Figure 22-13. Self-Test and Calibration Logic



22.8.1.1 Calibration Conversion

The calibration conversion also needs to meet the minimum sampling time specification for the ADC. This value is typically 1 us. The Event Group sample time register (ADEVSAMP) is used to specify the number of ADCLK cycles for the calibration conversion.

The BRIDGE EN and HILO bits (ADCALCR.9:8) control the voltage to the calibration reference device shown in Figure 22-15. The positions of the switches in calibration mode are listed in Table 22-2.

Table 22-2. Calibration Reference Voltages[1]

CAL_EN	BRIDGE_EN	HILO	S1	S2	S3	S4	S5	Reference Voltage
1	0	0	1	0	1	0	0	$(AD_{REFHI} \times R1 + AD_{REFLO} \times R2) / (R1 + R2)$
1	0	1	0	1	0	1	0	$(AD_{REFLO} \times R1 + AD_{REFHI} \times R2) / (R1 + R2)$
1	1	0	0	1	1	0	0	AD_{REFLO}
1	1	1	1	0	0	1	0	AD_{REFHI}
0	X	X	0	0	0	0	1	V_{in}

[1] The state of the switches in this table assumes that self-test mode is not enabled.

When CAL_ST (ADCALCR.16) is set, a calibration conversion is started. The voltage source selected via the bits BRIDGE_EN and HILO is converted once (single conversion mode) and the digital result is returned to the calibration and correction register, ADCALR, where it can be read by the CPU. The CAL_ST bit acts as a flag and must be polled by the CPU. It is held set during the conversion process and automatically clears to indicate the end of the reference voltage conversion.

NOTE: No Interrupt for end of calibration

The ADC does not generate an interrupt to signal the end of the calibration conversion. The application must poll the CAL_ST bit to determine the end of the calibration conversion.

After the CAL_ST bit is set by the application program, it can only be reset by the end of the ongoing conversion generated by the ADC core. If the calibration conversion is interrupted (CAL_EN bit is cleared), the CAL_ST bit is held at 1 until a new calibration conversion has been set and completed. Setting the CAL_ST bit while calibration is disabled (CAL_EN = 0) has no effect; however, in this situation, setting CAL_EN immediately starts a calibration conversion. When the calibration conversion is interrupted by an ADC enable (ADC_EN = 0, CAL_EN = 1, and CAL_ST = 1), a new conversion is automatically restarted as soon as the ADC enable bit is released (ADC_EN = 1).

22.8.1.2 Calibration and Offset Error Correction Sequences

The number of measurements and the source to measure for an ADC calibration are application dependent. The CAL_ST bit must be set for each calibration source to be measured. While calibration mode is enabled, any available calibration sources can be converted according to the BRIDGE_EN and HILO bits (see Table 22-2). The digital results of the calibration measurements should be read from ADCALR by the application after each reference conversion so that a correction value can be computed and written back into ADCALR.

When the application has the necessary calibration data, it should compute the offset error correction value and load it into the calibration and correction register, ADCALR. After the CAL_EN bit is cleared, normal conversion mode restarts, continuing from where it was frozen, but with the addition of self-correction data.

In normal mode, the self-correction system adds the correction value stored in ADCALR to each digital result before it is written to the respective group’s FIFO.

The basic calibration routine is as follows:

1. Enable calibration via CAL_EN (ADCALCR.0).
2. Select the voltage source via BRIDGE_EN and HILO (ADCALCR.9:8).
3. Start the conversion with CAL_ST (ADCALCR.16).
4. Wait for CAL_ST to go to 0.
5. Get the results from ADCALR and save to memory.
6. Loop to step 2 until the calibration conversion data is collected for the desired reference voltages.
7. Compute the error correction value using calibration data saved in memory.
8. Load the ADCALR register with the 2s complement of the computed error correction value.
9. Disable calibration mode.

At this point, the ADC can be configured for normal operation, and it corrects each digital result with the error correction value loaded in ADCALR.

NOTE: Prevent ADC Calibration Data From Being Overwritten

In calibration mode, the conversion result is written to ADCALR which overwrites any previous calibration data; therefore, the ADCALR register must be read before a new conversion is started.

For no correction, a value of 0x0000 must be written to ADCALR. In noncalibration mode, the ADCALR register can be read and written. Any value written to ADCALR in normal mode (CAL_EN = 0) is added to each digital result from the ADC core.

22.8.1.3 Mid-Point Calibration

Because of its connections to the ADC’s reference voltage (VrefHi, VrefLo), the precision of the calibration reference is voltage independent. On the other hand, the accuracy of the switched bridge resistor (R1 and R2) relies on the manufacturing process deviation. Consequently, the mid-point voltage’s accuracy can be affected due to the imperfections in the two resistors (expected mismatch error is around 1.5%).

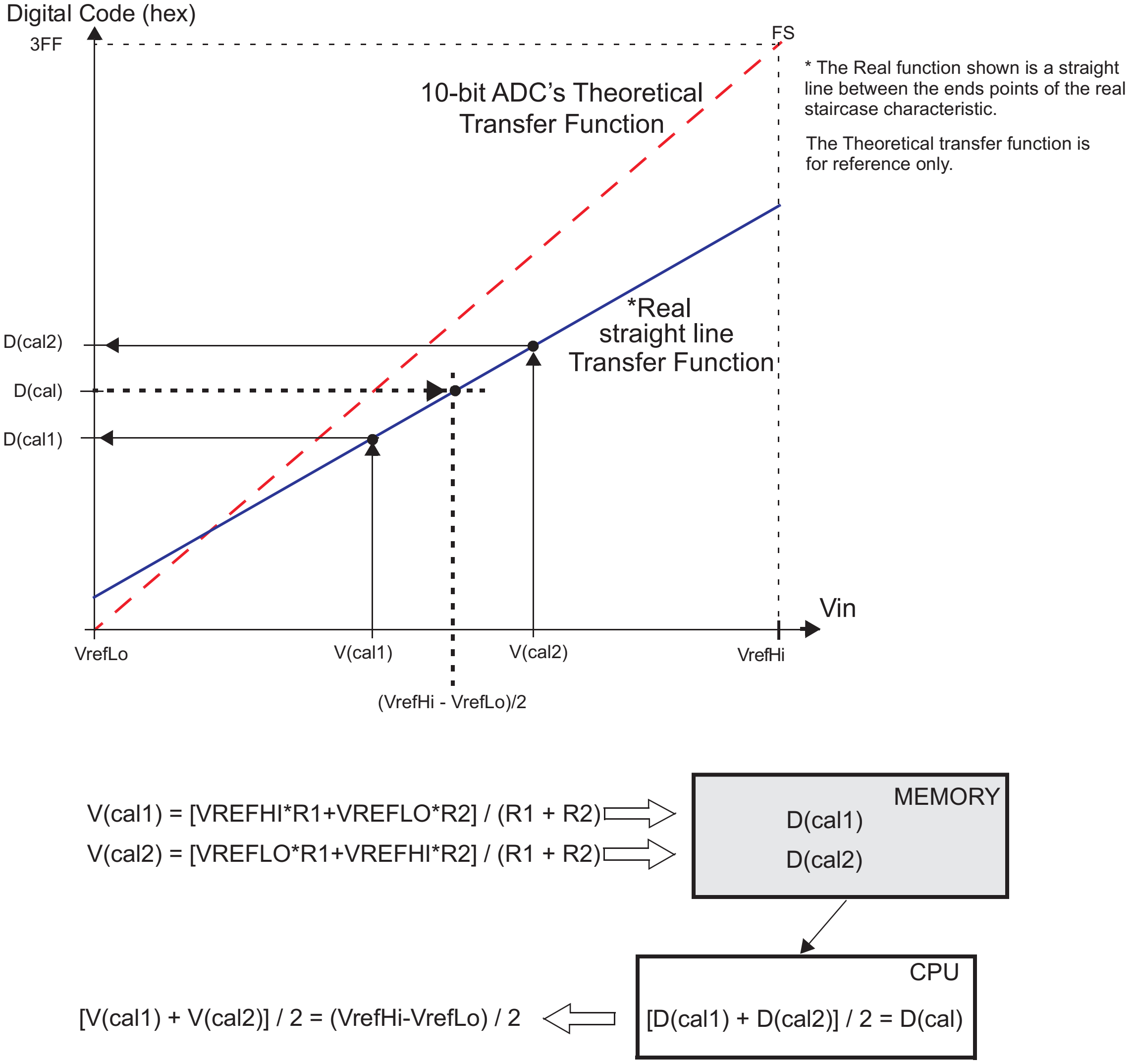
The switched reference voltage device has been specially designed to support a differential measurement of its mid-point voltage. This ensures the accuracy of the mid-point reference, and hence the efficiency of the calibration.

The differential mid-point calibration is software controlled; the algorithm (voltage source measurements and associated calculation) is inserted within the calibration software module included in the application program.

The basic differential mid-point calibration flow is illustrated here after:

1. The application program connects the voltage VrefHi to R1 and VrefLo to R2, (BRIDGE_EN = 0, HILO = 0), launches a conversion of the input voltage V(cal1), and stores the digital result D(cal1) into the memory.
2. Then the application program switches the voltage VrefHi to R2 and VrefLo to R1 (BRIDGE_EN = 0, HILO = 1), converts this new input voltage V(cal2) and again stores the issued digital result D(cal2) into the memory.
3. The actual value of the real middle point is obtained by computing the average of these two results. [D(cal1)+D(cal2)] /2; Figure 22-14 summarizes the mid-point calibration flow.

Figure 22-14. Mid-point Value Calculation

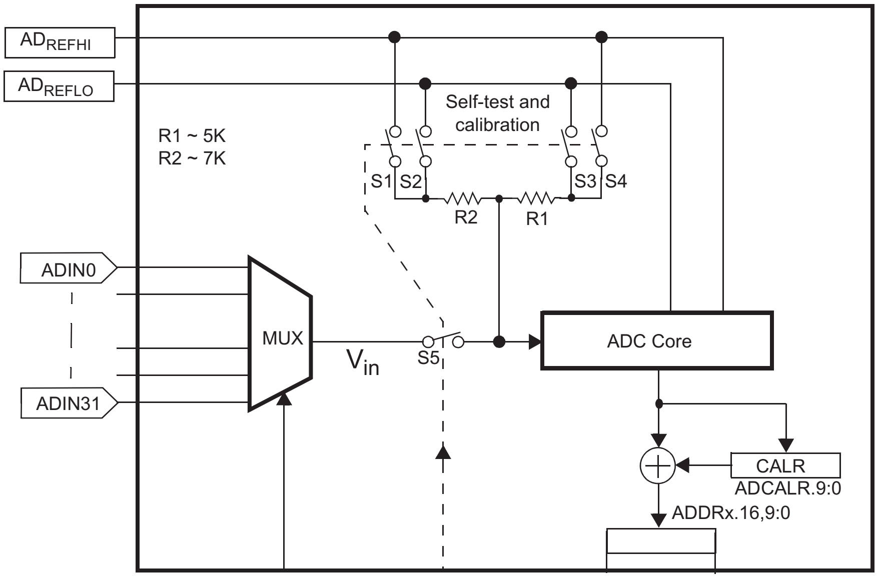

Digital Code (hex)
3FF
FS
10-bit ADC's Theoretical Transfer Function
* The Real function shown is a straight line between the ends points of the real staircase characteristic.
The Theoretical transfer function is for reference only.
*Real straight line Transfer Function
D(cal2)
D(cal)
D(cal1)
Vin
VrefLo
V(cal1)
V(cal2)
VrefHi
(VrefHi - VrefLo)/2
V(cal1) = [VREFHI*R1+VREFLO*R2] / (R1 + R2)
V(cal2) = [VREFLO*R1+VREFHI*R2] / (R1 + R2)
MEMORY
D(cal1)
D(cal2)
CPU
[D(cal1) + D(cal2)] / 2 = D(cal)
[V(cal1) + V(cal2)] / 2 = (VrefHi-VrefLo) / 2

22.8.2 ADC Self-Test Mode

The ADC module supports a self-test mode which can be used to detect an open or a short on the ADC input channels. Self-test mode is enabled by setting the SELF_TEST bit (ADCALCR.24). Any conversion type (continuous or single conversion, freeze enabled or non-freeze enabled, interrupts enabled or disabled) can be performed in this mode.

In normal mode, setting the self-test mode while a conversion sequence is in process can corrupt the current channel conversion results. However, the next channel in the sequence is converted correctly during the additional self-test cycle. The logic associated with both self-test and calibration is shown in Figure 22-15.

Figure 22-15. Self-Test and Calibration Logic



In self-test mode, a test voltage defined by the HILO bit (ADCALCR.8) is provided to the ADC core input through a resistor (see Table 22-3). To change the test source, this bit can be toggled before any single conversion mode request. Changing this bit while a conversion is in progress *can* corrupt the results if the source switches during the acquisition period.

Note that the switch S5 shown in Figure 22-15 is only for the purpose of explaining the self-test sequence. There is no physical switch.

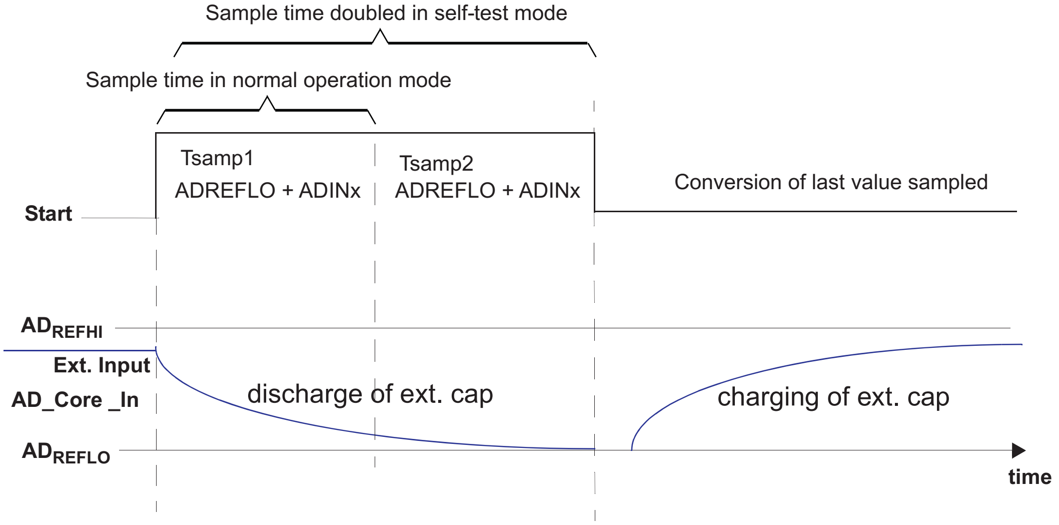
Table 22-3. Self-Test Reference Voltages[1]

SELF_TEST	HILO	S1	S2	S3	S4	S5	Reference Voltage
1	0	0	1	1	0	1	AD_{REFLO} via R1 || R2 connected to V_{in}
1	1	1	0	0	1	1	AD_{REFHI} via R1 || R2 connected to V_{in}
0	X	0	0	0	0	1	V_{in}

[1] Switches refer to Figure 22-15.

Conversions in self-test mode are started just as they are in the normal operating mode (see Section 22.3.6). The conversion starts according to the configuration set in the three mode control registers (ADEVMODECR, ADG1MODECR, ADG2MODECR) and the sampling time control registers (ADEVSAMP, ADG1SAMP, ADG2SAMP). The acquisition time for each conversion in self-test mode is extended to twice the normal configured acquisition time. The selected reference voltage and the input voltage from the ADINx input channel are both connected to the ADC internal sampling capacitor throughout this extended acquisition period. Figure 22-16 shows the self-test mode timing when the ADREFLO is chosen as the reference voltage for the self-test mode conversion. It also assumes an external capacitor connected to the ADC input channel.

Figure 22-16. Timing for Self-Test Mode



22.8.2.1 Use of Self-Test Mode to Determine Open/Short on ADC Input Channels

The following sequence needs to be used to deduce the ADC pin status:

- Convert the channel with self test enabled and with the reference voltage as Vreflo. Store the conversion result, say Vd.
- Convert the channel with self test enabled and with the reference voltage as Vrefhi. Store the conversion result, say Vu.
- Convert the channel with self test disabled. Store the conversion result, say Vn.

The results can be interpreted using the following table.

Table 22-4. Determination of ADC Input Channel Condition

Normal Conversion Result, Vn	Self-test Conversion Result, Vu	Self-test Conversion Result, Vd	Pin Condition
Vn	Vn < Vu < AD_{REFHI}	AD_{REFLO} < Vd < Vn	Good
AD_{REFHI}	AD_{REFHI}	approx. AD_{REFHI}	Shorted to AD_{REFHI}
AD_{REFLO}	approx. AD_{REFLO}	AD_{REFLO}	Shorted to AD_{REFLO}
Unknown	AD_{REFHI}	AD_{REFLO}	Open

22.8.3 ADC Power-Down Mode

This is an inactive mode in which the clocks to the ADC module are stopped leaving the module in a static state. The clock to the ADC core (ADCLK) is stopped whenever there are no ongoing conversions. This is the clock-gating implementation requirement. Also, the ADC module places the ADC core into the power down mode such that there is minimal current drawn from the ADC operating and reference supplies.

22.8.3.1 Powering Down Just The ADC Core

The ADC core can be individually powered down without stopping the clocks to the ADC module. This can be done by setting the POWERDOWN bit of the ADC Operating Mode Control Register (ADOPMODECR.3). Whenever a conversion is required the POWERDOWN bit must be cleared, and a minimum time $t_{d(PU-ADV)}$, (see the specific device data sheet for actual value) has to be allowed before starting a new conversion. This wait must be implemented in the application software.

22.8.3.2 Enhanced Power-Down Mode

A bit in the ADC operating mode control register, IDLE_PWRDN (ADOPMODECR.4) enables the enhanced power-down mode of the ADC.

Once this bit is set, the ADC module will power down the ADC core whenever there are no more ongoing or pending ADC conversions. The ADC core will be powered down regardless of the state of the POWERDOWN bit (ADOPMODECR.3).

The ADC module releases the ADC core from power down mode as soon as a new conversion is requested. The ADC logic state machine then has to wait for at least $t_{d(PU-ADV)}$ (see the device data sheet for actual value) before starting a new conversion. The IDLE_PWRDN bit will remain set at all times. The logic state machine can use this bit to determine that it needs to wait for a programmable number of VCLK cycles before it allows the input channel to be sampled. This time is configured by the ADC Power Up Delay Control register (ADPWRUPDLYCTRL).

If IDLE_PWRDN is not set, the ADC module does not wait for any additional delay before sampling the input channel and the application software has to take account of this required delay.

22.8.3.3 Managing Clocks to the ADC Module

The clock to the ADC module can be turned off via the appropriate Peripheral Central Resource (PCR) controller PSPWRDNSET register (check the specific device datasheet to identify the register and the bit to be set). If a conversion is ongoing when this bit is set, the ADC module will wait until the current conversion completes before allowing the ADC module clock to be stopped.

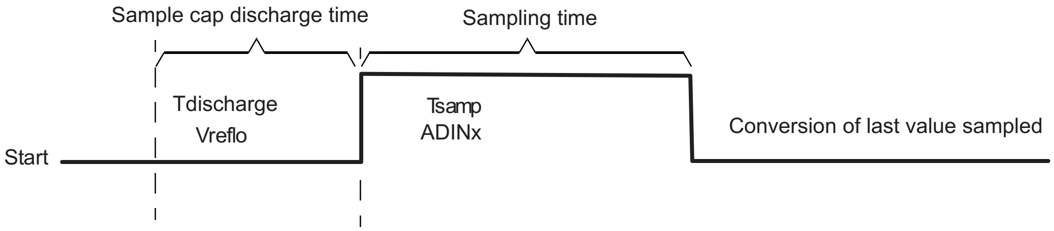
22.8.4 ADC Sample Capacitor Discharge Mode

This mode allows the charge on the ADC core's internal sampling capacitor to be discharged before starting the sampling phase of the next channel.

The ADC Sample Cap Discharge Mode is enabled by setting the SAMP_DIS_EN bit of the group's ADSAMPDISEN register. A discharge period for the sampling capacitor is added before the sampling period for each channel as shown in Figure 22-17. The duration of this discharge period is configurable via the corresponding group's SAMP_DIS_CYC field in the ADSAMPDISEN register. The discharge time is specified in terms of number of ADCLK cycles.

During the sample capacitor discharge period, the V_{REFLO} reference voltage is connected to the input voltage terminal of the ADC core. This allows any charge collected on the sampling capacitor from the previous conversion to be discharged to ground. The V_{REFLO} reference voltage is usually connected to ground.

Figure 22-17. Timing for Sample Capacitor Discharge Mode



22.9 ADC Results' RAM Special Features

The following sections describe some of the special features supported by the ADC module to enhance the results' RAM testability and integrity.

22.9.1 ADC Results' RAM Auto-Initialization

The ADC module allows the application to auto-initialize the ADC results' RAM to all zeros. The application must ensure that the ADC module is not in any of the conversion modes before triggering off the auto-initialization process.

The auto-initialization sequence is as follows:

1. Enable the global hardware memory initialization key by programming a value of 1010 to the bits [3-0] of the MINITGCR register of the System module.
2. Set the control bit for the ADC results' RAM in the MSINENA System module register. The bit 8 of the MSINENA register is used to control the initialization of the ADC1 results' RAM, while bit 14 controls the initialization of the ADC2 results' RAM. This starts the initialization process. The BUF_INIT_ACTIVE flag in the ADC module ADBNDEND register will get set to reflect that the initialization is ongoing.
3. When the memory initialization is completed, the corresponding status bit in the MINISTAT register will be set. Also, the BUF_INIT_ACTIVE flag will get cleared.

22.9.2 ADC Results' RAM Test Mode

In the defined conversion modes of the ADC, the application can only read from the ADC results' RAM. Only the ADC module is allowed to write to the results' RAM. A special test mode is defined to allow the application to also write into the ADC results' RAM - this mode is the ADC Results' RAM Test Mode. Only 32-bit reads and writes are allowed to the ADC results' RAM in this test mode.

NOTE: Contention on access to ADC Results' RAM

The ADC module cannot handle a contention between the application write to the results' RAM and the ADC writing a conversion result to the results' RAM. The application must ensure that the ADC is not likely to write a new conversion result to the results' RAM when the ADC Results' RAM Test Mode is enabled.

The ADC Results' RAM Test Mode is enabled by setting the RAM_TEST_EN bit in the ADOPMODECR.

22.9.3 ADC Results' RAM Parity

The below shows the ADC Results' RAM parity control registers.

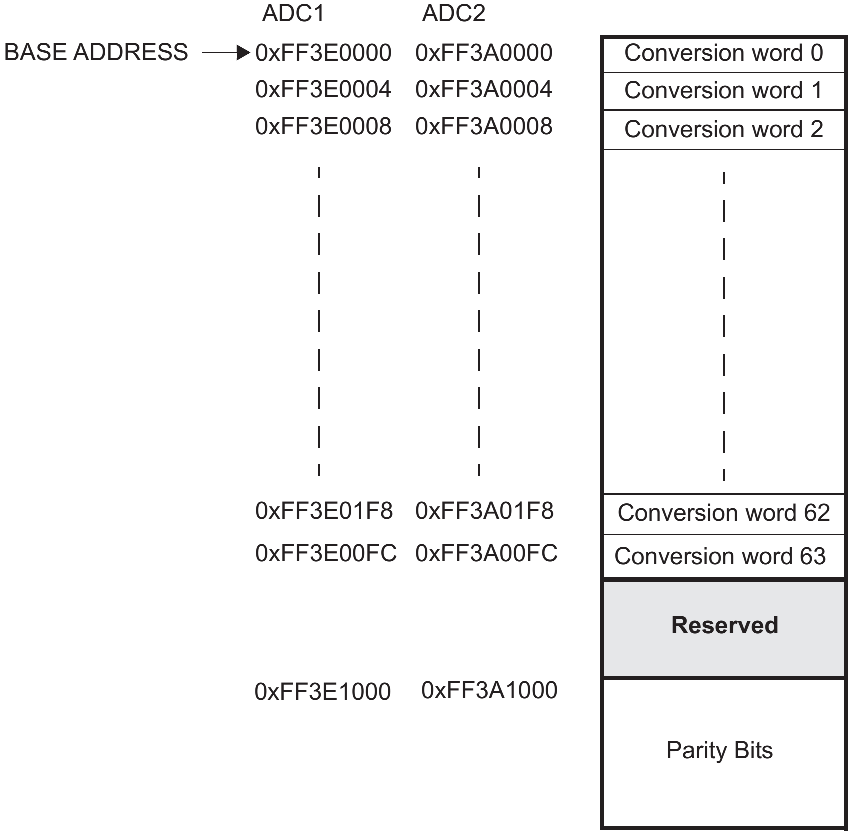
Parity checking is implemented using parity on a per-half word basis for the ADC RAM. That is, there is one parity bit for 16 bits of the ADC RAM. The polarity of the ADC RAM parity is controlled by the DEVCR1 register in the system module (address = 0xFFFFFFDC). The parity checking is enabled by the ADPARCR register. After reset, the parity checking is disabled and must be enabled if parity protection is required.

During a read access, the parity is calculated based on the data read from the ADC RAM and compared with the good parity value stored in the parity bits. If any word fails the parity check then the ADC generates an error signal hooked up to the Error Signaling Module (ESM). The ADC RAM address which generated the parity error is captured for host system debugging, and is frozen from being updated until it is read by the application.

Testing the Parity Checking Mechanism:

To test the parity checking mechanism itself, the parity RAM is made writable by the CPU in a special test mode. This is done by a control bit called TEST in the AD PAR CR register. Once this bit is set, the parity bits are mapped to an address starting at an address offset of 4KB from the base address of the ADC RAM. See Figure 22-18. The CPU can now manually insert parity errors. Note that the ADC RAM only supports 32-bit accesses.

Figure 22-18. ADC Memory Map in Parity Test Mode



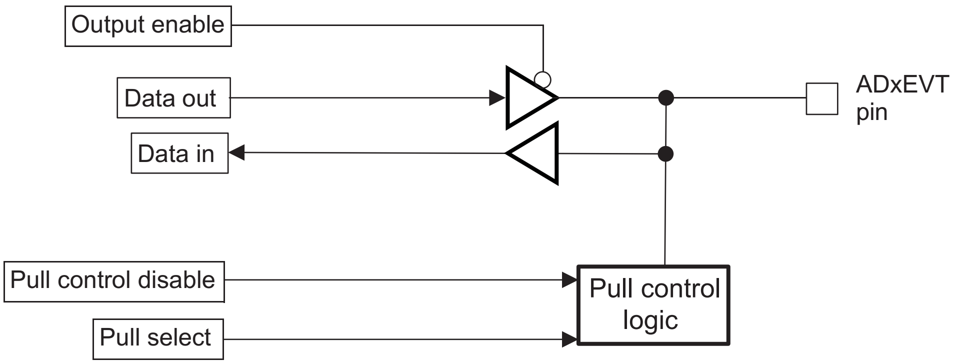
22.10 ADEVT Pin General Purpose I/O Functionality

The AD1EVT pin for ADC1 and AD2EVT pin for ADC2 can be configured as general-purpose I/O signals. The following sections describe the different ways in which the application can configure the ADxEVT pins.

22.10.1 GPIO Functionality

Figure 22-19 illustrates the GPIO functionality of the ADxEVT pin.

Figure 22-19. GPIO Functionality of ADxEVT



Once the device power-on reset is released, the ADC module controls the state of the ADxEVT pin.

- **Pull control:** The pull control can either be enabled or disabled by default (while system reset is active and after it is released). The actual default state of the pull control is specified in the device datasheet. The application can enable pull control by clearing the PDIS (pull control disable) bit in the ADEVTPDIS register. In this case, if the PSEL (pull select) bit in the ADEVTPSEL register is set, the pin will have a pull-up. If the PSEL bit is cleared, the pin will have a pull-down. If the PDIS bit is set in the control register, there is no pull-up or pull-down on the pin.

NOTE: Pull Behavior when ADxEVT is configured as output

If the ADxEVT pin is configured as output, then the pulls are disabled automatically. If the pin is configured as input, the pulls are enabled or disabled depending on bit PDIS in the pull disable register ADEVTPDIS.

- **Output buffer:** The ADxEVT pin can be driven as an output pin if the ADEVTDIR bit is set in the pin direction control register.
- **Open-Drain Feature:** The open drain output capability is enabled via the ADEVTPDR control register. The ADxEVT pin must be also configured to be an output pin for this mode.
  - The output buffer is enabled if a low signal is being driven on to the pin.
  - The output buffer is disabled if a high signal is being driven on to the pin.

22.10.2 Summary

The behavior of the output buffer, and the pull control is summarized in Table 22-5. The input buffer for the ADxEVT pins are enabled once the device power-on reset is released.

Table 22-5. Output Buffer and Pull Control Behavior for ADxEVT as GPIO Pins

System Reset Active?	Pin Direction (DIR) [1][2]	Pull Disable (PDIS) [1][3]	Pull Select (PSEL) [1][4]	Pull Control	Output Buffer
Yes	X	X	X	Enabled	Disabled
No	0	0	0	Pull down	Disabled
No	0	0	1	Pull up	Disabled
No	0	1	0	Disabled	Disabled
No	0	1	1	Disabled	Disabled
No	1	X	X	Disabled	Enabled

[1] X = Don't care

[2] DIR = 0 for input, 1 for output

[3] PULDIS = 0 for enabling pull control, 1 for disabling pull control

[4] PULSEL = 0 for pull-down functionality, 1 for pull-up functionality

22.11 ADC_CONTROL Registers

Table 22-6 lists the memory-mapped registers for the ADC_CONTROL. All register offset addresses not listed in Table 22-6 should be considered as reserved locations and the register contents should not be modified.

All registers in the ADC module are 32-bit, word-aligned; 8-bit, 16-bit and 32-bit accesses are allowed. The application must ensure that the reserved bits are always written as 0 to ensure software compatibility to future revisions of the module. The base address of ADC1 registers is FFF7 C000h.

Table 22-6. ADC_CONTROL Registers

Offset	Acronym	Register Name	Section
0h	ADRSTCR	ADC Reset Control Register	Section 22.11.1
4h	ADOPMODECR	ADC Operating Mode Control Register	Section 22.11.2
8h	ADCLOCKCR	ADC Clock Control Register	Section 22.11.3
Ch	ADCALCR	ADC Calibration Mode Control Register	Section 22.11.4
10h	ADEVMODECR	ADC Event Group Operating Mode Control Register	Section 22.11.5
14h	ADG1MODECR	ADC Group1 Operating Mode Control Register	Section 22.11.6
18h	ADG2MODECR	ADC Group2 Operating Mode Control Register	Section 22.11.7
1Ch	ADEVSRC	ADC Trigger Source Select Register	Section 22.11.8
20h	ADG1SRC	ADC Group1 Trigger Source Select Register	Section 22.11.9
24h	ADG2SRC	ADC Group2 Trigger Source Select Register	Section 22.11.10
28h	ADEVINTENA	ADC Event Interrupt Enable Control Register	Section 22.11.11
2Ch	ADG1INTENA	ADC Group1 Interrupt Enable Control Register	Section 22.11.12
30h	ADG2INTENA	ADC Group2 Interrupt Enable Control Register	Section 22.11.13
34h	ADEVINTFLG	ADC Event Group Interrupt Flag Register	Section 22.11.14
38h	ADG1INTFLG	ADC Group1 Interrupt Flag Register	Section 22.11.15
3Ch	ADG2INTFLG	ADC Group2 Interrupt Flag Register	Section 22.11.16
40h	ADEVTHRINTCR	ADC Event Group Threshold Interrupt Control Register	Section 22.11.17
44h	ADG1THRINTCR	ADC Group1 Threshold Interrupt Control Register	Section 22.11.18
48h	ADG2THRINTCR	ADC Group2 Threshold Interrupt Control Register	Section 22.11.19
4Ch	ADEVDMACR	ADC Event Group DMA Control Register	Section 22.11.20
50h	ADG1DMACR	ADC Group1 DMA Control Register	Section 22.11.21
54h	ADG2DMACR	ADC Group2 DMA Control Register	Section 22.11.22
58h	ADBNDCR	ADC Results Memory Configuration Register	Section 22.11.23
5Ch	ADBNDEND	ADC Results Memory Size Configuration Register	Section 22.11.24
60h	ADEVSAMP	ADC Event Group Sampling Time Configuration Register	Section 22.11.25
64h	ADG1SAMP	ADC Group1 Sampling Time Configuration Register()	Section 22.11.26
68h	ADG2SAMP	ADC Group2 Sampling Time Configuration Register	Section 22.11.27
6Ch	ADEVSR	ADC Event Group Status Register	Section 22.11.28
70h	ADG1SR	ADC Group1 Status Register	Section 22.11.29
74h	ADG2SR	ADC Group2 Status Register	Section 22.11.30
78h	ADEVSEL	ADC Event Group Channel Select Register	Section 22.11.31
7Ch	ADG1SEL	ADC Group1 Channel Select Register	Section 22.11.32
80h	ADG2SEL	ADC Group2 Channel Select Register	Section 22.11.33
84h	ADCALR	ADC Calibration and Error Offset Correction Register	Section 22.11.34
88h	ADSMSTATE	ADC State Machine Status Register	Section 22.11.35
8Ch	ADLASTCONV	ADC Channel Last Conversion Value Register	Section 22.11.36
90h to ACh	ADEVBUFFER_0 to ADEVBUFFER_7	ADC Event Group Results FIFO Register	Section 22.11.37
B0h to CCh	ADG1BUFFER_0 to ADG1BUFFER_7	ADC Group1 Results FIFO Register	Section 22.11.38
D0h to ECh	ADG2BUFFER_0 to ADG2BUFFER_7	ADC Group2 Results FIFO Register	Section 22.11.39
F0h	ADEVEMUBUFFER	ADC Event Group Results Emulation FIFO Register	Section 22.11.40

Table 22-6. ADC_CONTROL Registers (continued)

Offset	Acronym	Register Name	Section
F4h	ADG1EMUBUFFER	ADC Group1 Results Emulation FIFO Register	Section 22.11.41
F8h	ADG2EMUBUFFER	ADC Group2 Results Emulation FIFO Register	Section 22.11.42
FCh	ADEVTDIR	ADC ADEVT Pin Direction Control Register	Section 22.11.43
100h	ADEVTOUT	ADC ADEVT Pin Output Value Control Register	Section 22.11.44
104h	ADEVTIN	ADC ADEVT Pin Input Value Register	Section 22.11.45
108h	ADEVTSET	ADC ADEVT Pin Set Register	Section 22.11.46
10Ch	ADEVTCLR	ADC ADEVT Pin Clear Register	Section 22.11.47
110h	ADEVTPDR	ADC ADEVT Pin Open Drain Enable Register	Section 22.11.48
114h	ADEVTPDIS	ADC ADEVT Pin Pull Control Disable Register	Section 22.11.49
118h	ADEVTPSEL	ADC ADEVT Pin Pull Control Select Register	Section 22.11.50
11Ch	ADEVSAMPDISEN	ADC Event Group Sample Cap Discharge Control Register	Section 22.11.51
120h	ADG1SAMPDISEN	ADC Group1 Sample Cap Discharge Control Register	Section 22.11.52
124h	ADG2SAMPDISEN	ADC Group2 Sample Cap Discharge Control Register	Section 22.11.53
128h	ADMAGINT1CR	ADC Magnitude Compare Interrupt 1 Control Register	Section 22.11.54
12Ch	ADMAGINT1MASK	ADC Magnitude Compare Mask 1 Register	Section 22.11.55
130h	ADMAGINT2CR	ADC Magnitude Compare Interrupt 2 Control Register	Section 22.11.56
134h	ADMAGINT2MASK	ADC Magnitude Compare Mask 2 Register	Section 22.11.57
138h	ADMAGINT3CR	ADC Magnitude Compare Interrupt 3 Control Register	Section 22.11.58
13Ch	ADMAGINT3MASK	ADC Magnitude Compare Mask 3 Register	Section 22.11.59
158h	ADMAGINTENASET	ADC Magnitude Compare Interrupt Enable Set Register	Section 22.11.60
15Ch	ADMAGINTENACLR	ADC Magnitude Compare Interrupt Enable Clear Register	Section 22.11.61
160h	ADMAGINTFLG	ADC Magnitude Compare Interrupt Flag Register	Section 22.11.62
164h	ADMAGINTOFF	ADC Magnitude Compare Interrupt Offset Register	Section 22.11.63
168h	ADEVFIFORESETCR	ADC Event Group FIFO Reset Control Register	Section 22.11.64
16Ch	ADG1FIFORESETCR	ADC Group1 FIFO Reset Control Register	Section 22.11.65
170h	ADG2FIFORESETCR	ADC Group2 FIFO Reset Control Register	Section 22.11.66
174h	ADEVRAMWRADDR	ADC Event Group RAM Write Address Register	Section 22.11.67
178h	ADG1RAMWRADDR	ADC Group1 RAM Write Address Register	Section 22.11.68
17Ch	ADG2RAMWRADDR	ADC Group2 RAM Write Address Register	Section 22.11.69
180h	ADPARCR	ADC Parity Control Register	Section 22.11.70
184h	ADPARADDR	ADC Parity Error Address Register	Section 22.11.71
188h	ADPWRUPDLYCTRL	ADC Power-Up Delay Control Register	Section 22.11.72
190h	ADEVCHNSELMODECTRL	ADC Event Group Channel Selection Mode Control Register	Section 22.11.73
194h	ADG1CHNSELMODECTRL	ADC Group1 Channel Selection Mode Control Register	Section 22.11.74
198h	ADG2CHNSELMODECTRL	ADC Group2 Channel Selection Mode Control Register	Section 22.11.75
19Ch	ADEVCURRCOUNT	ADC Event Group Current Count Register	Section 22.11.76
1A0h	ADEVMAXCOUNT	ADC Event Group Max Count Register	Section 22.11.77
1A4h	ADG1CURRCOUNT	ADC Group1 Current Count Register	Section 22.11.78
1A8h	ADG1MAXCOUNT	ADC Group1 Max Count Register	Section 22.11.79
1ACh	ADG2CURRCOUNT	ADC Group2 Current Count Register	Section 22.11.80
1B0h	ADG2MAXCOUNT	ADC Group2 Max Count Register	Section 22.11.81

Complex bit access types are encoded to fit into small table cells. Table 22-7 shows the codes that are used for access types in this section.

Table 22-7. ADC_CONTROL Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
RC	C R	to Clear Read
Write Type		
W	W	Write
W1C	1C W	1 to clear Write
W1S	1S W	1 to set Write
WP	P W	Requires privileged access Write
Reset or Default Value		
-n		Value after reset or the default value

22.11.1 ADRSTCR Register (Offset = 0h) [reset = 0h]

ADRSTCR is shown in Figure 22-20 and described in Table 22-8.

Return to Summary Table.

Figure 22-20. ADRSTCR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							RESET
R-0h							R/WP-0h

Table 22-8. ADRSTCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reads return 0, writes have no effect.
0	RESET	R/WP	0h	This bit is used to reset the ADC internal state machines and control and status registers. This reset state is held until this bit is cleared. Read in all modes, write in privileged mode. 0h = Module is released from the reset state. 1h = All the module's internal state machines and the control/status registers are reset.

22.11.2 ADOPMODECR Register (Offset = 4h) [reset = 00140000h]

ADOPMODECR is shown in Figure 22-21 and described in Table 22-9.

Return to Summary Table.

Figure 22-21. ADOPMODECR Register

31	30	29	28	27	26	25	24
10_12_BIT	RESERVED						COS
R/W-0h	R-0h						R/W-0h
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RESERVED			CHN_TEST_EN				RAM_TEST_EN
R-0h			R/W-Ah				R/W-0h
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RESERVED							POWER_DOWN
R-0h							R/W-0h
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RESERVED			IDLE_PWRDN	RESERVED			ADC_EN
R-0h			R/W-0h	R-0h			R/W-0h

Table 22-9. ADOPMODECR Register Field Descriptions

Bit	Field	Type	Reset	Description
31	10_12_BIT	R/W	0h	This bit controls the resolution of the ADC core. It also affects the size of the conversion results stored in the results' RAM. Any operation mode: 0h = ADC core and digital logic are configured to be in 10-bit resolution. This is the default mode of operation. 1h = ADC core and digital logic are configured to be in 12-bit resolution.
30-25	RESERVED	R	0h	Reads return 0, writes have no effect.
24	COS	R/W	0h	This bit affects emulation operation only. It defines whether the ADC core clock (ADCLK) is immediately halted when the CPU enters debug mode or if it should continue operating normally. **Note: If COS = 0 when the CPU enters the emulation mode, then the accuracy of the conversion results can be affected depending on how long the module stays in the emulation mode.** Any operation mode: 0h = ADC module halts all ongoing conversions immediately after emulation mode is entered. 1h = ADC module continues all ongoing conversions as per the configurations of the three conversion groups.
23-21	RESERVED	R	0h	Reads return 0, writes have no effect.
20-17	CHN_TEST_EN	R/W	Ah	Enable the input channels' impedance measurement mode. This mode is reserved for use by TI. Any other value = Input impedance measurement mode is disabled. Any operation mode: 5h = Input impedance measurement mode is enabled. Ah = Input impedance measurement mode is disabled.
16	RAM_TEST_EN	R/W	0h	Enable the ADC Results' RAM Test Mode. Refer to ADC Results' RAM Test Mode subsection for more details. Any operation mode: 0h = ADC RAM Test Mode is disabled. The application cannot write to the ADC RAM by the CPU or the DMA. 1h = ADC RAM Test Mode is enabled. The application can directly write to the ADC RAM by the CPU or the DMA.

Table 22-9. ADOPMODECR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
15-9	RESERVED	R	0h	Reads return 0, writes have no effect.
8	POWER_DOWN	R/W	0h	ADC Power Down. This bit powers down only the ADC core, the digital logic in the sequencer stays active. To release the core from power down mode, this bit must be cleared. If a conversion is ongoing, the ADC module waits until the current conversion is completed before powering down the ADC core. Also refer to ADC Power-Up Delay Control Register (ADPWRUPDLYCTRL). Any operation mode: 0h = The state of the ADC core is controlled by the IDLE_PWRDN bit, or by a global power down mode entry. 1h = ADC core is in the power-down state.
7-5	RESERVED	R	0h	Reads return 0, writes have no effect.
4	IDLE_PWRDN	R/W	0h	ADC Power Down When Idle. When this bit is set, the ADC module automatically powers down the ADC core whenever there are no conversions ongoing or pending. This is the enhanced power-down mode. Also refer to ADC Power-Up Delay Control Register (ADPWRUPDLYCTRL). Any operation mode: 0h = The ADC stays in the normal operating mode even if no conversions are ongoing or pending. The power down state is entered only by configuring the POWER_DOWN bit or via a global power down mode entry. 1h = Enhanced power down mode is enabled.
3-1	RESERVED	R	0h	Reads return 0, writes have no effect.
0	ADC_EN	R/W	0h	ADC Enable. This bit must be set to allow the ADC module to be configured to perform any conversions. Any operation mode: 0h = No ADC conversions occur. The input channel select registers: ADEVSEL, ADG1SEL, and ADG2SEL are held at their reset values. 1h = ADC can now be configured.

22.11.3 ADCLOCKCR Register (Offset = 8h) [reset = 0h]

ADCLOCKCR is shown in Figure 22-22 and described in Table 22-10.

Return to Summary Table.

Figure 22-22. ADCLOCKCR Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="27">RESERVED</td><td colspan="5">PS</td></tr>
<tr><td colspan="27">R-0h</td><td colspan="5">R/W-0h</td></tr>
</table>

Table 22-10. ADCLOCKCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reads return 0, writes have no effect.
4-0	PS	R/W	0h	ADC Clock Prescaler. These bits define the prescaler value for the ADC core clock (ADCLK). The ADCLK is generated by dividing down the input bus clock (VCLK) to the ADC module from 0 to 1Fh. **Note: The supported range for the ADC clock frequency is specified in the device datasheet. The ADC clock prescaler must be configured to meet this datasheet specification.** Any operation mode: $t_{C(ADCLK)} = t_{C(VCLK)}$ x (PS[4:0] + 1) where $t_{C(ADCLK)}$ is the period of the ADCLK and $t_{C(VCLK)}$ is the period of the VCLK.

22.11.4 ADCALCR Register (Offset = Ch) [reset = 0h]

ADCALCR is shown in Figure 22-23 and described in Table 22-11.

Return to Summary Table.

Figure 22-23. ADCALCR Register

31	30	29	28	27	26	25	24
RESERVED							SELF_TEST
R-0h							R/W-0h
23	22	21	20	19	18	17	16
RESERVED							CAL_ST
R-0h							R/W1S-0h
15	14	13	12	11	10	9	8
RESERVED						BRIDGE_EN	HILO
R-0h						R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED							CAL_EN
R-0h							R/W-0h

Table 22-11. ADCALCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reads return 0, writes have no effect.
24	SELF_TEST	R/W	0h	ADC Self Test Enable. When this bit is Set, either AD_{REFHI} or AD_{REFLO} is connected through a resistor to the selected input channel. The desired con-version mode is configured in the group mode control registers. For more details on the ADC Self Test Mode, refer to ADC Self-Test Mode subsection. Any operation mode: 0h = ADC Self Test mode is disabled. 1h = ADC Self Test mode is enabled.
23-17	RESERVED	R	0h	Reads return 0, writes have no effect.
16	CAL_ST	R/W1S	0h	ADC Calibration Conversion Start. Setting the CAL_ST bit while the CAL_EN bit is set starts conversion of the selected reference voltage. The ADC module uses the sample time configured in the Event Group sample time configuration register (ADEVSAMP) for the calibration conversion. Any operation mode: 0h (W) = Writing 0 to this bit has no effect. 0h (R) = Calibration conversion has completed, or has not yet been started. 1h (W) = ADC module starts calibration conversion. 1h (R) = Calibration conversion is in the progress.
15-10	RESERVED	R	0h	Reads return 0, writes have no effect.
9	BRIDGE_EN	R/W	0h	Bridge Enable. When set with the HILO bit, BRIDGE_EN allows a reference voltage to be converted in calibration mode. Calibration Reference Voltages table defines the four different reference voltages that can be selected.
8	HILO	R/W	0h	ADC Self Test mode and Calibration Mode Reference Source Selection. In the ADC Self Test mode, this bit defines the test voltage to be combined through a resistor with the selected input pin voltage. Refer to ADC Self-Test Mode subsection for details on the ADC Self Test Mode. In the ADC Calibration Mode, this bit defines the reference source polarity. Refer to ADC Error Calibration Mode subsection for details on the ADC Calibration Mode. In the ADC module's normal operating mode, this bit has no effect.

Table 22-11. ADCALCR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-1	RESERVED	R	0h	Reads return 0, writes have no effect.
0	CAL_EN	R/W	0h	ADC Calibration Enable. When this bit is Set, the input channel multiplexor is disconnected and the calibration reference voltage is connected to the ADC core input. The calibration reference voltage is selected by the combination of the BRIDGE_EN and HILO. The actual conversion of this reference voltage starts when the CAL_ST bit is set. If the CAL_ST bit is already set when the CAL_EN bit is set, then the calibration conversion is immediately started. Refer to ADC Error Calibration Mode subsection for more details on the ADC calibration mode. Any operation mode: 0h = Calibration mode is disabled. 1h = Calibration mode is enabled.

22.11.5 ADEVMODECR Register (Offset = 10h) [reset = 0h]

ADEVMODECR is shown in Figure 22-24 and described in Table 22-12.

Return to Summary Table.

The format of the ADEVMODECR is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module.

Figure 22-24. ADEVMODECR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							

23	22	21	20	19	18	17	16
RESERVED							No_Reset_on_ ChnSel
R-0h							R/W-0h

15	14	13	12	11	10	9	8
RESERVED						EV_DATA_FMT	
R-0h						R/W-0h	

7	6	5	4	3	2	1	0
RESERVED		EV_CHID	OVR_EV_ RAM_IGN	RESERVED	EV_8BIT	EV_MODE	FRZ_EV
R-0h		R/W-0h	R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h

Table 22-12. ADEVMODECR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reads return 0, writes have no effect.
16	No_Reset_on_ChnSel	R/W	0h	No Event Group Results Memory Reset on New Channel Select. This bit determines whether the event group results' RAM is reset whenever a non-zero value is written to the event group channel select register. If the event group conversions are ongoing (active or frozen), then writing a non-zero value to the event group channel select register always resets the event group results RAM. Any operation mode: 0h = Event group results RAM is reset when a non-zero value is written to event group channel select register, even if event group conversions are completed. 1h = Event group results RAM is not reset when a non-zero value is written to event group channel select register, and event group conversions are completed.
15-10	RESERVED	R	0h	Reads return 0, writes have no effect.
9-8	EV_DATA_FMT	R/W	0h	Event Group Read Data Format. **Note: This field is only applicable when the ADC module is configured to be a 12-bit ADC module. This field is reserved when the module is configured as a 10-bit ADC module.** This field determines the format in which the conversion results are read out of the Event group results RAM when using the FIFO interface, that is, when reading from the ADEVBUFFER or ADEVEMUBUFFER locations. Any operation mode: 0h = Conversion results are read out in full 12-bit format. This is the default mode. 1h = Conversion results are read out in 10-bit format. Bits 11-2 of the 12-bit conversion result are returned as the 10-bit conversion result. 2h = Conversion results are read out in 8-bit format. Bits 11-4 of the 12-bit conversion result are returned as the 8-bit conversion result. 3h = Reserved. The full 12-bit conversion result is returned if programmed.

Table 22-12. ADEVMODECR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	RESERVED	R	0h	Reads return 0, writes have no effect.
5	EV_CHID	R/W	0h	Enable Channel Id for the Event Group conversion results to be read. This bit only affects the “read from FIFO” mode. The ADC always stores the channel id in the results RAM. Any 16-bit read performed in the “read from RAM” mode returns the 5-bit channel id along with the 10-bit conversion result. Any operation mode: 0h = Bits 14-10, the channel id field, of the data read from the Event Group results’ FIFO is read as 00000b. 1h = Bits 14-10, the channel id field, of the data read from the Event Group results’ FIFO contains the number of the ADC analog input to which the conversion result belongs.
4	OVR_EV_RAM_IGN	R/W	0h	This bit allows the ADC module to overwrite the contents of the Event Group results memory under an overrun condition. Any operation mode: 0h = The ADC cannot overwrite the contents of the Event Group results memory. When an overrun of this memory occurs, the software needs to read out all the contents of this memory before the ADC is able to write a new conversion result for the Event Group. 1h = When an overrun of the Event Group results memory occurs, the ADC proceeds to overwrite the contents with any new conversion results for the Event Group, starting with the first location in this memory.
3	RESERVED	R	0h	Reads return 0, writes have no effect.
2	EV_8BIT	R/W	0h	Event Group 8-bit result mode. **Note: This bit is only applicable when the ADC module is configured to be a 10-bit ADC module. This bit is reserved when the module is configured as a 12-bit ADC module.** This bit allows the Event Group conversion results to be read out in an 8-bit format. This bit only applies to the read from FIFO mode. The lower 2 bits of the 10-bit conversion result are discarded and the upper 8 bits are shifted right two places to form the 8-bit conversion result. Any operation mode: 0h = The Event Group conversion result is read out as a 10-bit value in the “read from Event Group FIFO” mode. 1h = The Event Group conversion result is read out as an 8-bit value in the “read from Event Group FIFO” mode.
1	EV_MODE	R/W	0h	Event Group Conversion Mode. This bit defines whether the input channels selected for conversion in the Event Group are converted only once per trigger, or are continuously converted. Any operation mode: 0h = The channels selected for conversion in the Event Group are converted only once when the selected event trigger condition occurs. 1h = The channels selected for conversion in the Event Group are converted continuously when the selected event trigger condition occurs.

Table 22-12. ADEVMODECR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	FRZ_EV	R/W	0h	Event Group Freeze Enable. This bit allows an Event Group conversion sequence to be frozen if a Group1 or a Group2 conversion is requested. The Event Group conversion is kept frozen while the Group1 or Group2 conversion is active, and continues from where it was frozen once the Group1 or Group2 conversions are completed. While the Event Group conversion is frozen, the EV_STOP status flag in the ADEVSR register indicates that the Event Group conversions have stopped. This bit gets cleared when the Event Group conversions resume. Any operation mode: 0h = Event Group conversions cannot be frozen. All the channels selected for conversion in the Event Group are converted before the ADC switches over to servicing any other conversion group. 1h = Event Group conversions are frozen whenever there is a request for conversion from Group1 or Group2.

22.11.6 ADG1MODECR Register (Offset = 14h) [reset = 0h]

ADG1MODECR is shown in Figure 22-25 and described in Table 22-13.

Return to Summary Table.

The format of the ADG1MODECR is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module.

Figure 22-25. ADG1MODECR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							No_Reset_on_ChnSel
R-0h							R/W-0h
15	14	13	12	11	10	9	8
RESERVED						G1_DATA_FMT	
R-0h						R/W-0h	
7	6	5	4	3	2	1	0
RESERVED		G1_CHID	OVR_G1_RAM_IGN	G1_HW_TRIG	G1_8BIT	G1_MODE	FRZ_G1
R-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

Table 22-13. ADG1MODECR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reads return 0, writes have no effect.
16	No_Reset_on_ChnSel	R/W	0h	No Group1 Results Memory Reset on New Channel Select. This bit determines whether the group1 results' RAM is reset whenever a non-zero value is written to the group1 channel select register. If the group1 conversions are ongoing (active or frozen), then writing a nonzero value to the group1 channel select register always resets the group1 results RAM. Any operation mode: 0h = Group1 results RAM is reset when a non-zero value is written to group1 channel select register, even if group1 conversions are completed. 1h = Group1 results RAM is not reset when a non-zero value is written to group1 channel select register, and group1 conversions are completed.
15-10	RESERVED	R	0h	Reads return 0, writes have no effect.
9-8	G1_DATA_FMT	R/W	0h	Group1 Read Data Format. **Note: This field is only applicable when the ADC module is configured to be a 12-bit ADC module. This field is reserved when the module is configured as a 10-bit ADC module.** This field determines the format in which the conversion results are read out of the group1 results RAM when using the FIFO interface, that is, when reading from the ADG1BUFFER or ADG1EMUBUFFER locations. Any operation mode: 0h = Conversion results are read out in full 12-bit format. This is the default mode. 1h = Conversion results are read out in 10-bit format. Bits 11-2 of the 12-bit conversion result are returned as the 10-bit conversion result. 2h = Conversion results are read out in 8-bit format. Bits 11-4 of the 12-bit conversion result are returned as the 8-bit conversion result. 3h = Reserved. The full 12-bit conversion result is returned if programmed.

Table 22-13. ADG1MODECR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	RESERVED	R	0h	Reads return 0, writes have no effect.
5	G1_CHID	R/W	0h	Enable Channel Id for the Group1 conversion results to be read. This bit only affects the “read from FIFO” mode. The ADC always stores the channel id in the results RAM. Any 16-bit read performed in the “read from RAM” mode returns the 5-bit channel id along with the 10-bit conversion result. Any operation mode: 0h = Bits 14-10, the channel id field, of the data read from the Group1 results’ FIFO is read as 00000b. 1h = Bits 14-10, the channel id field, of the data read from the Group1 results’ FIFO contains the number of the ADC analog input to which the conversion result belongs.
4	OVR_G1_RAM_IGN	R/W	0h	This bit allows the ADC module to overwrite the contents of the Group1 results memory under an overrun condition. Any operation mode: 0h = The ADC cannot overwrite the contents of the Group1 results memory. When an overrun of this memory occurs, the software needs to read out all the contents of this memory before the ADC is able to write a new conversion result for the Group1. 1h = When an overrun of the Group1 results memory occurs, the ADC proceeds to overwrite the contents with any new conversion results for the Group1, starting with the first location in this memory.
3	G1_HW_TRIG	R/W	0h	Group1 Hardware Triggered. This bit allows the Group1 to be hardware triggered. The Group1 is software triggered by default. For more details on how to trigger a conversion group, refer to How to Start a Conversion subsection. Any operation mode: 0h = The Group1 is software-triggered. A Group1 conversion starts whenever the Group1 channel select register (ADG1SEL) is written with a non-zero value. 1h = The Group1 is hardware-triggered. A Group1 conversion starts whenever the Group1 channel select register has a non-zero value, and the specified hardware trigger occurs. The hardware trigger for the Group1 is specified in the Group1 Trigger Source register (ADG1SRC).
2	G1_8BIT	R/W	0h	Group1 8-bit result mode. **Note: This bit is only applicable when the ADC module is configured to be a 10-bit ADC module. This bit is reserved when the module is configured as a 12-bit ADC module.** This bit allows the Group1 conversion results to be read out in an 8-bit format. This bit only applies to the “read from FIFO” mode. The lower 2 bits of the 10-bit conversion result are discarded and the upper 8 bits are shifted right two places to form the 8-bit conversion result. Any operation mode: 0h = The Group1 conversion result is read out as a 10-bit value in the “read from Group1 FIFO” mode. 1h = The Group1 conversion result is read out as an 8-bit value in the “read from Group1 FIFO” mode.
1	G1_MODE	R/W	0h	Group1 Conversion Mode. This bit defines whether the input channels selected for conversion in the Group1 are converted only once, or are continuously converted. Any operation mode: 0h = The channels selected for conversion in the Group1 are converted only once. 1h = The channels selected for conversion in the Group1 are converted continuously.

Table 22-13. ADG1MODECR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	FRZ_G1	R/W	0h	Group1 Freeze Enable. This bit allows a Group1 conversion sequence to be frozen if an Event Group or a Group2 conversion is requested. The Group1 conversion is kept frozen while the Event Group or Group2 conversion is active, and continues from where it was frozen once the Event Group or Group2 conversions are completed. While the Group1 conversion is frozen, the G1_STOP status flag in the ADG1SR register indicates that the Group1 conversions have stopped. This bit gets cleared when the Group1 conversions resume. Any operation mode: 0h = Group1 conversions cannot be frozen. All the channels selected for conversion in the Group1 are converted before the ADC switches over to servicing any other conversion group. 1h = Group1 conversions are frozen whenever there is a request for conversion from Event Group or Group2.

22.11.7 ADG2MODECR Register (Offset = 18h) [reset = 0h]

ADG2MODECR is shown in Figure 22-26 and described in Table 22-14.

Return to Summary Table.

The format of the ADG2MODECR is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module.

Figure 22-26. ADG2MODECR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RESERVED							No_Reset_on_ChnSel
R-0h							R/W-0h
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RESERVED						G2_DATA_FMT	
R-0h						R/W-0h	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RESERVED		G2_CHID	OVR_G2_RAM_IGN	G2_HW_TRIG	G2_8BIT	G2_MODE	FRZ_G2
R-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

Table 22-14. ADG2MODECR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reads return 0, writes have no effect.
16	No_Reset_on_ChnSel	R/W	0h	No Group2 Results Memory Reset on New Channel Select. This bit determines whether the group2 results' RAM is reset whenever a non-zero value is written to the group2 channel select register. If the group2 conversions are ongoing (active or frozen), then writing a nonzero value to the group2 channel select register always resets the group2 results RAM. Any operation mode: 0h = Group2 results RAM is reset when a non-zero value is written to group2 channel select register, even if group2 conversions are completed. 1h = Group2 results RAM is not reset when a non-zero value is written to group2 channel select register, and group2 conversions are completed.
15-10	RESERVED	R	0h	Reads return 0, writes have no effect.
9-8	G2_DATA_FMT	R/W	0h	Group2 Read Data Format. **Note: This field is only applicable when the ADC module is configured to be a 12-bit ADC module. This field is reserved when the module is configured as a 10-bit ADC module.** This field determines the format in which the conversion results are read out of the group1 results RAM when using the FIFO interface, that is, when reading from the ADG2BUFFER or ADG2EMUBUFFER locations. Any operation mode: 0h = Conversion results are read out in full 12-bit format. This is the default mode. 1h = Conversion results are read out in 10-bit format. Bits 11-2 of the 12-bit conversion result are returned as the 10-bit conversion result. 2h = Conversion results are read out in 8-bit format. Bits 11-4 of the 12-bit conversion result are returned as the 8-bit conversion result. 3h = Reserved. The full 12-bit conversion result is returned if programmed.

Table 22-14. ADG2MODECR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
7-6	RESERVED	R	0h	Reads return 0, writes have no effect.
5	G2_CHID	R/W	0h	Enable Channel Id for the Group2 conversion results to be read. This bit only affects the “read from FIFO” mode. The ADC always stores the channel id in the results RAM. Any 16-bit read performed in the “read from RAM” mode returns the 5-bit channel id along with the 10-bit conversion result. Any operation mode: 0h = Bits 14-10, the channel id field, of the data read from the Group2 results’ FIFO is read as 00000b. 1h = Bits 14-10, the channel id field, of the data read from the Group2 results’ FIFO contains the number of the ADC analog input to which the conversion result belongs.
4	OVR_G2_RAM_IGN	R/W	0h	This bit allows the ADC module to overwrite the contents of the Group2 results memory under an overrun condition. Any operation mode: 0h = The ADC cannot overwrite the contents of the Group2 results memory. When an overrun of this memory occurs, the software needs to read out all the contents of this memory before the ADC is able to write a new conversion result for the Group2. 1h = When an overrun of the Group2 results memory occurs, the ADC proceeds to overwrite the contents with any new conversion results for the Group2, starting with the first location in this memory.
3	G2_HW_TRIG	R/W	0h	Group2 Hardware Triggered. This bit allows the Group2 to be hardware triggered. The Group2 is software triggered by default. For more details on how to trigger a conversion group, refer to How to Start a Conversion subsection. Any operation mode: 0h = The Group2 is software-triggered. A Group2 conversion starts whenever the Group2 channel select register (ADG2SEL) is written with a non-zero value. 1h = The Group2 is hardware-triggered. A Group2 conversion starts whenever the Group2 channel select register has a non-zero value, and the specified hardware trigger occurs. The hardware trigger for the Group2 is specified in the Group2 Trigger Source register (ADG2SRC).
2	G2_8BIT	R/W	0h	Group2 8-bit result mode. **Note: This bit is only applicable when the ADC module is configured to be a 10-bit ADC module. This bit is reserved when the module is configured as a 12-bit ADC module.** This bit allows the Group2 conversion results to be read out in an 8-bit format. This bit only applies to the “read from FIFO” mode. The lower 2 bits of the 10-bit conversion result are discarded and the upper 8 bits are shifted right two places to form the 8-bit conversion result. Any operation mode: 0h = The Group2 conversion result is read out as a 10-bit value in the “read from Group2 FIFO” mode. 1h = The Group2 conversion result is read out as an 8-bit value in the “read from Group2 FIFO” mode.
1	G2_MODE	R/W	0h	Group2 Conversion Mode. This bit defines whether the input channels selected for conversion in the Group2 are converted only once, or are continuously converted. Any operation mode: 0h = The channels selected for conversion in the Group2 are converted only once. 1h = The channels selected for conversion in the Group2 are converted continuously.

Table 22-14. ADG2MODECR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	FRZ_G2	R/W	0h	Group2 Freeze Enable. This bit allows a Group2 conversion sequence to be frozen if an Event Group or a Group1 conversion is requested. The Group2 conversion is kept frozen while the Event Group or Group1 conversion is active, and continues from where it was frozen once the Event Group or Group1 conversions are completed. While the Group2 conversion is frozen, the G2_STOP status flag in the ADG2SR register indicates that the Group2 conversions have stopped. This bit gets cleared when the Group2 conversions resume. Any operation mode: 0h = Group2 conversions cannot be frozen. All the channels selected for conversion in the Group2 are converted before the ADC switches over to servicing any other conversion group. 1h = Group2 conversions are frozen whenever there is a request for conversion from Event Group or Group1.

22.11.8 ADEVSRC Register (Offset = 1Ch) [reset = 0h]

ADEVSRC is shown in Figure 22-27 and described in Table 22-15.

Return to Summary Table.

Figure 22-27. ADEVSRC Register

<table>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="3">RESERVED</td><td>EV_EDG_
BOTH</td><td>EV_EDG_
SEL</td><td colspan="3">EV_SRC</td></tr>
<tr><td colspan="3">R-0h</td><td>R/W-0h</td><td>R/W-0h</td><td colspan="3">R/W-0h</td></tr>
</table>

Table 22-15. ADEVSRC Register Field Descriptions

Bit	Field	Type	Reset	Description
7-5	RESERVED	R	0h	Reads return 0, writes have no effect.
4	EV_EDG_BOTH	R/W	0h	EV Group Trigger Edge Polarity Select. This bit configures the event group to be triggered on both rising and falling edge detected on the selected trigger source. Any operation mode: 0h = The conversion is triggered only upon detecting an edge defined by the EV_EDGE_SEL bit. 1h = The conversion is triggered upon detecting either a rising or falling edge.
3	EV_EDG_SEL	R/W	0h	Event Group Trigger Edge Polarity Select. This bit determines the polarity of the transition on the selected source that triggers the Event Group conversion. Any operation mode: 0h = A high-to-low transition on the selected source triggers the Event Group conversion. 1h = A low-to-high transition on the selected source triggers the Event Group conversion.
2-0	EV_SRC	R/W	0h	Event Group Trigger Source. Any operation mode: 0h = The ADC module allows a trigger source to be selected for the Event Group from up to eight options. These options are device-specific and the device data manual must be referred to identify the actual trigger sources from 0 to 7h.

22.11.9 ADG1SRC Register (Offset = 20h) [reset = 0h]

ADG1SRC is shown in Figure 22-28 and described in Table 22-16.

Return to Summary Table.

Figure 22-28. ADG1SRC Register

7	6	5	4	3	2	1	0
RESERVED			G1_EDG_BOTH	G1_EDG_SEL	G1_SRC		
R-0h			R/W-0h	R/W-0h	R/W-0h		

Table 22-16. ADG1SRC Register Field Descriptions

Bit	Field	Type	Reset	Description
7-5	RESERVED	R	0h	Reads return 0, writes have no effect.
4	G1_EDG_BOTH	R/W	0h	Group1 Trigger Edge Polarity Select. This bit configures the group1 to be triggered on both rising and falling edge detected on the selected trigger source. Any operation mode: 0h = The conversion is triggered only upon detecting an edge defined by the G1_EDGE_SEL bit. 1h = The conversion is triggered upon detecting either a rising or falling edge.
3	G1_EDG_SEL	R/W	0h	Group1 Trigger Edge Polarity Select. This bit determines the polarity of the transition on the selected source that triggers the Group1 conversion. Any operation mode: 0h = A high-to-low transition on the selected source triggers the Group1 conversion. 1h = A low-to-high transition on the selected source triggers the Group1 conversion.
2-0	G1_SRC	R/W	0h	Group1 Trigger Source. Any operation mode: 0h = The ADC module allows a trigger source to be selected for the Group1 from up to eight options. These options are device-specific and the device data manual must be referred to identify the actual trigger sources from 0 to 7h.

22.11.10 ADG2SRC Register (Offset = 24h) [reset = 0h]

ADG2SRC is shown in Figure 22-29 and described in Table 22-17.

Return to Summary Table.

Figure 22-29. ADG2SRC Register

7	6	5	4	3	2	1	0
RESERVED			G2_EDG_BOTH	G2_EDG_SEL	G2_SRC		
R-0h			R/W-0h	R/W-0h	R/W-0h		

Table 22-17. ADG2SRC Register Field Descriptions

Bit	Field	Type	Reset	Description
7-5	RESERVED	R	0h	Reads return 0, writes have no effect.
4	G2_EDG_BOTH	R/W	0h	Group2 Trigger Edge Polarity Select. This bit configures the group2 to be triggered on both rising and falling edge detected on the selected trigger source. Any operation mode: 0h = The conversion is triggered only upon detecting an edge defined by the G2_EDGE_SEL bit. 1h = The conversion is triggered upon detecting either a rising or falling edge.
3	G2_EDG_SEL	R/W	0h	Group2 Trigger Edge Polarity Select. This bit determines the polarity of the transition on the selected source that triggers the Group2 conversion. Any operation mode: 0h = A high-to-low transition on the selected source triggers the Group2 conversion. 1h = A low-to-high transition on the selected source triggers the Group2 conversion.
2-0	G2_SRC	R/W	0h	Group2 Trigger Source. Any operation mode: 0h = The ADC module allows a trigger source to be selected for the Group2 from up to eight options. These options are device-specific and the device data manual must be referred to identify the actual trigger sources from 0 to 7h.

22.11.11 ADEVINTENA Register (Offset = 28h) [reset = 0h]

ADEVINTENA is shown in Figure 22-30 and described in Table 22-18.

Return to Summary Table.

Figure 22-30. ADEVINTENA Register

7	6	5	4	3	2	1	0
RESERVED				EV_END_ INT_EN	RESERVED	EV_OVR_ INT_EN	EV_THR_ INT_EN
R-0h				R/W-0h	R-0h	R/W-0h	R/W-0h

Table 22-18. ADEVINTENA Register Field Descriptions

Bit	Field	Type	Reset	Description
7-4	RESERVED	R	0h	Reads return 0, writes have no effect.
3	EV_END_INT_EN	R/W	0h	Event Group Conversion End Interrupt Enable. Refer to Group Conversion End Interrupt subsection for more details on the conversion end interrupts. Any operation mode: 0h = No interrupt is generated when conversion of all the channels selected for conversion in the Event Group is done. 1h = An Event Group conversion end interrupt is generated when conversion of all the channels selected for conversion in the Event Group is done.
2	RESERVED	R	0h	Reads return 0, writes have no effect.
1	EV_OVR_INT_EN	R/W	0h	Event Group Memory Overrun Interrupt Enable. A memory overrun occurs when the ADC tries to write a new conversion result to the Event Group results memory which is already full. For more details on the overrun interrupts, refer to Group Memory Overrun Interrupt subsection. Any operation mode: 0h = No interrupt is generated if an Event Group memory overrun occurs. 1h = An Event Group memory overrun interrupt is generated if an Event Group memory overrun condition occurs.
0	EV_THR_INT_EN	R/W	0h	Event Group Threshold Interrupt Enable. An Event Group threshold interrupt occurs when the programmed Event Group threshold counter counts down to 0. Refer to Group Memory Threshold Interrupt subsection for more details. Any operation mode: 0h = No interrupt is generated if the Event Group threshold counter reaches 0. 1h = An Event Group threshold interrupt is generated if the Event Group threshold counter reaches 0.

22.11.12 ADG1INTENA Register (Offset = 2Ch) [reset = 0h]

ADG1INTENA is shown in Figure 22-31 and described in Table 22-19.

Return to Summary Table.

Figure 22-31. ADG1INTENA Register

7	6	5	4	3	2	1	0
RESERVED				G1_END_ INT_EN	RESERVED	G1_OVR_ INT_EN	G1_THR_ INT_EN
R-0h				R/W-0h	R-0h	R/W-0h	R/W-0h

Table 22-19. ADG1INTENA Register Field Descriptions

Bit	Field	Type	Reset	Description
7-4	RESERVED	R	0h	Reads return 0, writes have no effect.
3	G1_END_INT_EN	R/W	0h	Group1 Conversion End Interrupt Enable. Refer to Group Conversion End Interrupt subsection for more details on the conversion end interrupts. Any operation mode: 0h = No interrupt is generated when conversion of all the channels selected for conversion in the Group1 is done. 1h = A Group1 conversion end interrupt is generated when conversion of all the channels selected for conversion in the Group1 is done.
2	RESERVED	R	0h	Reads return 0, writes have no effect.
1	G1_OVR_INT_EN	R/W	0h	Group1 Memory Overrun Interrupt Enable. A memory overrun occurs when the ADC tries to write a new conversion result to the Group1 results memory which is already full. For more details on the overrun interrupts, refer to Group Memory Overrun Interrupt subsection. Any operation mode: 0h = No interrupt is generated if a Group1 memory overrun occurs. 1h = A Group1 memory overrun interrupt is generated if a Group1 memory overrun condition occurs.
0	G1_THR_INT_EN	R/W	0h	Group1 Threshold Interrupt Enable. A Group1 threshold interrupt occurs when the programmed Group1 threshold counter counts down to 0. Refer to Group Memory Threshold Interrupt subsection for more details. Any operation mode: 0h = No interrupt is generated if the Group1 threshold counter reaches 0. 1h = A Group1 threshold interrupt is generated if the Group1 threshold counter reaches 0.

22.11.13 ADG2INTENA Register (Offset = 30h) [reset = 0h]

ADG2INTENA is shown in Figure 22-32 and described in Table 22-20.

Return to Summary Table.

Figure 22-32. ADG2INTENA Register

7	6	5	4	3	2	1	0
RESERVED				G2_END_ INT_EN	RESERVED	G2_OVR_ INT_EN	G2_THR_ INT_EN
R-0h				R/W-0h	R-0h	R/W-0h	R/W-0h

Table 22-20. ADG2INTENA Register Field Descriptions

Bit	Field	Type	Reset	Description
7-4	RESERVED	R	0h	Reads return 0, writes have no effect.
3	G2_END_INT_EN	R/W	0h	Group2 Conversion End Interrupt Enable. Refer to Group Conversion End Interrupt subsection for more details on the conversion end interrupts. Any operation mode: 0h = No interrupt is generated when conversion of all the channels selected for conversion in the Group2 is done. 1h = A Group2 conversion end interrupt is generated when conversion of all the channels selected for conversion in the Group2 is done.
2	RESERVED	R	0h	Reads return 0, writes have no effect.
1	G2_OVR_INT_EN	R/W	0h	Group2 Memory Overrun Interrupt Enable. A memory overrun occurs when the ADC tries to write a new conversion result to the Group2 results memory which is already full. For more details on the overrun interrupts, refer to Group Memory Overrun Interrupt subsection. Any operation mode: 0h = No interrupt is generated if a Group2 memory overrun occurs. 1h = A Group2 memory overrun interrupt is generated if a Group2 memory overrun condition occurs.
0	G2_THR_INT_EN	R/W	0h	Group2 Threshold Interrupt Enable. A Group2 threshold interrupt occurs when the programmed Group2 threshold counter counts down to 0. Refer to Group Memory Threshold Interrupt subsection for more details. Any operation mode: 0h = No interrupt is generated if the Group2 threshold counter reaches 0. 1h = A Group2 threshold interrupt is generated if the Group2 threshold counter reaches 0.

22.11.14 ADEVINTFLG Register (Offset = 34h) [reset = 4h]

ADEVINTFLG is shown in Figure 22-33 and described in Table 22-21.

Return to Summary Table.

Figure 22-33. ADEVINTFLG Register

7	6	5	4	3	2	1	0
RESERVED				EV_END	EV_MEM_ EMPTY	EV_MEM_ OVERRUN	EV_THR_ INT_FLG
R-0h				R/W1C-0h	R-1h	R-0h	R/W1C-0h

Table 22-21. ADEVINTFLG Register Field Descriptions

Bit	Field	Type	Reset	Description
7-4	RESERVED	R	0h	Reads return 0, writes have no effect.
3	EV_END	R/W1C	0h	Event Group Conversion End. This bit is set only if the Event Group conversions are configured to be in the single-conversion mode. This bit is cleared by any one of the following ways: By writing a 1 to this bit. By writing a 1 to the Event Group status register bit 0 (EV_END). By reading one conversion result from the Event Group results' memory in the read from FIFO mode. By writing a new set of channels to the Event Group channel select register. Any operation mode read: 0h = All the channels selected for conversion in the Event Group have not yet been converted. 1h = All the channels selected for conversion in the Event Group have been converted. An Event Group conversion end interrupt is generated, if enabled, when this bit gets set.
2	EV_MEM_EMPTY	R	1h	Event Group Results Memory Empty. This is a read-only bit, writes have no effect. It is not a source of an interrupt from the ADC module. Any operation mode read: 0h = The Event Group results memory is not empty. 1h = The Event Group results memory is empty.
1	EV_MEM_OVERRUN	R	0h	Event Group Memory Overrun. This is a read-only bit, writes have no effect. Any operation mode read: 0h = Event Group results memory has not overrun. 1h = Event Group results memory has overrun.
0	EV_THR_INT_FLG	R/W1C	0h	Event Group Threshold Interrupt Flag. This bit is cleared by writing a 1, writing a 0 has no effect. Any operation mode read: 0h = The number of conversions completed for the Event Group is smaller than the threshold programmed in the Event Group interrupt threshold register. 1h = The number of conversions completed for the Event Group is equal to or greater than the threshold programmed in the Event Group interrupt threshold register.

22.11.15 ADG1INTFLG Register (Offset = 38h) [reset = 4h]

ADG1INTFLG is shown in Figure 22-34 and described in Table 22-22.

Return to Summary Table.

Figure 22-34. ADG1INTFLG Register

7	6	5	4	3	2	1	0
RESERVED				G1_END	G1_MEM_ EMPTY	G1_MEM_ OVERRUN	G1_THR_ INT_FLG
R-0h				R/W1C-0h	R-1h	R-0h	R/W1C-0h

Table 22-22. ADG1INTFLG Register Field Descriptions

Bit	Field	Type	Reset	Description
7-4	RESERVED	R	0h	Reads return 0, writes have no effect.
3	G1_END	R/W1C	0h	Group1 Conversion End. This bit is set only if the Group1 conversions are configured to be in the single-conversion mode. This bit is cleared by any one of the following ways: By writing a 1 to this bit. By writing a 1 to the Group1 status register bit 0 (G1_END). By reading one conversion result from the Group1 results' memory in the read from FIFO mode. By writing a new set of channels to the Group1 channel select register. Any operation mode read: 0h = All the channels selected for conversion in the Group1 have not yet been converted. 1h = All the channels selected for conversion in the Group1 have been converted. A Group1 conversion end interrupt is generated, if enabled, when this bit gets set.
2	G1_MEM_EMPTY	R	1h	Group1 Results Memory Empty. This is a read-only bit, writes have no effect. It is not a source of an interrupt from the ADC module. Any operation mode read: 0h = The Group1 results memory is not empty. 1h = The Group1 results memory is empty.
1	G1_MEM_OVERRUN	R	0h	Group1 Memory Overrun. This is a read-only bit, writes have no effect. Any operation mode read: 0h = Group1 results memory has not overrun. 1h = Group1 results memory has overrun.
0	G1_THR_INT_FLG	R/W1C	0h	Group1 Threshold Interrupt Flag. This bit is cleared by writing a 1, writing a 0 has no effect. Any operation mode read: 0h = The number of conversions completed for the Group1 is smaller than the threshold programmed in the Group1 interrupt threshold register. 1h = The number of conversions completed for the Group1 is equal to or greater than the threshold programmed in the Group1 interrupt threshold register.

22.11.16 ADG2INTFLG Register (Offset = 3Ch) [reset = 4h]

ADG2INTFLG is shown in Figure 22-35 and described in Table 22-23.

Return to Summary Table.

Figure 22-35. ADG2INTFLG Register

7	6	5	4	3	2	1	0
RESERVED				G2_END	G2_MEM_ EMPTY	G2_MEM_ OVERRUN	G2_THR_ INT_FLG
R-0h				R/W1C-0h	R-1h	R-0h	R/W1C-0h

Table 22-23. ADG2INTFLG Register Field Descriptions

Bit	Field	Type	Reset	Description
7-4	RESERVED	R	0h	Reads return 0, writes have no effect.
3	G2_END	R/W1C	0h	Group2 Conversion End. This bit is set only if the Group2 conversions are configured to be in the single-conversion mode. This bit is cleared by any one of the following ways: By writing a 1 to this bit. By writing a 1 to the Group2 status register bit 0 (G2_END). By reading one conversion result from the Group2 results' memory in the read from FIFO mode. By writing a new set of channels to the Group2 channel select register. Any operation mode read: 0h = All the channels selected for conversion in the Group2 have not yet been converted. 1h = All the channels selected for conversion in the Group2 have been converted. A Group2 conversion end interrupt is generated, if enabled, when this bit gets set.
2	G2_MEM_EMPTY	R	1h	Group2 Results Memory Empty. This is a read-only bit, writes have no effect. It is not a source of an interrupt from the ADC module. Any operation mode read: 0h = The Group2 results memory is not empty. 1h = The Group2 results memory is empty.
1	G2_MEM_OVERRUN	R	0h	Group2 Memory Overrun. This is a read-only bit, writes have no effect. Any operation mode read: 0h = Group2 results memory has not overrun. 1h = Group2 results memory has overrun.
0	G2_THR_INT_FLG	R/W1C	0h	Group2 Threshold Interrupt Flag. This bit is cleared by writing a 1, writing a 0 has no effect. Any operation mode read: 0h = The number of conversions completed for the Group2 is smaller than the threshold programmed in the Group2 interrupt threshold register. 1h = The number of conversions completed for the Group2 is equal to or greater than the threshold programmed in the Group2 interrupt threshold register.

22.11.17 ADEVTHRINTCR Register (Offset = 40h) [reset = 0h]

ADEVTHRINTCR is shown in Figure 22-36 and described in Table 22-24.

Return to Summary Table.

Figure 22-36. ADEVTHRINTCR Register

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Sign_Extension							EV_THR								
R-0h							R/W-0h								

Table 22-24. ADEVTHRINTCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reads return 0, writes have no effect.
15-9	Sign_Extension	R	0h	These bits always read the same as the bit 8 of this register.
8-0	EV_THR	R/W	0h	Event Group Threshold Counter. Before ADC conversions begin on the Event Group, this field is initialized to the number of conversion results that the Event Group memory should contain before interrupting the CPU. This counter decrements when the ADC module writes a new conversion result to the Event Group results' memory. The counter increments for each read of a conversion result from the Event Group results' memory in the “read from FIFO” mode. The threshold counter is not affected for a direct read from the Event Group results' memory. Also, a simultaneous ADC write and a CPU and DMA read from the Event Group FIFO leaves the threshold counter unchanged. In case of an Event Group Results' memory overrun condition, if new conversion results are not allowed to overwrite the existing memory contents, then the Event Group threshold counter is not decremented. Refer to Group Memory Threshold Interrupt subsection for more details on the threshold interrupts.

22.11.18 ADG1THRINTCR Register (Offset = 44h) [reset = 0h]

ADG1THRINTCR is shown in Figure 22-37 and described in Table 22-25.

Return to Summary Table.

Figure 22-37. ADG1THRINTCR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0h															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Sign_Extension							G1_THR								
R-0h							R/W-0h								

Table 22-25. ADG1THRINTCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reads return 0, writes have no effect.
15-9	Sign_Extension	R	0h	These bits always read the same as the bit 8 of this register.
8-0	G1_THR	R/W	0h	Group1 Threshold Counter. Before ADC conversions begin on the Group1, this field is initialized to the number of conversion results that the Group1 memory should contain before interrupting the CPU. This counter decrements when the ADC module writes a new conversion result to the Group1 results' memory. The counter increments for each read of a conversion result from the Group1 results' memory in the “read from FIFO” mode. The threshold counter is not affected for a direct read from the group1 results' memory. Also, a simultaneous ADC write and a CPU and DMA read from the Group1 FIFO leaves the threshold counter unchanged. In case of an Group1 Results' memory overrun condition, if new conversion results are not allowed to overwrite the existing memory contents, then the Group1 threshold counter is not decremented. Refer to Group Memory Threshold Interrupt subsection for more details on the threshold interrupts.

22.11.19 ADG2THRINTCR Register (Offset = 48h) [reset = 0h]

ADG2THRINTCR is shown in Figure 22-38 and described in Table 22-26.

Return to Summary Table.

Figure 22-38. ADG2THRINTCR Register

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Sign_Extension							G2_THR								
R-0h							R/W-0h								

Table 22-26. ADG2THRINTCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reads return 0, writes have no effect.
15-9	Sign_Extension	R	0h	These bits always read the same as the bit 8 of this register.
8-0	G2_THR	R/W	0h	Group2 Threshold Counter. Before ADC conversions begin on the Group2, this field is initialized to the number of conversion results that the Group2 memory should contain before interrupting the CPU. This counter decrements when the ADC module writes a new conversion result to the Group2 results' memory. The counter increments for each read of a conversion result from the Group2 results' memory in the "read from FIFO" mode. The threshold counter is not affected for a direct read from the group2 results' memory. Also, a simultaneous ADC write and a CPU and DMA read from the Group2 FIFO leaves the threshold counter unchanged. In case of an Group2 Results' memory overrun condition, if new conversion results are not allowed to overwrite the existing memory contents, then the Group2 threshold counter is not decremented. Refer to Group Memory Threshold Interrupt subsection for more details on the threshold interrupts.

22.11.20 ADEVDMACR Register (Offset = 4Ch) [reset = 0h]

ADEVDMACR is shown in Figure 22-39 and described in Table 22-27.

Return to Summary Table.

Figure 22-39. ADEVDMACR Register

31	30	29	28	27	26	25	24
RESERVED							EV_BLOCKS
R-0h							R/W-0h
23	22	21	20	19	18	17	16
EV_BLOCKS							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				DMA_EV_END	EV_BLK_XFER	RESERVED	EV_DMA_EN
R-0h				R/W-0h	R/W-0h	R-0h	R/W-0h

Table 22-27. ADEVDMACR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reads return 0, writes have no effect.
24-16	EV_BLOCKS	R/W	0h	Number of Event Group Result buffers to be transferred using DMA if the ADC module is configured to generate a DMA request. If the Event Group is configured to use the block transfer mode of the DMA module, then the ADC module generates a DMA request after the Event Group results' memory accumulates EV_BLOCKS number of conversion results. This feature is designed to be used in place of the threshold interrupt for the Event Group. As a result, the EV_THR field of the Event Group Interrupt Threshold Control Register and the EV_BLOCKS field of the Event Group DMA Control Register are the same. Any operation mode: 0h = No DMA transfer occurs even if EV_BLK_XFER is set to 1. 1h = One DMA request is generated if the EV_BLK_XFER is set to 1 and the specified number of Event Group conversion results have been accumulated from 1h to 1FFh.
15-4	RESERVED	R	0h	Reads return 0, writes have no effect.
3	DMA_EV_END	R/W	0h	Event Group Conversion End DMA Transfer Enable. If DMA_EV_END bit is set to 1, EV_DMA_EN bit is ignored and DMA requests are generated every time the DMA_EV_END flag in the event group status register is set. The DMA_EV_END bit must be set before enabling conversions for the event group. Any operation mode read: 0h = ADC module generates a DMA request for each write to the Event group results RAM if EV_DMA_EN is set. 1h = ADC module generates a DMA request when the ADC has completed the conversions for all channels selected for conversion in the event group.

Table 22-27. ADEVDMACR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2	EV_BLK_XFER	R/W	0h	Event Group Block DMA Transfer Enable. If EV BLK XFER bit is set to 1, EV_DMA_EN bit is ignored and DMA requests are generated every time the Threshold Counter reaches 0 from a count value of 1. Any operation mode read: 0h = ADC module generates a DMA request for each write to the Event Group memory if EV_DMA_EN is set. 1h = ADC module generates a DMA request when the ADC has written EV_BLOCKS number of buffers into the Event Group memory.
1	RESERVED	R	0h	Reads return 0, writes have no effect.
0	EV_DMA_EN	R/W	0h	Event Group DMA Transfer Enable. Any operation mode read: 0h = ADC module does not generate a DMA request when it writes the conversion result to the Event Group memory. 1h = ADC module generates a DMA transfer when the ADC has written to the Event Group memory. The EV_BLK_XFER bit must be cleared to 0 for this DMA request to be generated.

22.11.21 ADG1DMACR Register (Offset = 50h) [reset = 0h]

ADG1DMACR is shown in Figure 22-40 and described in Table 22-28.

Return to Summary Table.

Figure 22-40. ADG1DMACR Register

31	30	29	28	27	26	25	24
RESERVED							G1_BLOCKS
R-0h							R/W-0h
23	22	21	20	19	18	17	16
G1_BLOCKS							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				DMA_G1_END	G1_BLK_XFER	RESERVED	G1_DMA_EN
R-0h				R/W-0h	R/W-0h	R-0h	R/W-0h

Table 22-28. ADG1DMACR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reads return 0, writes have no effect.
24-16	G1_BLOCKS	R/W	0h	Number of Group1 Result buffers to be transferred using DMA if the ADC module is configured to generate a DMA request. If the Group1 is configured to use the block transfer mode of the DMA module, then the ADC module generates a DMA request after the Group1 results' memory accumulates G1_BLOCKS number of conversion results. This feature is designed to be used in place of the threshold interrupt for the Group1. As a result, the G1_THR field of the Group1 Interrupt Threshold Control Register and the G1_BLOCKS field of the Group1 DMA Control Register are the same. Any operation mode: 0h = No DMA transfer occurs even if G1_BLK_XFER is set to 1. 1h = One DMA request is generated if the G1_BLK_XFER is set to 1 and the specified number of Group1 conversion results have been accumulated from 1h to 1FFh.
15-4	RESERVED	R	0h	Reads return 0, writes have no effect.
3	DMA_G1_END	R/W	0h	Group1 Conversion End DMA Transfer Enable. If DMA_G1_END bit is set to 1, G1_DMA_EN bit is ignored and DMA requests are generated every time the DMA_G1_END flag in the group 1 status register is set. The DMA_G1_END bit must be set before enabling conversions for the group 1. Any operation mode read: 0h = ADC module generates a DMA request for each write to the group1 results RAM if G1_DMA_EN is set. 1h = ADC module generates a DMA request when the ADC has completed the conversions for all channels selected for conversion in the group1.
2	G1_BLK_XFER	R/W	0h	Group1 Block DMA Transfer Enable. If G1 BLK_XFER bit is set to 1, G1_DMA_EN bit is ignored and DMA requests are generated every time the Threshold Counter reaches 0 from a count value of 1. Any operation mode read: 0h = ADC module generates a DMA request for each write to the Group1 memory if G1_DMA_EN is set. 1h = ADC module generates a DMA request when the ADC has written G1_BLOCKS number of buffers into the Group1 memory.

Table 22-28. ADG1DMACR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	RESERVED	R	0h	Reads return 0, writes have no effect.
0	G1_DMA_EN	R/W	0h	Group1 DMA Transfer Enable. Any operation mode read: 0h = ADC module does not generate a DMA request when it writes the conversion result to the Group1 memory. 1h = ADC module generates a DMA transfer when the ADC has written to the Group1 memory. The G1_BLK_XFER bit must be cleared to 0 for this DMA request to be generated.

22.11.22 ADG2DMACR Register (Offset = 54h) [reset = 0h]

ADG2DMACR is shown in Figure 22-41 and described in Table 22-29.

Return to Summary Table.

Figure 22-41. ADG2DMACR Register

31	30	29	28	27	26	25	24
RESERVED							G2_BLOCKS
R-0h							R/W-0h
23	22	21	20	19	18	17	16
G2_BLOCKS							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				DMA_G2_END	G2_BLK_XFER	RESERVED	G2_DMA_EN
R-0h				R/W-0h	R/W-0h	R-0h	R/W-0h

Table 22-29. ADG2DMACR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reads return 0, writes have no effect.
24-16	G2_BLOCKS	R/W	0h	Number of Group2 Result buffers to be transferred using DMA if the ADC module is configured to generate a DMA request. If the Group2 is configured to use the block transfer mode of the DMA module, then the ADC module generates a DMA request after the Group2 results' memory accumulates G2_BLOCKS number of conversion results. This feature is designed to be used in place of the threshold interrupt for the Group2. As a result, the G2_THR field of the Group2 Interrupt Threshold Control Register and the G2_BLOCKS field of the Group2 DMA Control Register are the same. Any operation mode: 0h = No DMA transfer occurs even if G2_BLK_XFER is set to 1. 1h = One DMA request is generated if the G2_BLK_XFER is set to 1 and the specified number of Group2 conversion results have been accumulated from 1h to 1FFh.
15-4	RESERVED	R	0h	Reads return 0, writes have no effect.
3	DMA_G2_END	R/W	0h	Group2 Conversion End DMA Transfer Enable. If DMA_G2_END bit is set to 1, G2_DMA_EN bit is ignored and DMA requests are generated every time the DMA_G2_END flag in the group 2 status register is set. The DMA_G2_END bit must be set before enabling conversions for the group 2. Any operation mode read: 0h = ADC module generates a DMA request for each write to the group2 results RAM if G2_DMA_EN is set. 1h = ADC module generates a DMA request when the ADC has completed the conversions for all channels selected for conversion in the group2.
2	G2_BLK_XFER	R/W	0h	Group2 Block DMA Transfer Enable. If G2_BLK_XFER bit is set to 1, G2_DMA_EN bit is ignored and DMA requests are generated every time the Threshold Counter reaches 0 from a count value of 1. Any operation mode read: 0h = ADC module generates a DMA request for each write to the Group2 memory if G2_DMA_EN is set. 1h = ADC module generates a DMA request when the ADC has written G2_BLOCKS number of buffers into the Group2 memory.

Table 22-29. ADG2DMACR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1	RESERVED	R	0h	Reads return 0, writes have no effect.
0	G2_DMA_EN	R/W	0h	Group2 DMA Transfer Enable. Any operation mode read: 0h = ADC module does not generate a DMA request when it writes the conversion result to the Group2 memory. 1h = ADC module generates a DMA transfer when the ADC has written to the Group2 memory. The G2_BLK_XFER bit must be cleared to 0 for this DMA request to be generated.

22.11.23 ADBNDCR Register (Offset = 58h) [reset = 0h]

ADBNDCR is shown in Figure 22-42 and described in Table 22-30.

Return to Summary Table.

Refer to *How Results are Stored in the Results' Memory* subsection for further details on how the conversion results are stored in the ADC results' RAM.

Figure 22-42. ADBNDCR Register

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

Table 22-30. ADBNDCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	R	0h	Reads return 0, writes have no effect.
24-16	BNDA	R/W	0h	Buffer Boundary A. These bits determine the memory available for the Event Group conversion results. The memory available is specified in terms of pairs of result buffers. A total of (2 x BNDA) buffers are available in the ADC results memory for storing Event Group conversion results from 0 to 1FFh. Any operation mode: 0h = Event Group conversions are not required. If Event Group conversions are performed with the BNDA value of 0, then the Event Group memory size defaults to 1024 words. For proper usage of the ADC results memory, configure the BNDA value to be non-zero and lower than the BNDB value.
15-9	RESERVED	R	0h	Reads return 0, writes have no effect.
8-0	BNDB	R/W	0h	Buffer Boundary B. These bits specify the number of buffers allocated for the Event Group plus the number of buffers allocated for the Group1. The number of buffer pairs allocated for storing Group1 conversion results is determined by subtracting BNDA from BNDB. As a result, BNDB must always be specified as greater than or equal to BNDA. A total of 2 x (BNDB - BNDA) buffers are available in the ADC results memory for storing Group1 conversion results from 0 to 1FFh. Any operation mode: 0h = Event Group as well as Group1 conversions are not required.

22.11.24 ADBNDEND Register (Offset = 5Ch) [reset = 0h]

ADBNDEND is shown in Figure 22-43 and described in Table 22-31.

Return to Summary Table.

Figure 22-43. ADBNDEND Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="7">RESERVED</td><td>BUF_INIT_ ACTIVE</td></tr>
<tr><td colspan="7">R-0h</td><td>R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="5">RESERVED</td><td colspan="3">BNDEND</td></tr>
<tr><td colspan="5">R-0h</td><td colspan="3">R/W-0h</td></tr>
</table>

Table 22-31. ADBNDEND Register Field Descriptions

Bit	Field	Type	Reset	Description
31-17	RESERVED	R	0h	Reads return 0, writes have no effect.
16	BUF_INIT_ACTIVE	R	0h	ADC Results Memory Auto-initialization Status. Any operation mode: 0h = ADC Results Memory is currently not being initialized, and the ADC is available. If this bit is read as 0 after triggering an auto-initialization of the ADC results memory, then the ADC results memory has been completely initialized to 0. For devices requiring parity checking on the ADC results memory, the parity bit in the results memory is also initialized according to the parity polarity. The parity polarity as well as the auto-initialization process is controlled by the System module. Refer to the Architecture chapter for more details. 1h = ADC results memory is being initialized, and the ADC is not available for conversion.
15-3	RESERVED	R	0h	Reads return 0, writes have no effect.
2-0	BNDEND	R/W	0h	Buffer Boundary End. These bits specify the total number of memory buffers available for storing the ADC conversion results. These bits should be programmed to match the number of ADC conversion result buffers required to be used for the application. Any operation mode: 0h = 16 words available for storing ADC conversion results. 1h = 32 words available for storing ADC conversion results. 2h = 64 words available for storing ADC conversion results. This is the maximum configuration allowed since the device supports 64 buffers each for ADC1 as well as ADC2. 3h = Reserved 4h = Reserved 5h = Reserved 6h = Reserved 7h = Reserved

22.11.25 ADEVSAMP Register (Offset = 60h) [reset = 0h]

ADEVSAMP is shown in Figure 22-44 and described in Table 22-32.

Return to Summary Table.

Figure 22-44. ADEVSAMP Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="20">RESERVED</td><td colspan="12">EV_ACQ</td></tr>
<tr><td colspan="20">R-0h</td><td colspan="12">R/W-0h</td></tr>
</table>

Table 22-32. ADEVSAMP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0h	Reads return 0, writes have no effect.
11-0	EV_ACQ	R/W	0h	Event Group Acquisition Time. These bits define the sampling window (SW) for the Event Group conversions. SW = EV_ACQ + 2 in terms of ADCLK cycles. There are two factors that determine the minimum sampling window value required: First, the ADC module design requires that software is greater than or equal to 3 ADCLK cycles. Second, the ADC input impedance necessitates a certain minimum sampling time. This needs to be assured by configuring the EV_ACQ value properly considering the frequency of the ADCLK signal. Refer to the device datasheet to determine the minimum sampling time for this device.

22.11.26 ADG1SAMP Register (Offset = 64h) [reset = 0h]

ADG1SAMP is shown in Figure 22-45 and described in Table 22-33.

Return to Summary Table.

Figure 22-45. ADG1SAMP Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																				G1_ACQ											
R-0h																				R/W-0h											

Table 22-33. ADG1SAMP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0h	Reads return 0, writes have no effect.
11-0	G1_ACQ	R/W	0h	Group1 Acquisition Time. These bits define the sampling window (SW) for the Group1 conversions. SW = G1_ACQ + 2 in terms of ADCLK cycles. There are two factors that determine the minimum sampling window value required: First, the ADC module design requires that software is greater than or equal to 3 ADCLK cycles. Second, the ADC input impedance necessitates a certain minimum sampling time. This needs to be assured by configuring the G1_ACQ value properly considering the frequency of the ADCLK signal. Refer to the device datasheet to determine the minimum sampling time for this device.

22.11.27 ADG2SAMP Register (Offset = 68h) [reset = 0h]

ADG2SAMP is shown in Figure 22-46 and described in Table 22-34.

Return to Summary Table.

Figure 22-46. ADG2SAMP Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																				G2_ACQ											
R-0h																				R/W-0h											

Table 22-34. ADG2SAMP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0h	Reads return 0, writes have no effect.
11-0	G2_ACQ	R/W	0h	Group2 Acquisition Time. These bits define the sampling window (SW) for the Group2 conversions. SW = G2_ACQ + 2 in terms of ADCLK cycles. There are two factors that determine the minimum sampling window value required: First, the ADC module design requires that software is greater than or equal to 3 ADCLK cycles. Second, the ADC input impedance necessitates a certain minimum sampling time. This needs to be assured by configuring the G2_ACQ value properly considering the frequency of the ADCLK signal. Refer to the device datasheet to determine the minimum sampling time for this device.

22.11.28 ADEVSR Register (Offset = 6Ch) [reset = 8h]

ADEVSR is shown in Figure 22-47 and described in Table 22-35.

Return to Summary Table.

Figure 22-47. ADEVSR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RESERVED							
R-0h							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RESERVED							
R-0h							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RESERVED				EV_MEM_EMPTY	EV_BUSY	EV_STOP	EV_END
R-0h				R-1h	R-0h	R-0h	R/W-0h

Table 22-35. ADEVSR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reads return 0, writes have no effect.
3	EV_MEM_EMPTY	R	1h	Event Group Results Memory Empty. This bit is effectively used only when the conversion results are read out of the Event Group results memory in the "read from FIFO" mode. Any operation mode read: 0h = The Event Group results memory has valid conversion results. 1h = The Event Group results memory is empty, or does not contain any unread conversion results.
2	EV_BUSY	R	0h	Event Group Conversion Busy. Any operation mode read: 0h = Event Group conversions are neither in progress nor frozen. 1h = Event Group conversions are either in progress, or are frozen for servicing some other group. This bit is always set when the Event Group is configured to be in the continuous conversion mode.
1	EV_STOP	R	0h	Event Group Conversion Stopped. Any operation mode read: 0h = Event Group conversions are not currently frozen. 1h = Event Group conversions are currently frozen.
0	EV_END	R/W	0h	Event Group Conversions Ended. This bit is cleared under the following conditions: By reading a conversion result from the Event Group results memory in the "read from FIFO" mode. By writing a new value to the Event Group channel select register ADEVSEL. By writing a 1 to this bit. By disabling the ADC module by clearing the ADC_EN bit in the ADC operating mode control register (ADOPMODECR). Any operation mode read: 0h = Event Group conversions have either not been started or have not yet completed since the last time this status bit was cleared. 1h = The conversion for all the channels selected in the Event Group has completed.

22.11.29 ADG1SR Register (Offset = 70h) [reset = 8h]

ADG1SR is shown in Figure 22-48 and described in Table 22-36.

Return to Summary Table.

Figure 22-48. ADG1SR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				G1_MEM_EMPTY	G1_BUSY	G1_STOP	G1_END
R-0h				R-1h	R-0h	R-0h	R/W-0h

Table 22-36. ADG1SR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reads return 0, writes have no effect.
3	G1_MEM_EMPTY	R	1h	Group1 Results Memory Empty. This bit is effectively used only when the conversion results are read out of the Group1 results memory in the "read from FIFO" mode. Any operation mode read: 0h = The Group1 results memory has valid conversion results. 1h = The Group1 results memory is empty, or does not contain any unread conversion results.
2	G1_BUSY	R	0h	Group1 Conversion Busy. Any operation mode read: 0h = Group1 conversions are neither in progress nor frozen. 1h = Group1 conversions are either in progress, or are frozen for servicing some other group. This bit is always set when the Group1 is configured to be in the continuous conversion mode.
1	G1_STOP	R	0h	Group1 Conversion Stopped. Any operation mode read: 0h = Group1 conversions are not currently frozen. 1h = Group1 conversions are currently frozen.
0	G1_END	R/W	0h	Group1 Conversions Ended. This bit is cleared under the following conditions: By reading a conversion result from the Group1 results memory in the "read from FIFO" mode. By writing a new value to the Group1 channel select register ADG1SEL. By writing a 1 to this bit. By disabling the ADC module by clearing the ADC_EN bit in the ADC operating mode control register (ADOPMODECR. Any operation mode read: 0h = Group1 conversions have either not been started or have not yet completed since the last time this status bit was cleared. 1h = The conversion for all the channels selected in the Group1 has completed.

22.11.30 ADG2SR Register (Offset = 74h) [reset = 8h]

ADG2SR is shown in Figure 22-49 and described in Table 22-37.

Return to Summary Table.

Figure 22-49. ADG2SR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				G2_MEM_ EMPTY	G2_BUSY	G2_STOP	G2_END
R-0h				R-1h	R-0h	R-0h	R/W-0h

Table 22-37. ADG2SR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reads return 0, writes have no effect.
3	G2_MEM_EMPTY	R	1h	Group2 Results Memory Empty. This bit is effectively used only when the conversion results are read out of the Group2 results memory in the "read from FIFO" mode. Any operation mode read: 0h = The Group2 results memory has valid conversion results. 1h = The Group2 results memory is empty, or does not contain any unread conversion results.
2	G2_BUSY	R	0h	Group2 Conversion Busy. Any operation mode read: 0h = Group2 conversions are neither in progress nor frozen. 1h = Group2 conversions are either in progress, or are frozen for servicing some other group. This bit is always set when the Group2 is configured to be in the continuous conversion mode.
1	G2_STOP	R	0h	Group2 Conversion Stopped. Any operation mode read: 0h = Group2 conversions are not currently frozen. 1h = Group2 conversions are currently frozen.
0	G2_END	R/W	0h	Group2 Conversions Ended. This bit is cleared under the following conditions: By reading a conversion result from the Group2 results memory in the "read from FIFO" mode. By writing a new value to the Group2 channel select register ADG2SEL. By writing a 1 to this bit. By disabling the ADC module by clearing the ADC_EN bit in the ADC operating mode control register (ADOPMODECR). Any operation mode read: 0h = Group2 conversions have either not been started or have not yet completed since the last time this status bit was cleared. 1h = The conversion for all the channels selected in the Group2 has completed.

22.11.31 ADEVSEL Register (Offset = 78h) [reset = 0h]

ADEVSEL is shown in Figure 22-50 and described in Table 22-38.

Return to Summary Table.

Clearing ADEVSEL During a Conversion: Writing 0000h to ADEVSEL stops the Event Group conversions. This does not cause the ADC Event Group results Memory pointer or the Event Group Threshold Register to be reset.

Writing A Non-Zero Value To ADEVSEL During a Conversion: Writing a new value to ADEVSEL while a Channel in Event Group is being converted results in a new conversion sequence starting immediately with the highest priority channel in the new ADEVSEL selection. This also causes the ADC Event Group Results Memory pointer to be reset so that the memory allocated for storing the Event Group conversion results gets overwritten. Care should be taken to re-program the corresponding Interrupt Threshold Counter or DMA Threshold Counter again so that correct number of conversions happen before a Threshold interrupt or Block DMA request is generated.

ADC1 supports up to 24 channels and ADC2 supports up to 16 channels on the microcontroller.

Figure 22-50. ADEVSEL Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">RESERVED</td><td colspan="24">EV_SEL</td></tr>
<tr><td colspan="8">R-0h</td><td colspan="24">R/W-0h</td></tr>
</table>

Table 22-38. ADEVSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reads return 0, writes have no effect.
23-0	EV_SEL	R/W	0h	Event Group channels selected. Any non-zero value, the channels marked by the bit positions that are set to 1 are converted in ascending order when the Event Group is triggered. Any operation mode: 0h = No ADC input channel is selected for conversion in the Event Group.

22.11.32 ADG1SEL Register (Offset = 7Ch) [reset = 0h]

ADG1SEL is shown in Figure 22-51 and described in Table 22-39.

Return to Summary Table.

Clearing ADG1SEL During a Conversion: Writing 0000h to ADG1SEL stops the Group1 conversions. This does not cause the ADC Group1 Results Memory pointer or the Group1 Threshold Register to be reset.

Writing A Non-Zero Value To ADG1SEL During a Conversion: Writing a new value to ADG1SEL while a Channel in Group1 is being converted results in a new conversion sequence starting immediately with the highest priority channel in the new ADG1SEL selection. This also causes the ADC Group1 Results Memory pointer to be reset so that the memory allocated for storing the Group1 conversion results gets overwritten. Care should be taken to re-program the corresponding Interrupt Threshold Counter or DMA Threshold Counter again so that correct number of conversions happen before a Threshold interrupt or Block DMA request is generated.

ADC1 supports up to 24 channels and ADC2 supports up to 16 channels on the microcontroller.

Figure 22-51. ADG1SEL Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								G1_SEL																							
R-0h								R/W-0h																							

Table 22-39. ADG1SEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reads return 0, writes have no effect.
23-0	G1_SEL	R/W	0h	Group1 channels selected. Any non-zero value, the channels marked by the bit positions that are set to 1 are converted in ascending order when the Group1 is triggered. Any operation mode: 0h = No ADC input channel is selected for conversion in the Group1.

22.11.33 ADG2SEL Register (Offset = 80h) [reset = 0h]

ADG2SEL is shown in Figure 22-52 and described in Table 22-40.

Return to Summary Table.

Clearing ADG2SEL During a Conversion: Writing 0000h to ADG2SEL stops the Group2 conversions. This does not cause the ADC Group2 Results Memory pointer or the Group2 Threshold Register to be reset.

Writing A Non-Zero Value To ADG2SEL During a Conversion: Writing a new value to ADG2SEL while a Channel in Group2 is being converted results in a new conversion sequence starting immediately with the highest priority channel in the new ADG2SEL selection. This also causes the ADC Group2 Results Memory pointer to be reset so that the memory allocated for storing the Group2 conversion results gets overwritten. Care should be taken to re-program the corresponding Interrupt Threshold Counter or DMA Threshold Counter again so that correct number of conversions happen before a Threshold interrupt or Block DMA request is generated.

ADC1 supports up to 24 channels and ADC2 supports up to 16 channels on the microcontroller.

Figure 22-52. ADG2SEL Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">RESERVED</td><td colspan="24">G2_SEL</td></tr>
<tr><td colspan="8">R-0h</td><td colspan="24">R/W-0h</td></tr>
</table>

Table 22-40. ADG2SEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reads return 0, writes have no effect.
23-0	G2_SEL	R/W	0h	Group2 channels selected. Any non-zero value, the channels marked by the bit positions that are set to 1 are converted in ascending order when the Group2 is triggered. Any operation mode: 0h = No ADC input channel is selected for conversion in the Group2.

22.11.34 ADCALR Register (Offset = 84h) [reset = 0h]

ADCALR is shown in Figure 22-53 and described in Table 22-41.

Return to Summary Table.

The format of the ADCALR is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module.

Figure 22-53. ADCALR Register

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

Table 22-41. ADCALR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0h	Reads return 0, writes have no effect.
11-0	ADCALR	R/W	0h	ADC Calibration Result and Offset Error Correction Value. **Note: The actual size of the ADCALR field is 12 bits or 10 bits depending on whether the ADC is configured to be in 12-bit or 10-bit resolution mode, respectively. Bits 11-10 are reserved when the module is configured as a 10-bit ADC module.** The ADC module writes the results of the calibration conversions to this register. The application is required to use these conversion results and determine the ADC offset error. The application then computes the correction for the offset error and this correction value needs to be written back to the ADCALR register in the 2's complement form. During normal conversion (when calibration is disabled), the ADCALR register contents are automatically added to each digital output from the ADC core before it is stored in the ADC results memory. For more details on error calibration, refer to ADC Error Calibration Mode subsection. 0h = ADC Calibration Result and Offset Error Correction Value.

22.11.35 ADSMSTATE Register (Offset = 88h) [reset = 0h]

ADSMSTATE is shown in Figure 22-54 and described in Table 22-42.

Return to Summary Table.

Figure 22-54. ADSMSTATE Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">RESERVED</td></tr>
<tr><td colspan="16">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="12">RESERVED</td><td colspan="4">SM_STATE</td></tr>
<tr><td colspan="12">R-0h</td><td colspan="4">R-0h</td></tr>
</table>

Table 22-42. ADSMSTATE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reads return 0, writes have no effect.
3-0	SM_STATE	R	0h	ADC State Machine Current State. These bits reflect the current state of the state machine and are reserved for use by TI for debug purposes.

22.11.36 ADLASTCONV Register (Offset = 8Ch) [reset = X]

ADLASTCONV is shown in Figure 22-55 and described in Table 22-43.

Return to Summary Table.

Figure 22-55. ADLASTCONV Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">RESERVED</td><td colspan="24">LAST_CONV</td></tr>
<tr><td colspan="8">R-0h</td><td colspan="24">R-X</td></tr>
</table>

Table 22-43. ADLASTCONV Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	RESERVED	R	0h	Reads return 0, writes have no effect.
23-0	LAST_CONV	R	X	ADC Input Channel's Last Converted Value. This register indicates whether the last converted value for a particular input channel was lower or higher than the mid-point of the reference voltage. In other words, this register acts as a digital input register and is read by the application to determine the digital level at the input pins. This data is only valid for an input channel if it has been converted at least once. Any operation mode read for each bit of this register: 0h = A level lower than the midpoint reference voltage was measured at the last conversion for this channel. 1h = A level higher than or equal to the midpoint reference voltage was measured at the last conversion for this channel.

22.11.37 ADEVBUFFER_0 to ADEVBUFFER_7 Register (Offset = 90h to ACh) [reset = 80000000h]

ADEVBUFFER_0 to ADEVBUFFER_7 is shown in Figure 22-56 and Figure 22-57, and described in Table 22-44.

Return to Summary Table.

There are 8 ADC Event Group Results' FIFO Registers (ADEVBUFFER) at offsets 90h to ACh. The format of the data read from the ADEVBUFFER locations is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module.

Figure 22-56. ADEVBUFFER_0 to ADEVBUFFER_7 Register (12-bit)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td></tr>
<tr><td>EV_EMPTY</td><td colspan="7">RESERVED</td></tr>
<tr><td>R-1h</td><td colspan="7">R-0h</td></tr>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="3">RESERVED</td><td colspan="5">EV_CHID</td></tr>
<tr><td colspan="3">R-0h</td><td colspan="5">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="4">RESERVED</td><td colspan="4">EV_DR</td></tr>
<tr><td colspan="4">R-0h</td><td colspan="4">R-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">EV_DR</td></tr>
<tr><td colspan="8">R-0h</td></tr>
</table>

Figure 22-57. ADEVBUFFER_0 to ADEVBUFFER_7 Register (10-bit)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td>EV_EMPTY</td><td colspan="5">EV_CHID</td><td colspan="2">EV_DR</td></tr>
<tr><td>R-1h</td><td colspan="5">R-0h</td><td colspan="2">R-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">EV_DR</td></tr>
<tr><td colspan="8">R-0h</td></tr>
</table>

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 22-44. ADEVBUFFER_0 to ADEVBUFFER_7 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	EV_EMPTY	R	1h	Event Group FIFO Empty. This bit is applicable only when the "read from FIFO" mode is used for reading the Event Group conversion results. Any operation mode read: 0h = The data in the EV_DR field of this buffer is valid. 1h = The data in the EV_DR field of this buffer is not valid and there are no valid data in the Event Group results memory.
30-21	RESERVED	R	0h	Reads return 0, writes have no effect.
20-16	EV_CHID	R	0h	Event Group Channel Id. These bits are also applicable only when the "read from FIFO" mode is used for reading the Event Group conversion results. The conversion result in the EV_DR field of this buffer is from the ADC input channel number denoted by the EV_CHID field from 1h to 1Fh. Any operation mode read: 0h = The conversion result in the EV_DR field of this buffer is from the ADC input channel 0, or the channel id mode is disabled in the Event Group mode control register (ADEVMODECR).
15-12	RESERVED	R	0h	Reads return 0, writes have no effect.
11-0	EV_DR	R	0h	Event Group Digital Conversion Result. The Event Group results' FIFO location is aliased eight times, so that any word-aligned read from the address range 90h to AFh results in one conversion result to be read from the Event Group results' memory. This allows the ARM LDMIA instruction to read out up to 8 conversion results from the Event Group results' memory with just one instruction.

22.11.38 ADG1BUFFER_0 to ADG1BUFFER_7 Register (Offset = B0h to CCh) [reset = X]

ADG1BUFFER_0 to ADG1BUFFER_7 is shown in Figure 22-58 and Figure 22-59, and described in Table 22-45.

Return to Summary Table.

There are 8 ADC Group1 Results FIFO Registers (ADG1BUFFER) at offsets B0h to CCh. The format of the data read from the ADG1BUFFER locations is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module.

The reset value of ADG1BUFFER comes from RAM and is random after reset.

Figure 22-58. ADG1BUFFER_0 to ADG1BUFFER_7 Register (12-bit)

31	30	29	28	27	26	25	24
G1_EMPTY	RESERVED						
R-1h	R-0h						
23	22	21	20	19	18	17	16
RESERVED			G1_CHID				
R-0h			R-X				
15	14	13	12	11	10	9	8
RESERVED				G1_DR			
R-0h				R-X			
7	6	5	4	3	2	1	0
G1_DR							
R-X							

Figure 22-59. ADG1BUFFER_0 to ADG1BUFFER_7 Register (10-bit)

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
G1_EMPTY	G1_CHID					G1_DR	
R-1h	R-X					R-X	
7	6	5	4	3	2	1	0
G1_DR							
R-X							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 22-45. ADG1BUFFER_0 to ADG1BUFFER_7 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	G1_EMPTY	R	1h	Group1 FIFO Empty. This bit is applicable only when the "read from FIFO" mode is used for reading the Group1 conversion results. Any operation mode read: 0h = The data in the G1_DR field of this buffer is valid. 1h = The data in the G1_DR field of this buffer is not valid and there are no valid data in the Group1 results memory.
30-21	RESERVED	R	0h	Reads return 0, writes have no effect.
20-16	G1_CHID	R	X	Group1 Channel Id. These bits are also applicable only when the "read from FIFO" mode is used for reading the Group1 conversion results. The conversion result in the G1_DR field of this buffer is from the ADC input channel number denoted by the G1_CHID field from 1h to 1Fh. Any operation mode read: 0h = The conversion result in the G1_DR field of this buffer is from the ADC input channel 0, or the channel id mode is disabled in the Group1 mode control register (ADG1MODECR).
15-12	RESERVED	R	0h	Reads return 0, writes have no effect.
11-0	G1_DR	R	X	Group1 Digital Conversion Result. The Group1 results' FIFO location is aliased eight times, so that any word-aligned read from the address range B0h to CFh results in one conversion result to be read from the Group1 results' memory. This allows the ARM LDMIA instruction to read out up to 8 conversion results from the Group1 results' memory with just one instruction.

22.11.39 ADG2BUFFER_0 to ADG2BUFFER_7 Register (Offset = D0h to ECh) [reset = X]

ADG2BUFFER_0 to ADG2BUFFER_7 is shown in Figure 22-60 and Figure 22-61, and described in Table 22-46.

Return to Summary Table.

There are 8 ADC Group2 Results FIFO Registers (ADG2BUFFER) at offsets D0 to ECh. The format of the data read from the ADG2BUFFER locations is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module.

The reset value of ADG2BUFFER comes from RAM and is random after reset.

Figure 22-60. ADG2BUFFER_0 to ADG2BUFFER_7 Register (12-bit)

31	30	29	28	27	26	25	24
G2_EMPTY	RESERVED						
R-1h	R-0h						
23	22	21	20	19	18	17	16
RESERVED			G2_CHID				
R-0h			R-X				
15	14	13	12	11	10	9	8
RESERVED				G2_DR			
R-0h				R-X			
7	6	5	4	3	2	1	0
G2_DR							
R-X							

Figure 22-61. ADG2BUFFER_0 to ADG2BUFFER_7 Register (10-bit)

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
G2_EMPTY	G2_CHID					G2_DR	
R-1h	R-X					R-X	
7	6	5	4	3	2	1	0
G2_DR							
R-X							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 22-46. ADG2BUFFER_0 to ADG2BUFFER_7 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	G2_EMPTY	R	1h	Group2 FIFO Empty. This bit is applicable only when the "read from FIFO" mode is used for reading the Group2 conversion results. Any operation mode read: 0h = The data in the G2_DR field of this buffer is valid. 1h = The data in the G2_DR field of this buffer is not valid and there are no valid data in the Group2 results memory.
30-21	RESERVED	R	0h	Reads return 0, writes have no effect.
20-16	G2_CHID	R	X	Group2 Channel Id. These bits are also applicable only when the "read from FIFO" mode is used for reading the Group2 conversion results. The conversion result in the G2_DR field of this buffer is from the ADC input channel number denoted by the G2_CHID field from 1h to 1Fh. Any operation mode read: 0h = The conversion result in the G2_DR field of this buffer is from the ADC input channel 0, or the channel id mode is disabled in the Group2 mode control register (ADG2MODECR).
15-12	RESERVED	R	0h	Reads return 0, writes have no effect.
11-0	G2_DR	R	X	Group2 Digital Conversion Result. The Group2 results' FIFO location is aliased eight times, so that any word-aligned read from the address range D0h to EFh results in one conversion result to be read from the Group2 results' memory. This allows the ARM LDMIA instruction to read out up to 8 conversion results from the Group2 results' memory with just one instruction.

22.11.40 ADEVEMUBUFFER Register (Offset = F0h) [reset = X]

ADEVEMUBUFFER is shown in Figure 22-62 and Figure 22-63, and described in Table 22-47.

Return to Summary Table.

The format of the data read from the ADEVEMUBUFFER locations is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module.

A read from this location also gives out one conversion result from the Event Group results' memory along with the EV_EMPTY status bit and the optional channel id. However, this read does not affect any of the status flags in the Event Group interrupt flag register or the Event Group status register. This register is useful for debuggers.

The reset value of ADEVEMUBUFFER comes from RAM and is random after reset.

Figure 22-62. ADEVEMUBUFFER Register (12-bit)

31	30	29	28	27	26	25	24
EV_EMPTY	RESERVED						
R-1h	R-0h						
23	22	21	20	19	18	17	16
RESERVED			EV_CHID				
R-0h			R-X				
15	14	13	12	11	10	9	8
RESERVED				EV_DR			
R-0h				R-X			
7	6	5	4	3	2	1	0
EV_DR							
R-X							

Figure 22-63. ADEVEMUBUFFER Register (10-bit)

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
EV_EMPTY	EV_CHID					EV_DR	
R-1h	R-X					R-X	
7	6	5	4	3	2	1	0
EV_DR							
R-X							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 22-47. ADEVEMUBUFFER Register Field Descriptions

Bit	Field	Type	Reset	Description
31	EV_EMPTY	R	1h	Event Group FIFO Empty. This bit is applicable only when the "read from FIFO" mode is used for reading the Event Group conversion results. Any operation mode read: 0h = The data in the EV_DR field of this buffer is valid. 1h = The data in the EV_DR field of this buffer is not valid and there are no valid data in the Event Group results memory.
30-21	RESERVED	R	0h	Reads return 0, writes have no effect.
20-16	EV_CHID	R	X	Event Group Channel Id. These bits are also applicable only when the "read from FIFO" mode is used for reading the Event Group conversion results. The conversion result in the EV_DR field of this buffer is from the ADC input channel number denoted by the EV_CHID field from 1h to 1Fh. Any operation mode read: 0h = The conversion result in the EV_DR field of this buffer is from the ADC input channel 0, or the channel id mode is disabled in the Event Group mode control register (ADEVMODECR).
15-12	RESERVED	R	0h	Reads return 0, writes have no effect.
11-0	EV_DR	R	X	Event Group Digital Conversion Result. The Event Group results' FIFO location is aliased eight times, so that any word-aligned read from the address range 90h to AFh results in one conversion result to be read from the Event Group results' memory. This allows the ARM LDMIA instruction to read out up to 8 conversion results from the Event Group results' memory with just one instruction.

22.11.41 ADG1EMUBUFFER Register (Offset = F4h) [reset = X]

ADG1EMUBUFFER is shown in Figure 22-64 and Figure 22-65, and described in Table 22-48.

Return to Summary Table.

The format of the data read from the ADG1EMUBUFFER locations is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module.

A read from this location also gives out one conversion result from the Group1 results' memory along with the G1_EMPTY status bit and the optional channel id. However, this read does not affect any of the status flags in the Group1 interrupt flag register or the Group1 status register. This register is useful for debuggers.

The reset value of ADG1EMUBUFFER comes from RAM and is random after reset.

Figure 22-64. ADG1EMUBUFFER Register (12-bit)

31	30	29	28	27	26	25	24
G1_EMPTY	RESERVED						
R-1h	R-0h						
23	22	21	20	19	18	17	16
RESERVED			G1_CHID				
R-0h			R-X				
15	14	13	12	11	10	9	8
RESERVED				G1_DR			
R-0h				R-X			
7	6	5	4	3	2	1	0
G1_DR							
R-X							

Figure 22-65. ADG1EMUBUFFER Register (10-bit)

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
G1_EMPTY	G1_CHID					G1_DR	
R-1h	R-X					R-X	
7	6	5	4	3	2	1	0
G1_DR							
R-X							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 22-48. ADG1EMUBUFFER Register Field Descriptions

Bit	Field	Type	Reset	Description
31	G1_EMPTY	R	1h	Group1 FIFO Empty. This bit is applicable only when the "read from FIFO" mode is used for reading the Group1 conversion results. Any operation mode read: 0h = The data in the G1_DR field of this buffer is valid. 1h = The data in the G1_DR field of this buffer is not valid and there are no valid data in the Group1 results memory.
30-21	RESERVED	R	0h	Reads return 0, writes have no effect.
20-16	G1_CHID	R	X	Group1 Channel Id. These bits are also applicable only when the "read from FIFO" mode is used for reading the Group1 conversion results. The conversion result in the G1_DR field of this buffer is from the ADC input channel number denoted by the G1_CHID field from 1h to 1Fh. Any operation mode read: 0h = The conversion result in the G1_DR field of this buffer is from the ADC input channel 0, or the channel id mode is disabled in the Group1 mode control register (ADG1MODECR).
15-12	RESERVED	R	0h	Reads return 0, writes have no effect.
11-0	G1_DR	R	X	Group1 Digital Conversion Result. The Group1 results' FIFO location is aliased eight times, so that any word-aligned read from the address range B0h to CFh results in one conversion result to be read from the Group1 results' memory. This allows the ARM LDMIA instruction to read out up to 8 conversion results from the Group1 results' memory with just one instruction.

22.11.42 ADG2EMUBUFFER Register (Offset = F8h) [reset = X]

ADG2EMUBUFFER is shown in Figure 22-66 and Figure 22-67, and described in Table 22-49.

Return to Summary Table.

The format of the data read from the ADG2EMUBUFFER locations is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module.

A read from this location also gives out one conversion result from the Group2 results' memory along with the G2_EMPTY status bit and the optional channel id. However, this read does not affect any of the status flags in the Group2 interrupt flag register or the Group2 status register. This register is useful for debuggers.

The reset value of ADG2EMUBUFFER comes from RAM and is random after reset.

Figure 22-66. ADG2EMUBUFFER Register (12-bit)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td></tr>
<tr><td>G2_EMPTY</td><td colspan="7">RESERVED</td></tr>
<tr><td>R-1h</td><td colspan="7">R-0h</td></tr>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="3">RESERVED</td><td colspan="5">G2_CHID</td></tr>
<tr><td colspan="3">R-0h</td><td colspan="5">R-X</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="4">RESERVED</td><td colspan="4">G2_DR</td></tr>
<tr><td colspan="4">R-0h</td><td colspan="4">R-X</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">G2_DR</td></tr>
<tr><td colspan="8">R-X</td></tr>
</table>

Figure 22-67. ADG2EMUBUFFER Register (10-bit)

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td>G2_EMPTY</td><td colspan="5">G2_CHID</td><td colspan="2">G2_DR</td></tr>
<tr><td>R-1h</td><td colspan="5">R-X</td><td colspan="2">R-X</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">G2_DR</td></tr>
<tr><td colspan="8">R-X</td></tr>
</table>

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 22-49. ADG2EMUBUFFER Register Field Descriptions

Bit	Field	Type	Reset	Description
31	G2_EMPTY	R	1h	Group2 FIFO Empty. This bit is applicable only when the "read from FIFO" mode is used for reading the Group2 conversion results. Any operation mode read: 0h = The data in the G2_DR field of this buffer is valid. 1h = The data in the G2_DR field of this buffer is not valid and there are no valid data in the Group2 results memory.
30-21	RESERVED	R	0h	Reads return 0, writes have no effect.
20-16	G2_CHID	R	X	Group2 Channel Id. These bits are also applicable only when the "read from FIFO" mode is used for reading the Group2 conversion results. The conversion result in the G2_DR field of this buffer is from the ADC input channel number denoted by the G2_CHID field from 1h to 1Fh. Any operation mode read: 0h = The conversion result in the G2_DR field of this buffer is from the ADC input channel 0, or the channel id mode is disabled in the Group2 mode control register (ADG2MODECR).
15-12	RESERVED	R	0h	Reads return 0, writes have no effect.
11-0	G2_DR	R	X	Group2 Digital Conversion Result. The Group2 results' FIFO location is aliased eight times, so that any word-aligned read from the address range D0h to EFh results in one conversion result to be read from the Group2 results' memory. This allows the ARM LDMIA instruction to read out up to 8 conversion results from the Group2 results' memory with just one instruction.

22.11.43 ADEVTDIR Register (Offset = FCh) [reset = 0h]

ADEVTDIR is shown in Figure 22-68 and described in Table 22-50.

Return to Summary Table.

Figure 22-68. ADEVTDIR Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="7">RESERVED</td><td>ADEVT_DIR</td></tr>
<tr><td colspan="7">R-0h</td><td>R/W-0h</td></tr>
</table>

Table 22-50. ADEVTDIR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reads return 0, writes have no effect.
0	ADEVT_DIR	R/W	0h	ADEVT Pin Direction. Any operation mode: 0h = ADEVT is an input pin, the output buffer is disabled. 1h = ADEVT is an output pin, the output buffer is enabled.

22.11.44 ADEVTOUT Register (Offset = 100h) [reset = X]

ADEVTOUT is shown in Figure 22-69 and described in Table 22-51.

Return to Summary Table.

The reset value of ADEVTOUT is unknown.

Figure 22-69. ADEVTOUT Register

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

Table 22-51. ADEVTOUT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reads return 0, writes have no effect.
0	ADEVT_OUT	R/W	X	ADEVT Pin Output Value. This bit determines the logic level to be output to the ADEVT pin when the pin is configured to be an output pin. Any operation mode: 0h = Output logic LOW on the ADEVT pin. 1h = Output logic HIGH on the ADEVT pin.

22.11.45 ADEVTIN Register (Offset = 104h) [reset = X]

ADEVTIN is shown in Figure 22-70 and described in Table 22-52.

Return to Summary Table.

The reset value of ADEVTIN is unknown.

Figure 22-70. ADEVTIN Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="7">RESERVED</td><td>ADEVT_IN</td></tr>
<tr><td colspan="7">R-0h</td><td>R-X</td></tr>
</table>

Table 22-52. ADEVTIN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reads return 0, writes have no effect.
0	ADEVT_IN	R	X	ADEVT Pin Input Value. This is a read-only bit that reflects the logic level on the ADEVT pin. Any operation mode read: 0h = Logic LOW present on the ADEVT pin. 1h = Logic HIGH present on the ADEVT pin.

22.11.46 ADEVTSET Register (Offset = 108h) [reset = X]

ADEVTSET is shown in Figure 22-71 and described in Table 22-53.

Return to Summary Table.

The reset value of ADEVTSET is unknown.

Figure 22-71. ADEVTSET Register

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

Table 22-53. ADEVTSET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reads return 0, writes have no effect.
0	ADEVT_SET	R/W	X	ADEVT Pin Set. This bit drives the output of the ADEVT pin high. A read from this bit always returns the current state of the ADEVT pin. Any operation mode: 0h = Output value on the ADEVT pin is unchanged. 1h = Output logic HIGH on the ADEVT pin, if the pin is configured to be an output pin.

22.11.47 ADEVTCLR Register (Offset = 10Ch) [reset = X]

ADEVTCLR is shown in Figure 22-72 and described in Table 22-54.

Return to Summary Table.

The reset value of ADEVTCLR is unknown.

Figure 22-72. ADEVTCLR Register

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

Table 22-54. ADEVTCLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reads return 0, writes have no effect.
0	ADEVT_CLR	R/W	X	ADEVT Pin Clear. A read from this bit always returns the current state of the ADEVT pin. Any operation mode: 0h = Output value on the ADEVT pin is unchanged. 1h = Output logic LOW on the ADEVT pin, if the pin is configured to be an output pin.

22.11.48 ADEVTPDR Register (Offset = 110h) [reset = 0h]

ADEVTPDR is shown in Figure 22-73 and described in Table 22-55.

Return to Summary Table.

Figure 22-73. ADEVTPDR Register

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

Table 22-55. ADEVTPDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reads return 0, writes have no effect.
0	ADEVT_PDR	R/W	0h	ADEVT Pin Open Drain Enable. This bit enables the open-drain capability for the ADEVT pin if it is configured to be an output and a logic HIGH is being driven on to the pin. Any operation mode: 0h = Output value on the ADEVT pin is logic HIGH. 1h = The ADEVT pin is tristated.

22.11.49 ADEVTPDIS Register (Offset = 114h) [reset = 0h]

ADEVTPDIS is shown in Figure 22-74 and described in Table 22-56.

Return to Summary Table.

Figure 22-74. ADEVTPDIS Register

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

Table 22-56. ADEVTPDIS Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reads return 0, writes have no effect.
0	ADEVT_PDIS	R/W	0h	ADEVT Pin Pull Control Disable. This bit enables or disables the pull control on the ADEVT pin if it is configured to be an input pin. Any operation mode: 0h = Pull on ADEVT pin is enabled. 1h = Pull on ADEVT pin is disabled.

22.11.50 ADEVTPSEL Register (Offset = 118h) [reset = 0h]

ADEVTPSEL is shown in Figure 22-75 and described in Table 22-57.

Return to Summary Table.

Figure 22-75. ADEVTPSEL Register

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

Table 22-57. ADEVTPSEL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reads return 0, writes have no effect.
0	ADEVT_PSEL	R/W	0h	ADEVT Pin Pull Control Select. This bit selects a pull-down or pull-up on the ADEVT pin if it is configured to be an input pin. Any operation mode: 0h = Pull down is selected on ADEVT pin. 1h = Pull up is selected on ADEVT pin.

22.11.51 ADEVSAMPDISEN Register (Offset = 11Ch) [reset = 0h]

ADEVSAMPDISEN is shown in Figure 22-76 and described in Table 22-58.

Return to Summary Table.

Figure 22-76. ADEVSAMPDISEN Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="8">EV_SAMP_DIS_CYC</td></tr>
<tr><td colspan="8">R/W-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="7">RESERVED</td><td>EV_SAMP_
DIS_EN</td></tr>
<tr><td colspan="7">R-0h</td><td>R/W-0h</td></tr>
</table>

Table 22-58. ADEVSAMPDISEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reads return 0, writes have no effect.
15-8	EV_SAMP_DIS_CYC	R/W	0h	Event Group sample cap discharge cycles. These bits specify the duration in terms of ADCLK cycles for which the ADC internal sampling capacitor is allowed to discharge before sampling the input channel voltage.
7-1	RESERVED	R	0h	Reads return 0, writes have no effect.
0	EV_SAMP_DIS_EN	R/W	0h	Event Group sample cap discharge enable. Any operation mode: 0h = Event Group sample cap discharge mode is disabled. 1h = Event Group sample cap discharge mode is enabled. The ADC internal sampling capacitor is connected to the V_{refLo} reference voltage for a duration specified by the EV_SAMP_DIS_CYC field. After this discharge time has expired, the selected ADC input channel is sampled and converted normally based on the Event Group settings.

22.11.52 ADG1SAMPDISEN Register (Offset = 120h) [reset = 0h]

ADG1SAMPDISEN is shown in Figure 22-77 and described in Table 22-59.

Return to Summary Table.

Figure 22-77. ADG1SAMPDISEN Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RESERVED							
R-0h							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
G1_SAMP_DIS_CYC							
R/W-0h							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RESERVED							G1_SAMP_DIS_EN
R-0h							R/W-0h

Table 22-59. ADG1SAMPDISEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reads return 0, writes have no effect.
15-8	G1_SAMP_DIS_CYC	R/W	0h	Group1 sample cap discharge cycles. These bits specify the duration in terms of ADCLK cycles for which the ADC internal sampling capacitor is allowed to discharge before sampling the input channel voltage.
7-1	RESERVED	R	0h	Reads return 0, writes have no effect.
0	G1_SAMP_DIS_EN	R/W	0h	Group1 sample cap discharge enable. Any operation mode: 0h = Group1 sample cap discharge mode is disabled. 1h = Group1 sample cap discharge mode is enabled. The ADC internal sampling capacitor is connected to the V_{refLo} reference voltage for a duration specified by the G1_SAMP_DIS_CYC field. After this discharge time has expired, the selected ADC input channel is sampled and converted normally based on the Group1 settings.

22.11.53 ADG2SAMPDISEN Register (Offset = 124h) [reset = 0h]

ADG2SAMPDISEN is shown in Figure 22-78 and described in Table 22-60.

Return to Summary Table.

Figure 22-78. ADG2SAMPDISEN Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
G2_SAMP_DIS_CYC							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED							G2_SAMP_DIS_EN
R-0h							R/W-0h

Table 22-60. ADG2SAMPDISEN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reads return 0, writes have no effect.
15-8	G2_SAMP_DIS_CYC	R/W	0h	Group2 sample cap discharge cycles. These bits specify the duration in terms of ADCLK cycles for which the ADC internal sampling capacitor is allowed to discharge before sampling the input channel voltage.
7-1	RESERVED	R	0h	Reads return 0, writes have no effect.
0	G2_SAMP_DIS_EN	R/W	0h	Group2 sample cap discharge enable. Any operation mode: 0h = Group2 sample cap discharge mode is disabled. 1h = Group2 sample cap discharge mode is enabled. The ADC internal sampling capacitor is connected to the V_{refLo} reference voltage for a duration specified by the G2_SAMP_DIS_CYC field. After this discharge time has expired, the selected ADC input channel is sampled and converted normally based on the Group2 settings.

22.11.54 ADMAGINT1CR Register (Offset = 128h) [reset = 0h]

ADMAGINT1CR is shown in Figure 22-79 and Figure 22-80, and described in Table 22-61.

Return to Summary Table.

The format of the ADMAGINT1CR is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module. The ADC module supports up to three magnitude compare interrupts.

Figure 22-79. ADMAGINT1CR Register (12-bit)

31	30	29	28	27	26	25	24
RESERVED						MAG_THR1	
R-0h						R/W-0h	
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
MAG_THR1							
R/W-0h							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CHN_THR_ COMP1	CMP_GE_LT1	RESERVED	COMP_CHID1				
R/W-0h	R/W-0h	R-0h	R/W-0h				
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RESERVED			MAG_CHID1				
R-0h			R/W-0h				

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Figure 22-80. ADMAGINT1CR Register (10-bit)

31	30	29	28	27	26	25	24
RESERVED				MAG_THR1			
R-0h				R/W-0h			
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
MAG_THR1							
R/W-0h							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CHN_THR_ COMP1	CMP_GE_LT1	RESERVED	COMP_CHID1				
R/W-0h	R/W-0h	R-0h	R/W-0h				
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RESERVED			MAG_CHID1				
R-0h			R/W-0h				

Table 22-61. ADMAGINT1CR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	Reads return 0, writes have no effect.
27-16	MAG_THR1	R/W	0h	These bits specify the 12-bit or 10-bit compare value that the ADC uses for the comparison with the MAG_CHID1 channel's conversion result.
15	CHN_THR_COMP1	R/W	0h	Channel OR Threshold comparison. Both the MAG_CHID1 and the COMP_CHID1 channel must have been converted at least once for the ADC to perform the comparison. Any operation mode: 0h = The ADC module compares the MAG_CHID1 channel's conversion result with the fixed threshold value specified by the MAG_THR1 field. 1h = The ADC module compares the MAG_CHID1 channel's conversion result with the last conversion result for the COMP_CHID1 channel.
14	CMP_GE_LT1	R/W	0h	Greater than or equal to OR Less than comparison operator. Any operation mode: 0h = The ADC module checks if the conversion result is lower than the reference value (fixed threshold or COMP_CHID1 conversion result). 1h = The ADC module checks if the conversion result is greater than or equal to the reference value (fixed threshold or COMP_CHID1 conversion result).
13	RESERVED	R	0h	Reads return 0, writes have no effect.
12-8	COMP_CHID1	R/W	0h	These bits specify the channel number from 0 to 31 whose last conversion result is compared with the MAG_CHID1 channel's conversion result.
7-5	RESERVED	R	0h	Reads return 0, writes have no effect.
4-0	MAG_CHID1	R/W	0h	These bits specify the channel number from 0 to 31 for which the conversion result needs to be monitored by the ADC.

22.11.55 ADMAGINT1MASK Register (Offset = 12Ch) [reset = 0h]

ADMAGINT1MASK is shown in Figure 22-81 and described in Table 22-62.

Return to Summary Table.

The format of the ADMAGINT1MASK is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module. The ADC module supports up to three magnitude compare interrupts.

Figure 22-81. ADMAGINT1MASK Register

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED				MAG_INT1_MASK											
R-0h				R/W-0h											

Table 22-62. ADMAGINT1MASK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0h	Reads return 0, writes have no effect.
11-0	MAG_INT1_MASK	R/W	0h	These bits specify the mask for the comparison in order to generate the magnitude compare interrupt #1. **Note: The actual size of the MAG_INT1_MASK field is 12 bits or 10 bits depending on whether the ADC is configured to be in 12-bit or 10-bit resolution mode, respectively. Bits 11-10 are reserved when the module is configured as a 10-bit ADC module.** Any operation mode: 0h = The ADC module does not mask the corresponding bit for the comparison. 1h = The ADC module masks the corresponding bit for the comparison.

22.11.56 ADMAGINT2CR Register (Offset = 130h) [reset = 0h]

ADMAGINT2CR is shown in Figure 22-82 and Figure 22-83, and described in Table 22-63.

Return to Summary Table.

The format of the ADMAGINT2CR is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module. The ADC module supports up to three magnitude compare interrupts.

Figure 22-82. ADMAGINT2CR Register (12-bit)

31	30	29	28	27	26	25	24
RESERVED						MAG_THR2	
R-0h						R/W-0h	
23	22	21	20	19	18	17	16
MAG_THR2							
R/W-0h							
15	14	13	12	11	10	9	8
CHN_THR_ COMP2	CMP_GE_LT2	RESERVED	COMP_CHID2				
R/W-0h	R/W-0h	R-0h	R/W-0h				
7	6	5	4	3	2	1	0
RESERVED			MAG_CHID2				
R-0h			R/W-0h				

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Figure 22-83. ADMAGINT2CR Register (10-bit)

31	30	29	28	27	26	25	24
RESERVED				MAG_THR2			
R-0h				R/W-0h			
23	22	21	20	19	18	17	16
MAG_THR2							
R/W-0h							
15	14	13	12	11	10	9	8
CHN_THR_ COMP2	CMP_GE_LT2	RESERVED	COMP_CHID2				
R/W-0h	R/W-0h	R-0h	R/W-0h				
7	6	5	4	3	2	1	0
RESERVED			MAG_CHID2				
R-0h			R/W-0h				

Table 22-63. ADMAGINT2CR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	Reads return 0, writes have no effect.
27-16	MAG_THR2	R/W	0h	These bits specify the 12-bit or 10-bit compare value that the ADC uses for the comparison with the MAG_CHID2 channel's conversion result.
15	CHN_THR_COMP2	R/W	0h	Channel OR Threshold comparison. Both the MAG_CHID2 and the COMP_CHID2 channel must have been converted at least once for the ADC to perform the comparison. Any operation mode: 0h = The ADC module compares the MAG_CHID2 channel's conversion result with the fixed threshold value specified by the MAG_THR2 field. 1h = The ADC module compares the MAG_CHID2 channel's conversion result with the last conversion result for the COMP_CHID2 channel.
14	CMP_GE_LT2	R/W	0h	Greater than or equal to OR Less than comparison operator. Any operation mode: 0h = The ADC module checks if the conversion result is lower than the reference value (fixed threshold or COMP_CHID2 conversion result). 1h = The ADC module checks if the conversion result is greater than or equal to the reference value (fixed threshold or COMP_CHID2 conversion result).
13	RESERVED	R	0h	Reads return 0, writes have no effect.
12-8	COMP_CHID2	R/W	0h	These bits specify the channel number from 0 to 31 whose last conversion result is compared with the MAG_CHID2 channel's conversion result.
7-5	RESERVED	R	0h	Reads return 0, writes have no effect.
4-0	MAG_CHID2	R/W	0h	These bits specify the channel number from 0 to 31 for which the conversion result needs to be monitored by the ADC.

22.11.57 ADMAGINT2MASK Register (Offset = 134h) [reset = 0h]

ADMAGINT2MASK is shown in Figure 22-84 and described in Table 22-64.

Return to Summary Table.

The format of the ADMAGINT2MASK is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module. The ADC module supports up to three magnitude compare interrupts.

Figure 22-84. ADMAGINT2MASK Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">RESERVED</td></tr>
<tr><td colspan="16">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="4">RESERVED</td><td colspan="12">MAG_INT2_MASK</td></tr>
<tr><td colspan="4">R-0h</td><td colspan="12">R/W-0h</td></tr>
</table>

Table 22-64. ADMAGINT2MASK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0h	Reads return 0, writes have no effect.
11-0	MAG_INT2_MASK	R/W	0h	These bits specify the mask for the comparison in order to generate the magnitude compare interrupt #2. **Note: The actual size of the MAG_INT2_MASK field is 12 bits or 10 bits depending on whether the ADC is configured to be in 12-bit or 10-bit resolution mode, respectively. Bits 11-10 are reserved when the module is configured as a 10-bit ADC module.** Any operation mode: 0h = The ADC module does not mask the corresponding bit for the comparison. 1h = The ADC module masks the corresponding bit for the comparison.

22.11.58 ADMAGINT3CR Register (Offset = 138h) [reset = 0h]

ADMAGINT3CR is shown in Figure 22-85 and Figure 22-86, and described in Table 22-65.

Return to Summary Table.

The format of the ADMAGINT3CR is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module. The ADC module supports up to three magnitude compare interrupts.

Figure 22-85. ADMAGINT3CR Register (12-bit)

31	30	29	28	27	26	25	24
RESERVED						MAG_THR3	
R-0h						R/W-0h	
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
MAG_THR3							
R/W-0h							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CHN_THR_ COMP3	CMP_GE_LT3	RESERVED	COMP_CHID3				
R/W-0h	R/W-0h	R-0h	R/W-0h				
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RESERVED			MAG_CHID3				
R-0h			R/W-0h				

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Figure 22-86. ADMAGINT3CR Register (10-bit)

31	30	29	28	27	26	25	24
RESERVED				MAG_THR3			
R-0h				R/W-0h			
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
MAG_THR3							
R/W-0h							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CHN_THR_ COMP3	CMP_GE_LT3	RESERVED	COMP_CHID3				
R/W-0h	R/W-0h	R-0h	R/W-0h				
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RESERVED			MAG_CHID3				
R-0h			R/W-0h				

Table 22-65. ADMAGINT3CR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	RESERVED	R	0h	Reads return 0, writes have no effect.
27-16	MAG_THR3	R/W	0h	These bits specify the 12-bit or 10-bit compare value that the ADC uses for the comparison with the MAG_CHID3 channel's conversion result.
15	CHN_THR_COMP3	R/W	0h	Channel OR Threshold comparison. Both the MAG_CHID3 and the COMP_CHID3 channel must have been converted at least once for the ADC to perform the comparison. Any operation mode: 0h = The ADC module compares the MAG_CHID3 channel's conversion result with the fixed threshold value specified by the MAG_THR3 field. 1h = The ADC module compares the MAG_CHID3 channel's conversion result with the last conversion result for the COMP_CHID3 channel.
14	CMP_GE_LT3	R/W	0h	Greater than or equal to OR Less than comparison operator. Any operation mode: 0h = The ADC module checks if the conversion result is lower than the reference value (fixed threshold or COMP_CHID3 conversion result). 1h = The ADC module checks if the conversion result is greater than or equal to the reference value (fixed threshold or COMP_CHID3 conversion result).
13	RESERVED	R	0h	Reads return 0, writes have no effect.
12-8	COMP_CHID3	R/W	0h	These bits specify the channel number from 0 to 31 whose last conversion result is compared with the MAG_CHID3 channel's conversion result.
7-5	RESERVED	R	0h	Reads return 0, writes have no effect.
4-0	MAG_CHID3	R/W	0h	These bits specify the channel number from 0 to 31 for which the conversion result needs to be monitored by the ADC.

22.11.59 ADMAGINT3MASK Register (Offset = 13Ch) [reset = 0h]

ADMAGINT3MASK is shown in Figure 22-87 and described in Table 22-66.

Return to Summary Table.

The format of the ADMAGINT3MASK is different based on whether the ADC module is configured to be a 12-bit or a 10-bit ADC module. The ADC module supports up to three magnitude compare interrupts.

Figure 22-87. ADMAGINT3MASK Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">RESERVED</td></tr>
<tr><td colspan="16">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="4">RESERVED</td><td colspan="12">MAG_INT3_MASK</td></tr>
<tr><td colspan="4">R-0h</td><td colspan="12">R/W-0h</td></tr>
</table>

Table 22-66. ADMAGINT3MASK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0h	Reads return 0, writes have no effect.
11-0	MAG_INT3_MASK	R/W	0h	These bits specify the mask for the comparison in order to generate the magnitude compare interrupt #3. **Note: The actual size of the MAG_INT3_MASK field is 12 bits or 10 bits depending on whether the ADC is configured to be in 12-bit or 10-bit resolution mode, respectively. Bits 11-10 are reserved when the module is configured as a 10-bit ADC module.** Any operation mode: 0h = The ADC module does not mask the corresponding bit for the comparison. 1h = The ADC module masks the corresponding bit for the comparison.

22.11.60 ADMAGINTENASET Register (Offset = 158h) [reset = 0h]

ADMAGINTENASET is shown in Figure 22-88 and described in Table 22-67.

Return to Summary Table.

Figure 22-88. ADMAGINTENASET Register

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

Table 22-67. ADMAGINTENASET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reads return 0, writes have no effect.
2-0	MAG_INT_ENA_SET	R/W	0h	Each of these three bits, when set, enable the corresponding magnitude compare interrupt. Any operation mode for each bit: 0h = The enable status of the corresponding magnitude compare interrupt is left unchanged. 1h = The corresponding magnitude compare interrupt is enabled.

22.11.61 ADMAGINTENACLR Register (Offset = 15Ch) [reset = 0h]

ADMAGINTENACLR is shown in Figure 22-89 and described in Table 22-68.

Return to Summary Table.

Figure 22-89. ADMAGINTENACLR Register

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

Table 22-68. ADMAGINTENACLR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reads return 0, writes have no effect.
2-0	MAG_INT_ENA_CLR	R/W	0h	Each of these three bits, when set, enable the corresponding magnitude compare interrupt. Any operation mode for each bit: 0h = The enable status of the corresponding magnitude compare interrupt is left unchanged. 1h = The corresponding magnitude compare interrupt is disabled.

22.11.62 ADMAGINTFLG Register (Offset = 160h) [reset = 0h]

ADMAGINTFLG is shown in Figure 22-90 and described in Table 22-69.

Return to Summary Table.

Figure 22-90. ADMAGINTFLG Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="5">RESERVED</td><td colspan="3">MAG_INT_FLG</td></tr>
<tr><td colspan="5">R-0h</td><td colspan="3">R/W-0h</td></tr>
</table>

Table 22-69. ADMAGINTFLG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-3	RESERVED	R	0h	Reads return 0, writes have no effect.
2-0	MAG_INT_FLG	R/W	0h	Magnitude Compare Interrupt Flags. These bits are polled by the application to determine if the magnitude compares have been evaluated as true. When a magnitude compare interrupt flag is set, the corresponding magnitude compare interrupt is generated if enabled. Any operation mode for each bit: 0h (W) = The corresponding flag is left unchanged. 0h (R) = The condition for the corresponding magnitude threshold interrupt was false. 1h (W) = The corresponding flag is cleared. The flag is also cleared by reading from the magnitude compare interrupt offset register. 1h (R) = The condition for the corresponding magnitude threshold interrupt was true.

22.11.63 ADMAGINTOFF Register (Offset = 164h) [reset = 0h]

ADMAGINTOFF is shown in Figure 22-91 and described in Table 22-70.

Return to Summary Table.

Figure 22-91. ADMAGINTOFF Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">RESERVED</td></tr>
<tr><td colspan="16">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="12">RESERVED</td><td colspan="4">MAG_INT_OFF</td></tr>
<tr><td colspan="12">R-0h</td><td colspan="4">RC-0h</td></tr>
</table>

Table 22-70. ADMAGINTOFF Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reads return 0, writes have no effect.
3-0	MAG_INT_OFF	RC	0h	Magnitude Compare Interrupt Offset. This field indexes the currently highest-priority magnitude compare interrupt. Interrupt 1 has the highest priority and interrupt 3 has the lowest priority among the magnitude compare interrupts. Writes to these bits have no effect. A read from this register clears this register as well as the corresponding magnitude compare interrupt flag in the ADMAGINTFLG register. However, a read from this register in emulation mode does not affect this register or the interrupt status flags. Any operation mode read: 0h = No magnitude compare interrupt is pending. 1h = Magnitude compare interrupt # 1 is pending. 2h = Magnitude compare interrupt # 2 is pending. 3h = Magnitude compare interrupt # 3 is pending. 4h = Reserved 5h = Reserved 6h = Reserved 7h = Reserved

22.11.64 ADEVFIFORESETCR Register (Offset = 168h) [reset = 0h]

ADEVFIFORESETCR is shown in Figure 22-92 and described in Table 22-71.

Return to Summary Table.

Figure 22-92. ADEVFIFORESETCR Register

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

Table 22-71. ADEVFIFORESETCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reads return 0, writes have no effect.
0	EV_FIFO_RESET	R/W	0h	ADC Event Group FIFO Reset. The application sets this bit in case of an overrun condition. This allows the ADC module to overwrite the contents of the Event Group results memory starting from the first location. The EV_FIFO_RESET bit only has the desired effect when the Event Group results memory is in an overrun condition. It must be used when the data already available in the results memory is discarded. If the application needs the Event Group memory to always be overwritten with the latest available conversion results, then the OVR_EV_RAM_IGN bit in the Event Group operating mode control register (ADEVMODECR) needs to be set to 1. 1h = The ADC module resets its internal Event Group results memory pointers. Then this bit automatically gets cleared, so that the ADC module allows the Event Group results memory to be overwritten only once each time this bit is set to 1. As a result, the EV_FIFO_RESET bit is always read as a 0.

22.11.65 ADG1FIFORESETCR Register (Offset = 16Ch) [reset = 0h]

ADG1FIFORESETCR is shown in Figure 22-93 and described in Table 22-72.

Return to Summary Table.

Figure 22-93. ADG1FIFORESETCR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							G1_FIFO_ RESET
R-0h							R/W-0h

Table 22-72. ADG1FIFORESETCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reads return 0, writes have no effect.
0	G1_FIFO_RESET	R/W	0h	ADC Group1 FIFO Reset. The application sets this bit in case of an overrun condition. This allows the ADC module to overwrite the contents of the Group1 results memory starting from the first location from 0 to 1. The G1_FIFO_RESET bit only has the desired effect when the Group1 results memory is in an overrun condition. It must be used when the data already available in the results memory is discarded. If the application needs the Group1 memory to always be overwritten with the latest available conversion results, then the OVR_G1_RAM_IGN bit in the Group1 operating mode control register (ADG1MODECR) needs to be set to 1. 1h = The ADC module resets its internal Group1 results memory pointers. Then this bit automatically gets cleared, so that the ADC module allows the Group1 results memory to be overwritten only once each time this bit is set to 1. As a result, the G1_FIFO_RESET bit is always read as a 0.

22.11.66 ADG2FIFORESETCR Register (Offset = 170h) [reset = 0h]

ADG2FIFORESETCR is shown in Figure 22-94 and described in Table 22-73.

Return to Summary Table.

Figure 22-94. ADG2FIFORESETCR Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="7">RESERVED</td><td>G2_FIFO_ RESET</td></tr>
<tr><td colspan="7">R-0h</td><td>R/W-0h</td></tr>
</table>

Table 22-73. ADG2FIFORESETCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	RESERVED	R	0h	Reads return 0, writes have no effect.
0	G2_FIFO_RESET	R/W	0h	ADC Group2 FIFO Reset. The application sets this bit in case of an overrun condition. This allows the ADC module to overwrite the contents of the Group2 results memory starting from the first location from 0 to 1. The G2_FIFO_RESET bit only has the desired effect when the Group2 results memory is in an overrun condition. It must be used when the data already available in the results memory is discarded. If the application needs the Group2 memory to always be overwritten with the latest available conversion results, then the OVR_G2_RAM_IGN bit in the Group2 operating mode control register (ADG2MODECR) needs to be set to 1. 1h = The ADC module resets its internal Group2 results memory pointers. Then this bit automatically gets cleared, so that the ADC module allows the Group2 results memory to be overwritten only once each time this bit is set to 1. As a result, the G2_FIFO_RESET bit is always read as a 0.

22.11.67 ADEVRAMWRADDR Register (Offset = 174h) [reset = 0h]

ADEVRAMWRADDR is shown in Figure 22-95 and described in Table 22-74.

Return to Summary Table.

Figure 22-95. ADEVRAMWRADDR Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">RESERVED</td></tr>
<tr><td colspan="16">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="7">RESERVED</td><td colspan="9">EV_RAM_ADDR</td></tr>
<tr><td colspan="7">R-0h</td><td colspan="9">R/W-0h</td></tr>
</table>

Table 22-74. ADEVRAMWRADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Reads return 0, writes have no effect.
8-0	EV_RAM_ADDR	R/W	0h	Event Group results memory write pointer. This field shows the address of the location where the next Event Group conversion result is stored. This is specified in terms of the buffer number. The application reads this register to determine the number of valid Event Group conversion results available until that time.

22.11.68 ADG1RAMWRADDR Register (Offset = 178h) [reset = 0h]

ADG1RAMWRADDR is shown in Figure 22-96 and described in Table 22-75.

Return to Summary Table.

Figure 22-96. ADG1RAMWRADDR Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">RESERVED</td></tr>
<tr><td colspan="16">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="7">RESERVED</td><td colspan="9">G1_RAM_ADDR</td></tr>
<tr><td colspan="7">R-0h</td><td colspan="9">R/W-0h</td></tr>
</table>

Table 22-75. ADG1RAMWRADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Reads return 0, writes have no effect.
8-0	G1_RAM_ADDR	R/W	0h	Group1 results memory write pointer. This field shows the address of the location where the next Group1 conversion result is stored. This is specified in terms of the buffer number. The application reads this register to determine the number of valid Group1 conversion results available until that time.

22.11.69 ADG2RAMWRADDR Register (Offset = 17Ch) [reset = 0h]

ADG2RAMWRADDR is shown in Figure 22-97 and described in Table 22-76.

Return to Summary Table.

Figure 22-97. ADG2RAMWRADDR Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">RESERVED</td></tr>
<tr><td colspan="16">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="7">RESERVED</td><td colspan="9">G2_RAM_ADDR</td></tr>
<tr><td colspan="7">R-0h</td><td colspan="9">R/W-0h</td></tr>
</table>

Table 22-76. ADG2RAMWRADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Reads return 0, writes have no effect.
8-0	G2_RAM_ADDR	R/W	0h	Group2 results memory write pointer. This field shows the address of the location where the next Group2 conversion result is stored. This is specified in terms of the buffer number. The application reads this register to determine the number of valid Group2 conversion results available until that time.

22.11.70 ADPARCR Register (Offset = 180h) [reset = 5h]

ADPARCR is shown in Figure 22-98 and described in Table 22-77.

Return to Summary Table.

Figure 22-98. ADPARCR Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="7">RESERVED</td><td>TEST</td></tr>
<tr><td colspan="7">R-0h</td><td>R/WP-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="4">RESERVED</td><td colspan="4">PARITY_ENA</td></tr>
<tr><td colspan="4">R-0h</td><td colspan="4">R/WP-5h</td></tr>
</table>

Table 22-77. ADPARCR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	R	0h	Reads return 0, writes have no effect.
8	TEST	R/WP	0h	This bit maps the parity bits into the ADC results' RAM frame so that the application accesses them. Any operation mode read, privileged mode write: 0h = The parity bits are not memory-mapped. 1h = The parity bits are memory mapped.
7-4	RESERVED	R	0h	Reads return 0, writes have no effect.
3-0	PARITY_ENA	R/WP	5h	Enable parity checking. These bits enable or disable the parity check on read operations and the parity calculation on write operations to the ADC results memory. If parity checking is enabled and a parity error is detected, the ADC module sends a parity error signal to the System module. Any other value other than 5h, parity check is enabled. Any operation mode read, privileged mode write: 5h = Parity check is disabled.

22.11.71 ADPARADDR Register (Offset = 184h) [reset = X]

ADPARADDR is shown in Figure 22-99 and described in Table 22-78.

Return to Summary Table.

Figure 22-99. ADPARADDR Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED				ERROR_ADDRESS			
R-0h				R-X			
7	6	5	4	3	2	1	0
ERROR_ADDRESS						RESERVED	
R-X						R-0h	

Table 22-78. ADPARADDR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-12	RESERVED	R	0h	Reads return 0, writes have no effect.
11-2	ERROR_ADDRESS	R	X	These bits hold the address of the first parity error generated in the ADC results' RAM. Reading bits [11:0] provides the 32-bit aligned address. This error address is frozen from being updated until it is read by the application. In emulation mode, this address is maintained frozen even when read.
1-0	RESERVED	R	0h	Reads return 0, writes have no effect.

22.11.72 ADPWRUPDLYCTRL Register (Offset = 188h) [reset = 0h]

ADPWRUPDLYCTRL is shown in Figure 22-100 and described in Table 22-79.

Return to Summary Table.

Figure 22-100. ADPWRUPDLYCTRL Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="22">RESERVED</td><td colspan="10">PWRUP_DLY</td></tr>
<tr><td colspan="22">R-0h</td><td colspan="10">R/W-0h</td></tr>
</table>

Table 22-79. ADPWRUPDLYCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0h	Reads return 0, writes have no effect.
9-0	PWRUP_DLY	R/W	0h	This register defines the number of VCLK cycles that the ADC state machine has to wait after releasing the ADC core from power down before starting a new conversion. Refer to ADC Power-Down Mode subsection for more details.

22.11.73 ADEVCHNSELMODECTRL Register (Offset = 190h) [reset = 5h]

ADEVCHNSELMODECTRL is shown in Figure 22-101 and described in Table 22-80.

Return to Summary Table.

Figure 22-101. ADEVCHNSELMODECTRL Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="4">RESERVED</td><td colspan="4">EV_ENH_CHNSEL_MODE_ENABLE</td></tr>
<tr><td colspan="4">R-0h</td><td colspan="4">R/W-5h</td></tr>
</table>

Table 22-80. ADEVCHNSELMODECTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reads return 0, writes have no effect.
3-0	EV_ENH_CHNSEL_MODE_ENABLE	R/W	5h	Enable enhanced channel selection mode for Event group. Refer to Enhanced Channel Selection Mode subsection for a description of the enhanced channel selection mode. Writing any value other than 5h or Ah to this field has no effect on the selected channel selection mode for the Event group, and the ADC module continues to use the channel selection mode that was previously programmed channel selection mode. 5h (R) = Indicates that the enhanced channel selection mode for Event group is not enabled. The default sequential channel selection mode is used for Event group conversions. 5h (W) = Disables the enhanced channel selection mode for Event group and enables the sequential channel selection mode. Ah (W) = Enables the enhanced channel selection mode for Event group. Ah (R) = Indicates that the enhanced channel selection mode for Event group is enabled.

22.11.74 ADG1CHNSELMODECTRL Register (Offset = 194h) [reset = 5h]

ADG1CHNSELMODECTRL is shown in Figure 22-102 and described in Table 22-81.

Return to Summary Table.

Figure 22-102. ADG1CHNSELMODECTRL Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="4">RESERVED</td><td colspan="4">G1_ENH_CHNSEL_MODE_ENABLE</td></tr>
<tr><td colspan="4">R-0h</td><td colspan="4">R/W-5h</td></tr>
</table>

Table 22-81. ADG1CHNSELMODECTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reads return 0, writes have no effect.
3-0	G1_ENH_CHNSEL_MODE_ENABLE	R/W	5h	Enable enhanced channel selection mode for Group1. Refer to Enhanced Channel Selection Mode subsection for a description of the enhanced channel selection mode. Writing any value other than 5h or Ah to this field has no effect on the selected channel selection mode for the Group1, and the ADC module continues to use the channel selection mode that was previously programmed channel selection mode. 5h (R) = Indicates that the enhanced channel selection mode for Group1 is not enabled. The default sequential channel selection mode is used for Group1 conversions. 5h (W) = Disables the enhanced channel selection mode for Group1 and enables the sequential channel selection mode. Ah (W) = Enables the enhanced channel selection mode for Group1. Ah (R) = Indicates that the enhanced channel selection mode for Group1 is enabled.

22.11.75 ADG2CHNSELMODECTRL Register (Offset = 198h) [reset = 5h]

ADG2CHNSELMODECTRL is shown in Figure 22-103 and described in Table 22-82.

Return to Summary Table.

Figure 22-103. ADG2CHNSELMODECTRL Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="4">RESERVED</td><td colspan="4">G2_ENH_CHNSEL_MODE_ENABLE</td></tr>
<tr><td colspan="4">R-0h</td><td colspan="4">R/W-5h</td></tr>
</table>

Table 22-82. ADG2CHNSELMODECTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	0h	Reads return 0, writes have no effect.
3-0	G2_ENH_CHNSEL_MODE_ENABLE	R/W	5h	Enable enhanced channel selection mode for Group2. Refer to Enhanced Channel Selection Mode subsection for a description of the enhanced channel selection mode. Writing any value other than 5h or Ah to this field has no effect on the selected channel selection mode for the Group2, and the ADC module continues to use the channel selection mode that was previously programmed channel selection mode. 5h (R) = Indicates that the enhanced channel selection mode for Group2 is not enabled. The default sequential channel selection mode is used for Group2 conversions. 5h (W) = Disables the enhanced channel selection mode for Group2 and enables the sequential channel selection mode. Ah (W) = Enables the enhanced channel selection mode for Group2. Ah (R) = Indicates that the enhanced channel selection mode for Group2 is enabled.

22.11.76 ADEVCURRCOUNT Register (Offset = 19Ch) [reset = 0h]

ADEVCURRCOUNT is shown in Figure 22-104 and described in Table 22-83.

Return to Summary Table.

Figure 22-104. ADEVCURRCOUNT Register

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

Table 22-83. ADEVCURRCOUNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reads return 0, writes have no effect.
4-0	EV_CURRENT_COUNT	R/W	0h	The CURRENT_COUNT value for the Event group conversions when enhanced channel selection mode is enabled. Refer to Enhanced Channel Selection Mode subsection for a description of the enhanced channel selection mode. A read from the ADEVCURRCOUNT register returns the value of the current index into the Event group's look-up table. This register resets to 0 on any of the following conditions: A peripheral reset occurs. An ADC software reset occurs via the ADC Reset Control Register (ADRSTCR). EV_CURRENT_COUNT becomes equal to EV_MAX_COUNT. Application write 0s to ADEVCURRCOUNT register. Event group's result RAM is reset.

22.11.77 ADEVMAXCOUNT Register (Offset = 1A0h) [reset = 0h]

ADEVMAXCOUNT is shown in Figure 22-105 and described in Table 22-84.

Return to Summary Table.

Figure 22-105. ADEVMAXCOUNT Register

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											EV_MAX_COUNT				
R-0h											R/W-0h				

Table 22-84. ADEVMAXCOUNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reads return 0, writes have no effect.
4-0	EV_MAX_COUNT	R/W	0h	The MAX_COUNT value for the Event group conversions when enhanced channel selection mode is enabled. Refer to Enhanced Channel Selection Mode subsection for a description of the enhanced channel selection mode. It is recommended to clear the Event group's Current Count register (ADEVCURRCOUNT) whenever the EV_MAX_COUNT is changed.

22.11.78 ADG1CURRCOUNT Register (Offset = 1A4h) [reset = 0h]

ADG1CURRCOUNT is shown in Figure 22-106 and described in Table 22-85.

Return to Summary Table.

Figure 22-106. ADG1CURRCOUNT Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="8">RESERVED</td></tr>
<tr><td colspan="8">R-0h</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="3">RESERVED</td><td colspan="5">G1_CURRENT_COUNT</td></tr>
<tr><td colspan="3">R-0h</td><td colspan="5">R/W-0h</td></tr>
</table>

Table 22-85. ADG1CURRCOUNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reads return 0, writes have no effect.
4-0	G1_CURRENT_COUNT	R/W	0h	The CURRENT_COUNT value for the Group1 conversions when enhanced channel selection mode is enabled. Refer to Enhanced Channel Selection Mode subsection for a description of the enhanced channel selection mode. A read from the ADG1CURRCOUNT register returns the value of the current index into the Group1's look-up table. This register resets to 0 on any of the following conditions: A peripheral reset occurs. An ADC software reset occurs via the ADC Reset Control Register (ADRSTCR). G1_CURRENT_COUNT becomes equal to G1_MAX_COUNT. Application write 0s to ADG1CURRCOUNT register. Group1's result RAM is reset.

22.11.79 ADG1MAXCOUNT Register (Offset = 1A8h) [reset = 0h]

ADG1MAXCOUNT is shown in Figure 22-107 and described in Table 22-86.

Return to Summary Table.

Figure 22-107. ADG1MAXCOUNT Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R-0h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED											G1_MAX_COUNT				
R-0h											R/W-0h				

Table 22-86. ADG1MAXCOUNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reads return 0, writes have no effect.
4-0	G1_MAX_COUNT	R/W	0h	The MAX_COUNT value for the Group1 conversions when enhanced channel selection mode is enabled. Refer to Enhanced Channel Selection Mode subsection for a description of the enhanced channel selection mode. It is recommended to clear the Group1's Current Count register (ADG1CURRCOUNT) whenever the G1_MAX_COUNT is changed.

22.11.80 ADG2CURRCOUNT Register (Offset = 1ACh) [reset = 0h]

ADG2CURRCOUNT is shown in Figure 22-108 and described in Table 22-87.

Return to Summary Table.

Figure 22-108. ADG2CURRCOUNT Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RESERVED							
R-0h							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RESERVED							
R-0h							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RESERVED			G2_CURRENT_COUNT				
R-0h			R/W-0h				

Table 22-87. ADG2CURRCOUNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reads return 0, writes have no effect.
4-0	G2_CURRENT_COUNT	R/W	0h	The CURRENT_COUNT value for the Group2 conversions when enhanced channel selection mode is enabled. Refer to Enhanced Channel Selection Mode subsection for a description of the enhanced channel selection mode. A read from the ADG2CURRCOUNT register returns the value of the current index into the Group2's look-up table. This register resets to 0 on any of the following conditions: A peripheral reset occurs. An ADC software reset occurs via the ADC Reset Control Register (ADRSTCR). G2_CURRENT_COUNT becomes equal to G2_MAX_COUNT. Application write 0s to ADG2CURRCOUNT register. Group2's result RAM is reset.

22.11.81 ADG2MAXCOUNT Register (Offset = 1B0h) [reset = 0h]

ADG2MAXCOUNT is shown in Figure 22-109 and described in Table 22-88.

Return to Summary Table.

Figure 22-109. ADG2MAXCOUNT Register

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">RESERVED</td></tr>
<tr><td colspan="16">R-0h</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="11">RESERVED</td><td colspan="5">G2_MAX_COUNT</td></tr>
<tr><td colspan="11">R-0h</td><td colspan="5">R/W-0h</td></tr>
</table>

Table 22-88. ADG2MAXCOUNT Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reads return 0, writes have no effect.
4-0	G2_MAX_COUNT	R/W	0h	The MAX_COUNT value for the Group2 conversions when enhanced channel selection mode is enabled. Refer to Enhanced Channel Selection Mode subsection for a description of the enhanced channel selection mode. It is recommended to clear the Group2's Current Count register (ADG2CURRCOUNT) whenever the G2_MAX_COUNT is changed.

High-End Timer (N2HET) Module

This chapter provides a general description of the High-End Timer (N2HET). The N2HET is a software-controlled timer with a dedicated specialized timer micromachine and a set of 30 instructions. The N2HET micromachine is connected to a port of up to 32 input/output (I/O) pins.

NOTE: This chapter describes a superset implementation of the N2HET module that includes features and functionality that require DMA. Since not all devices have DMA capability, consult your device-specific datasheet to determine the applicability of these features and functions to your device being used.

Topic **Page**

23.1 Overview 1072
23.2 N2HET Functional Description 1076
23.3 Angle Functions 1108
23.4 N2HET Control Registers 1135
23.5 HWAG Registers 1162
23.6 Instruction Set 1178

23.1 Overview

The N2HET is a fifth-generation Texas Instruments (TI) advanced intelligent timer module. It provides an enhanced feature set compared to previous generations.

This timer module provides sophisticated timing functions for real-time applications such as engine management or motor control. The high resolution hardware channels allow greater accuracy for widely used timing functions such as period and pulse measurements, output compare, and PWMs.

The reduced instruction set, based mostly on very simple, but comprehensive instructions, improves the definition and development cycle time of an application and its derivatives. The N2HET breakpoint feature, combined with various stop capabilities, makes the N2HET software application easy to debug.

23.1.1 Features

- Programmable timer for input and output timing functions
- Reduced instruction set (30 instructions) for dedicated time and angle functions
- Up to maximum of 128 96-bit words of instruction RAM protected by parity. Check your datasheet for the actual number of words implemented.
- User defined configuration of 25-bit virtual counters for timer, event counters and angle counters
- 7-bit hardware counters for each pin allow up to 32-bit resolution in conjunction with the 25-bit virtual counters
- Up to 32 pins usable for input signal measurements or output signal generation
- Programmable suppression filter for each input pin with adjustable suppression window
- Low CPU overhead and interrupt load
- Efficient data transfer to or from the CPU memory with dedicated High-End-Timer Transfer Unit (HTU) or DMA
- Diagnostic capabilities with different loopback mechanisms and pin status readback functionality
- Hardware Angle Generator (HWAG)

23.1.2 Major Advantages

In addition to classic time functions such as input capture or multiple PWMs, higher-level time functions can be easily implemented in the timer program main loop. Higher-level time functions include angle driven wave forms, angle- and time-driven pulses, and input pulse width modulation (PWM) duty cycle measurement.

Because of these high-level functions, data exchanges with the CPU are limited to the fundamental parameters of the application (periods, pulse widths, angle values, etc.); and the real-time constraints for parameter communication are dramatically minimized; for example, few interrupts are required and asynchronous parameter updates are allowed.

The reduced instruction set and simple execution flow control make it simple and easy to develop and modify programs. Simple algorithms can embed the entire flow control inside the N2HET program itself. More complex algorithms can take advantage of the CPU access to the N2HET RAM. With this, the CPU program can make calculations and can modify the timer program flow by changing the data and control fields of the N2HET RAM. CPU access to the N2HET RAM also improves the debug and development of timer programs. The CPU program can stop the N2HET and view the contents of the program, control, and data fields that reside in the N2HET RAM.

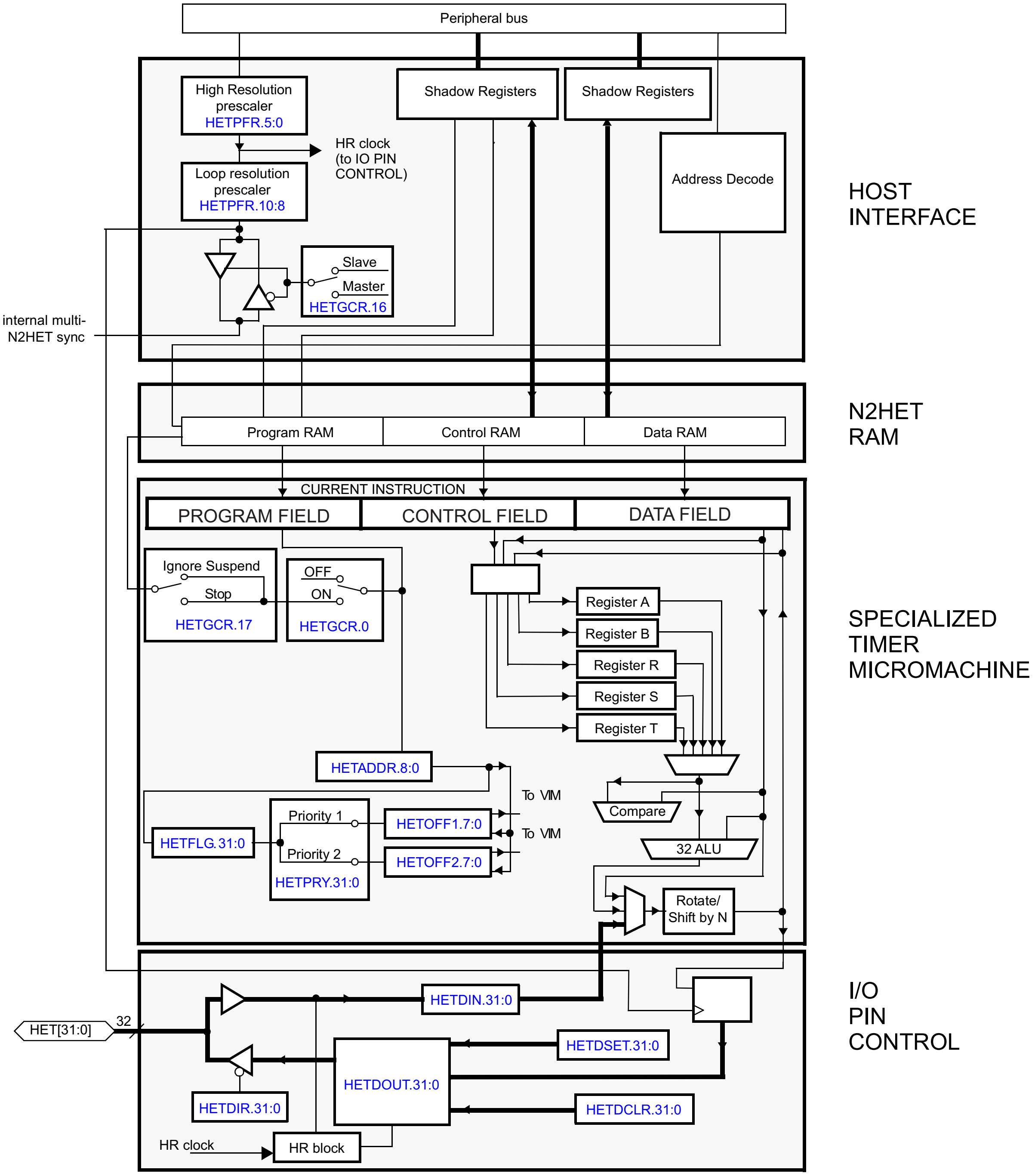
Finally, the modular structure provides maximum flexibility to address a wide range of applications. The timer resolution can be selected from two cascaded prescalers to adjust the loop resolution and HR clocks. The 32 I/O pins can provide any combination of input, period or pulse capture, and output compare, including high resolution for each channel.

23.1.3 Block Diagram

The N2HET module (see Figure 23-1) comprises four separate components:

- Host interface
- N2HET RAM
- Specialized timer micromachine
- I/O control (the N2HET is attached to an I/O port of up to 32 pins)

Figure 23-1. N2HET Block Diagram



23.1.4 Timer Module Structure and Execution

The timer consists of a specialized micromachine that operates a reduced instruction set. Two 25-bit registers and three 32-bit registers are available to manipulate information such as time, event counts, and angle values. System performance is improved by a wide instruction format (96 bits) that allows the N2HET to fetch the instructional operation code and data in one system cycle, thus increasing the speed at which data can be processed. The typical operations performed in the ALU are additions (count), compares, and magnitude compares (higher or same).

Each instruction is made up of a 32-bit program field, a 32-bit control field and a 32-bit data field. The N2HET execution unit fetches the complete 96-bit instruction in one cycle and executes it. All instructions include a 9-bit field for specifying the address of the next instruction to be executed. Some instructions also include a 9-bit conditional address, which is used as the next address whenever a particular condition is true. This makes controlling the flow of an N2HET program inexpensive; in many cases a separate branch instruction is not required.

The interface to the host CPU is based on both communication memory and control registers. The communication memory includes timer instructions (program and data). This memory is typically initialized by the CPU or DMA after reset before the timer starts execution. Once the timer program is loaded into the memory, the CPU starts the timer execution, and typically data parameters are then read or written into the timer memory. The control registers include bits for selecting timer clock, configuring I/O pins, and controlling the timer module.

The programmer implements timer functions by combining instructions in specific sequences. For instance, a single count (CNT) instruction implements a timer. A simple PWM generator can be implemented with a two instruction sequence: CNT and compare (ECMP or MCMP). A complex time function may include many instructions in the sequence. The total timer program is a set of instructions executed sequentially, one after the other. Reaching the end, the program must roll to the first instruction so that it behaves as a loop. The time for a loop to execute is referred to as a *loop resolution clock cycle* or *loop resolution period (LRP).* When the N2HET rolls over to the first instruction, the timer waits for the loop resolution clock to restart the execution of the loop to ensure that only one loop is executed for each loop resolution clock.

The longest path through an N2HET program must be completed within the loop resolution clock (LRP). Otherwise, the program will execute unpredictably because some instructions will not be executed each time through the loop. This effect creates a strong link between the accuracy of the timer functions and the number of functions (the number of instructions) the timer can perform. High resolution (HR) hardware timer extensions are available for each of the N2HET pins to help overcome this limitation.

The high resolution hardware timers operate from the *high resolution clock*, which may be configured for frequency multiples between 2 and 128 times the loop resolution clock frequency. This extending the resolution of timer events and measurements well beyond what is possible with only loop resolution instructions.

Most of the commonly used N2HET instructions can operate either at loop resolution or high resolution; with the restriction that for each pin at most one high resolution instruction can be executed per loop resolution period.

Certain instructions (MOV32, ADM32, ...) can modify the data fields of other instructions. This feature enables the N2HET program to implement double buffering on capture and compare functions. For example, an ECMP compare instruction can be followed by a MOV32 instruction that is conditionally executed when the ECMP instruction matches. The host CPU can update the next compare value by writing asynchronously to the data field of the MOV32 instruction instead of writing directly to the data field of the ECMP instruction. The copy from the buffer (MOV32 data field) to the compare register (ECMP data field) will occur when the MOV32 instruction is actually executed which occurs after the ECMP instruction matches its current compare value. This is the same behavior as one would expect from a double buffered hardware compare register.

Other instructions (MOV64, RADM64) can modify both the control and data fields of other instructions. This allows the N2HET to implement toggle functionality. For example, an ECMP instruction can be followed by a pair of MOV64 instructions. The MOV64 instruction updates the data field of the ECMP instruction to implement the double buffering behavior. But it also updates the control field of the ECMP instruction which allows it to change things like pin action and the conditional address. If one MOV64 instruction configures the ECMP pin action to SET while the second changes it to CLEAR, and the two MOV64 instructions update the conditional address to point to each other, then a single ECMP instruction can be used to toggle a pin each time the compare match occurs.

23.1.5 Performance

Most instructions execute in one cycle, but a few take two or three cycles.

The N2HET can generate many complex output waveforms without CPU interrupts. Where special algorithms are needed following a specific event (for example, missing teeth or a short/long input signal), a minimal number of interrupts to the CPU are needed freeing the CPU for other tasks.

23.1.6 N2HET Compared to NHET

N2HET enhancements from NHET include:

- Eight new instructions: ADD, ADC, SUB, SBB, AND, OR, XOR, RCNT
- Full set of ALU flags Carry (C), Negative (N), Zero (Z), Overflow (V)
- Branch instruction (BR) extended to support signed and unsigned arithmetic comparison conditions
- Two additional 32-bit temporary working registers R, S.
- New HETAND register for AND-Sharing of High Resolution structure between pairs of pins
- Improved high resolution PCNT instruction

23.1.7 NHET and N2HET Compared to HET

Compared to the HET module, the N2HET contains all of the enhancements described in Section 23.1.6 plus the following additional enhancements:

- New Interrupt Enable Set and Clear registers
- Capability to generate requests to the DMA module or the HET Transfer Unit (HTU) including new Request Enable Set and Clear registers
- N2HET RAM parity error detection
- Suppression filters for each of the 32 I/O channel and control register to configure the limiting frequency and counter clock
- Enhanced edge detection hardware that does not rely on the previous bit field in the control word of the N2HET instruction.
- The next, conditional and remote addresses are extended from 8 to 9 bits
- The loop resolution data fields are extended from 20 to 25 bits
- The high resolution data fields are extended from 5 to 7 bits
- Instructions with an adequate condition are able to specify the number of the request line, which triggers either the HET Transfer Unit (HTU) or the DMA module
- The CNT instruction provides a bit, which allows to configure either an equal comparison or a greater or equal comparison when comparing the selected register value with the Max-value
- The MOV32 instruction provides a new bit. If set to one the MOV32 will only perform the move, when the Z-flag is set. If set to zero the MOV32 will perform the move whenever it is executed (independent on the state of the Z-flag)
- There is a new instruction WCAPE, which is a combination of a time stamp and an edge counter
- New Open Drain, Pull Disable, and Pull Select registers

23.1.8 Instructions Features

The N2HET has the following instructions features:

- N2HET uses a RISC-based specialized timer micromachine to carry out a set of 30 instructions
- Instructions are implemented in a Very Long Instruction Word (VLIW) format (96-bits wide)
- The N2HET program execution is self-driven by external or internal events, branching to special routines based on input edges or output compares
- Instructions point to the next instruction executed, eliminating the need for a program counter
- Several instructions can change the program flow based on internal or external conditions

23.1.9 Program Usage

The N2HET instructions/program can be assembled with the N2HET assembler. The assembler generates a C-structure which can be included into the main application program. The application has to copy the content of the structure into the N2HET RAM, set up necessary registers and start the N2HET program execution. In addition to the C-structure, the assembler generates also a header file which makes it easy for the main application to access the different instructions and change for example the duty cycle of a PWM or read out the captured value of a specific signal edge.

23.2 N2HET Functional Description

The N2HET contains RAM into which N2HET code is loaded. The N2HET code is run by the specialized timer micromachine. The host interface and I/O control provide an interface to the CPU and external pins respectively.

23.2.1 Specialized Timer Micromachine

The N2HET has its own instruction set, detailed in Section 23.6.1. The timer micromachine reads each instruction from the N2HET RAM. The program and control fields contain the instructions for how the specialized timer micromachine executes the command. For most instructions, the data field stores the information that needs to be manipulated.

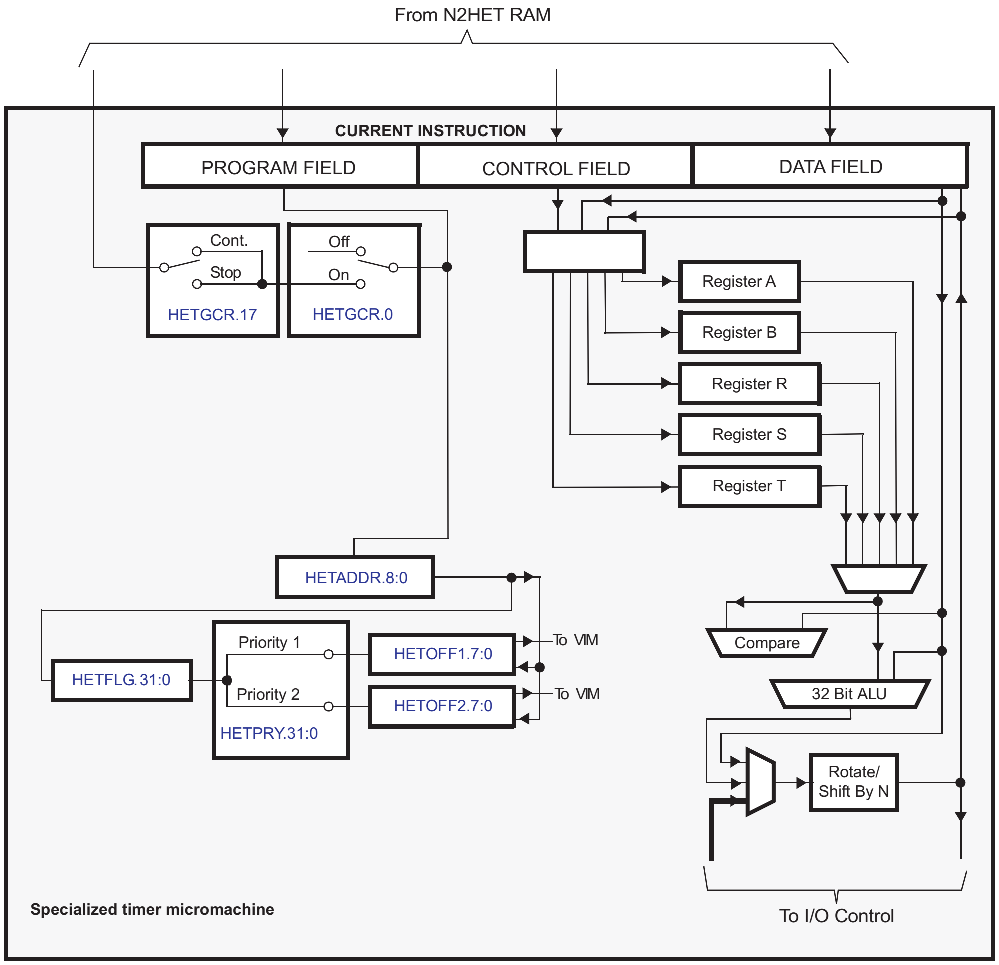
The specialized timer micromachine executes the instructions stored in the N2HET RAM sequentially. The N2HET program execution is self-driven by external or internal events. This means that input edges or output compares may force the program to branch to special routines using a conditional address.

Figure 23-2 shows some of the major operations that the N2HET can carry out, namely compares, captures, angle functions, additions, and shifts. The N2HET contains five registers (A, B, R, S, and T) used to hold compare or counter values and are used by the N2HET instructions. Data may be taken from the registers or the data field for manipulation; likewise, the data may be returned to the registers or the data field.

23.2.1.1 Time Slots and Resolution Loop

Each instruction requires a specific number of cycles or time slots to execute. The resolution specified in the prescaler bitfields determines the timer accuracy. All input captures, event counts, and output compares are executed once in each resolution loop. HR captures and compares are possible (up to N2HET clock accuracy) on the HR I/O pins. For more information about the HR I/O structure, see Section 23.2.5.

Figure 23-2. Specialized Timer Micromachine



23.2.1.2 Program Loop Time

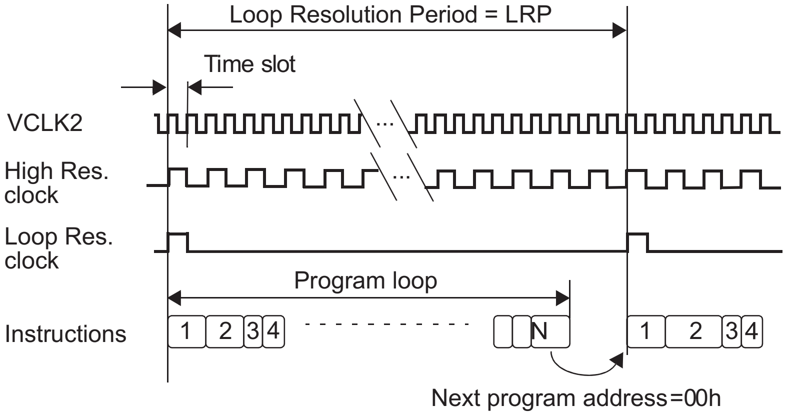
The program loop time is the sum of all cycles used for instruction execution. This time may vary from one loop to another if the N2HET program includes conditionally executed instructions.

The timer program restarts on every resolution loop. The start address is fixed at N2HET RAM address 00h. The longest path through a program must fit within one loop resolution period to guarantee complete accuracy.

The last instruction of a program must branch back to the fixed start address (next program address = 00h). When an N2HET program branches back to address 00h before the end of a loop resolution period, the N2HET detects this and pauses instruction execution until the beginning of the next loop resolution period.

The timing diagram in Figure 23-3 illustrates the program flow execution.

Figure 23-3. Program Flow Timings



23.2.1.3 Instruction Execution Sequence

The execution of a N2HET program begins with the first occurrence of the loop resolution clock, after the N2HET is turned on. At the first and subsequent occurrences of the loop resolution, the instruction at location address 00h is prefetched. The program execution begins at the occurrence of the loop resolution clock and continues executing the instructions until the program branches to 00h location. The instruction is prefetched at location 00h and execution flag is reset. The N2HET pauses instruction execution until the occurrence of the loop resolution clock and resumes normal execution.

N2HET programs must be written so that they complete execution and return to address 00h before the occurrence of the next loop resolution clock. If the N2HET program exceeds this execution time limit, then a program overflow condition occurs as described in Section 23.2.1.4.

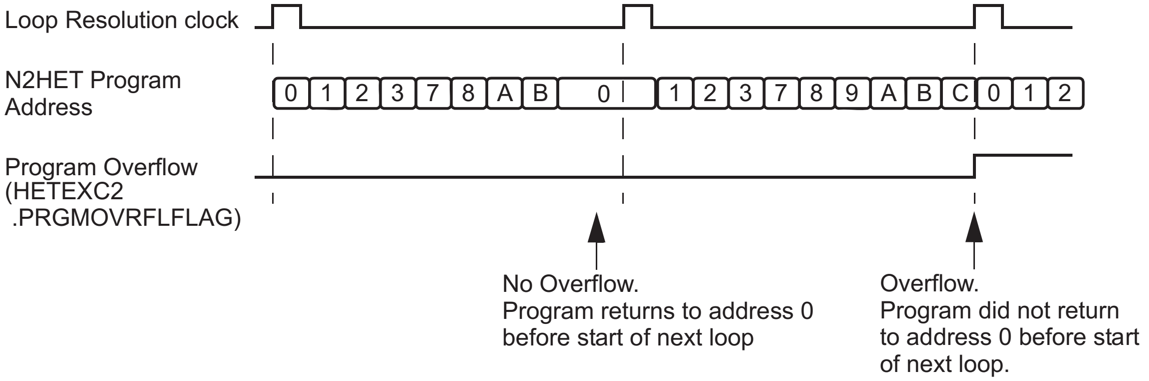
23.2.1.4 Program Overflow Condition

If the number of time slots used in a program loop exceeds the number available time slots in one loop resolution, the timer sets the program overflow interrupt flag located in the HETEXC2 register. To maintain synchronization of the I/Os, this condition should never be allowed to occur in a normal operation. The HETEXC2.PRGMOVRFLFLAG flag provides a mechanism for checking that the condition does not occur during the debug and validation phases.

As Figure 23-4 illustrates, when a program overflow occurs, the currently executing N2HET program sequence is interrupted and restarted at N2HETaddress 0 for the beginning of the next loop resolution clock period. Also, HETEXC2.PRGMOVRFLFLAG is set.

If the instruction that caused the overflow (instruction at address 0xC in Figure 23-4) has any pin actions selected, these pin actions will not be performed. However other actions of the instruction including register and RAM updates will still be performed.

Figure 23-4. Use of the Overflow Interrupt Flag (HETEXC2)


Loop Resolution clock
N2HET Program Address
0 1 2 3 7 8 A B 0 1 2 3 7 8 9 A B C 0 1 2
Program Overflow (HETEXC2 .PRGMOVRFLFLAG)
No Overflow. Program returns to address 0 before start of next loop
Overflow. Program did not return to address 0 before start of next loop.

23.2.1.5 Architectural Restrictions on N2HET Programs

Certain architectural restrictions apply to N2HET programs:

1. The size of an N2HET program must be greater than one instruction.
2. An extra wait state is incurred by any instruction that modifies a field in the next instruction to be executed.
3. Only one instruction (using high resolution) is allowed per high resolution pin.
4. Consecutive break points are not supported. Instructions with break points must have at least a distance of two instructions (for example, at addresses 1, 3, 5, 7, and so on, assuming the program executes linearly)

NOTE: While it would be unusual to code an N2HET program that is only one instruction long, it is trivial to modify such a program to meet the requirement of restriction 1. Simply add a second instruction to the program, which may be a simple branch to zero.

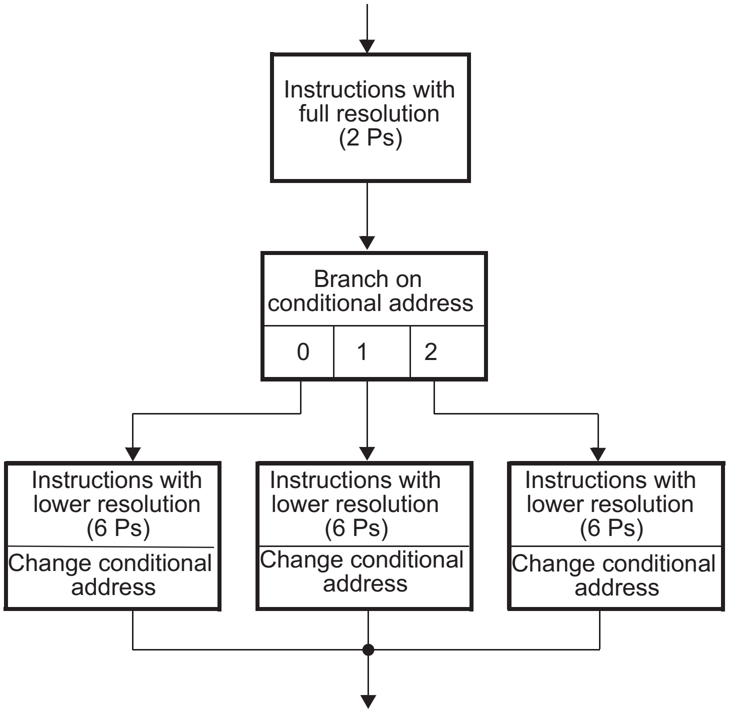
To enforce restriction 3, the high resolution pin structures respond only to the first instruction that is executed matching their pin number with hr_lr=HIGH, regardless of whether or not the en_pin_action field is ON. Subsequent instructions are ignored by the high resolution pin structure for the remainder of the loop resolution period.

23.2.1.6 Multi-Resolution Scheme

The N2HET has the capability to virtually extend the counter width by executing instructions only once every N loop resolution periods. This decreases the timer resolution, but extends the counter range which may be useful when generating or measuring slow signals. Figure 23-5 illustrates how a multi-resolution scheme may be implemented in an N2HET program. An unconditional Branch instruction and an index sequence, using a MOV64 instruction in each low resolution loop, is required to control this particular program flow.

NOTE: HR instructions must be placed in the main (full resolution) loop to ensure proper operation.

Figure 23-5. Multi-Resolution Operation Flow Example


Instructions with full resolution (2 Ps)
Branch on conditional address
0
1
2
Instructions with lower resolution (6 Ps)
Change conditional address
Instructions with lower resolution (6 Ps)
Change conditional address
Instructions with lower resolution (6 Ps)
Change conditional address

23.2.1.7 Debug Capability

The N2HET supports breakpoints to allow you to more easily debug your N2HET program. Figure 23-6 provides an illustration of the breakpoint mechanism.

The steps to enable an N2HET breakpoint are:

1. Make sure the device nTRST pin is high, since N2HET breakpoints are disabled whenever this pin is low. (Normally this is handled automatically when a JTAG debugger is attached).
2. Attach a JTAG debugger and connect to the device that has been already programmed with the N2HET code that needs to debugged. (downloading to on-chip flash is outside the scope of this section).
3. Execute the CPU program at least until the point where the N2HET program RAM has been initialized by the CPU.
4. Open a memory window in the N2HET registers.
5. Make sure HETEXC2.DEBUGSTATUSFLAG bit is cleared.
6. Open a memory window on the N2HET RAM
7. Set bit 22 in the program field of the instruction(s) on which you wish to break. Note that this instruction will be executed **before** the N2HET is halted - slightly different from how CPU breakpoints behave.
8. Make sure the CPU and N2HET are running, if they are halted then restart the CPU through the JTAG emulator (N2HET will start when the CPU starts).
9. Both the CPU and N2HET will halt when breakpoint is reached.

When the N2HET is halted, its state machines are frozen but all of the N2HET control registers can be accessed through the JTAG emulator interface.

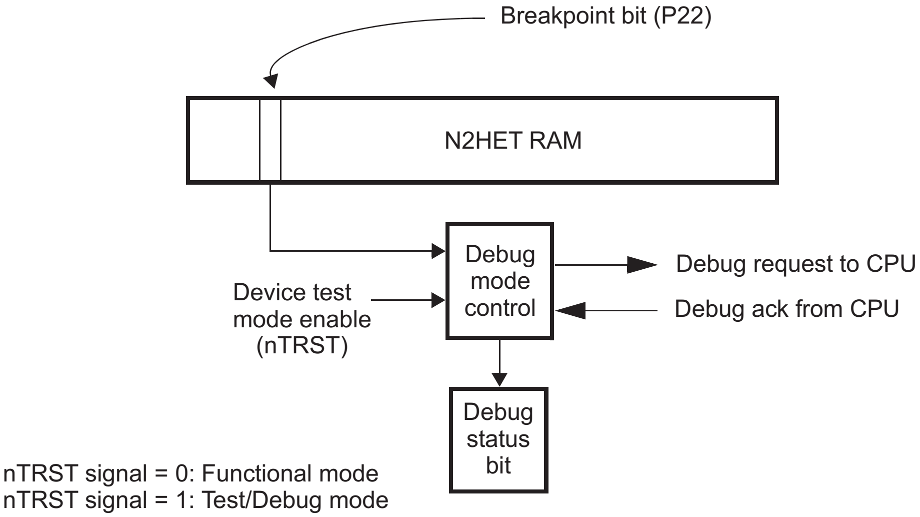
The current N2HET instruction address can be inspected by reading the HETADDR register; this should be pointing to the instruction that caused the breakpoint.

The N2HET internal working registers (A,B,R,S,T) are not directly visible through the JTAG emulator interface. If the content of these registers needs to be inspected, it is best to add an instruction like MOV32 which copies the register value to the N2HET RAM. This RAM location can be inspected when the N2HET halts.

To restart execution of both the CPU and the N2HET from the halted state:

1. Clear HETEXC2.DEBUGSTATUSFLAG bit.
2. Clear bit 22 in the program field of the instruction on which the breakpoint was reached.
3. Restart the CPU through the normal JTAG emulator procedure (‘Run’ or ‘Go’). The N2HET will automatically start executing when it sees that the CPU has exited the debug state.

Figure 23-6. Debug Control Configuration



NOTE: Consecutive break points are not supported. Instructions with break points must have at least a distance of two instructions (for example, at N2HET addresses 1, 3, 5, 7, and so on)

23.2.2 N2HET RAM Organization

The N2HET RAM is organized into two sections. The first contains the N2HET program itself. The second contains parity protection bits for the N2HET program.

Each N2HET instruction is 96-bits wide but aligned to a 128-bit boundary. Instructions consist of three 32-bit fields: Program, Control, and Data. Instructions are separated by a fourth unimplemented address to force alignment to 128-bit boundaries.

The integrity of the N2HET program can be protected by Parity. Parity protection is enabled through the N2HET Parity Control Register (HETPCR).

Table 23-1 shows the base addresses for N2HET RAM and N2HET Parity RAM.

Table 23-1. N2HET RAM Base Addresses

N2HET1 Base Address	N2HET2 Base Address	Memory
0xFF46_0000	0xFF44_0000	N2HET Instruction RAM (Program/Control/Data)
0xFF46_2000	0xFF44_2000	N2HET Parity RAM

23.2.2.1 N2HET RAM Banking

Because the CPU must make updates to the N2HET RAM while the N2HET is executing, for example to update the duty cycle value of a PWM, it is important to understand how the N2HET RAM organization facilitates simultaneous accesses by both the HOST CPU and the N2HET.

The N2HET RAM is implemented as 4 banks of 96-bit wide two port RAM. This means that there a total of 8 ports available; four read and four write. Normally the N2HET will use up to two of these ports at a time. One read port is used to allow the N2HET to prefetch the next N2HET instruction while a write port may be used to update the data or control fields that have changed as a result of executing the current instruction.

N2HET accesses to its own internal RAM are given priority over accesses from an external host (CPU or DMA), this makes N2HET program execution deterministic which is a critical requirement for a timer.

Most N2HET instructions execute in a single cycle. Cases where a wait state impacts the N2HET program execution time are:

- The current N2HET instruction writes data back to the next N2HET in the execution sequence.
- The external host reads from an N2HET instruction where the automatic read-clear option is set, while the N2HET is executing from/on the same address (See Section 23.2.4.3).

Except for the case of automatic read-clear, the external host is stalled when the host and N2HET have a bank conflict. However this will typically only result in a stall of one cycle, due to the N2HET bank ordering which is organized on the N2HET Address least significant bit boundaries (See Table 23-2).

Assuming most of the N2HET program executes linearly through the N2HET Address space; if a bank conflict does exist it is usually resolved in the next cycle as the N2HET program moves to the next bank. N2HET programmers should avoid writing a program that accesses the same bank of N2HET RAM on every cycle, as this could lock the external host out of the N2HET memory completely.

Table 23-2 describes the N2HET memory map, as viewed by the N2HET as well as from the memory space of the host CPU and DMA.

Table 23-2. N2HET RAM Bank Structure

N2HET Address	Host CPU or DMA Address Space				
Instruction	Program Field Address	Control Field Address	Data Field Address	Reserved Address	N2HET RAM Bank
000h	XX0000h	XX0004h	XX0008h	XX000Ch	A
001h	XX0010h	XX0014h	XX0018h	XX001Ch	B
002h	XX0020h	XX0024h	XX0028h	XX002Ch	C
003h	XX0030h	XX0034h	XX0038h	XX003Ch	D
004h	XX0040h	XX0044h	XX0048h	XX004Ch	A
:	:	:	:	:	:
03Fh	XX03F0h	XX03F4h	XX03F8h	XX03FCh	D
040h	XX0400h	XX0404h	XX0408h	XX040Ch	A
:	:	:	:	:	:
1FFh	XX1FF0h	XX1FF4h	XX1FF8h	XX1FFCh	D

NOTE: The external host interface supports any access size for reads, but only 32-bit writes to the N2HET RAM are supported. Reserved addresses should not be accessed, the result of doing so is indeterminate.

23.2.2.2 Parity Checking

The N2HET module can detect parity errors in N2HET RAM. As described in Section 23.2.2 the N2HET allows 32-bit writes only. Therefore N2HET RAM parity checking is implemented using one parity bit per 32-bit field in N2HET RAM.

Even or odd parity selection for N2HET parity detection can be configured in the system module. Parity calculation and checking can be enabled/disabled by a 4-bit key in HETPCR.

During a read access to the N2HET RAM, the parity is calculated based on the data read from the RAM and compared with the good parity value stored in the parity bits. The parity check is performed when the N2HET execution unit makes a read access to N2HET RAM, but also when a different master (for example, CPU, HTU, DMA) performs the read access. If any 32-bit-word fails the parity check then an error is signaled to the ESM module. The N2HET address, which generated the error is detected and is captured in HETPAR for host system debugging. The address is frozen from being updated until it is read by the bus master.

The N2HET execution unit reads the instructions, which are 96-bit wide. They contain the program-, control- and data-field whereby each is 32-bit wide. So when fetching N2HET instructions parity checking is performed on three words in parallel.

If a parity error is detected in two or more words in the same cycle then only one address (word at the lower address) is captured. The captured N2HET address is always aligned to a 32-bit word boundary.

During debug, parity checking is still performed on accesses originating from the on-chip host CPU and DMA. However, parity errors that are detected during an access initiated by the debugger itself are ignored.

23.2.2.3 Parity Error Detection Actions

Detection of a N2HET parity error causes the following actions:

1. An error is signaled to the ESM module.
2. The Parity Address Register (HETPAR) is loaded with the address of the faulty N2HET field.
3. N2HET execution immediately stops. (The instruction that triggered the parity error is not executed.)
4. The Turn-On/Off-Bit in the N2HET Global Configuration Register (HETGCR) is automatically cleared.
5. All N2HET internal flags are cleared.
6. All N2HET pins selected by N2HET Parity Pin Register (HETPPR) enter a predefined safe state.
7. Register HETDOUT is also updated to reflect changes in pin state due to HETPPR.

The safe state for N2HET pins selected through the HETPPR register depends on how the pin is configured in the HETDIR, HETPDR, and HETPSL registers. Table 23-3 explains how the safe state is determined.

Table 23-3. Pin Safe State Upon Parity Error Detection

Safe State	HETDIR	HETPDR	HETPSL
Drive Low	1	0	0
Drive High	1	0	1
High Impedance	1	1	x

23.2.2.4 Testing Parity Detection Logic

To test the parity detection logic, the parity RAM has to be made accessible to the CPU in order to allow a diagnostic program to insert parity errors. The control register bit HETPCR.TEST must be set in order to make the parity RAM accessible. Once HETPCR.TEST is set, the parity bits are accessible as described in Table 23-4.

Each 32-bit N2HET field has its own parity bit in the N2HET Parity RAM as shown in Table 23-4. There are no parity bits for the reserved fields, since there is no physical N2HET RAM for these fields.

Table 23-4. N2HET Parity Bit Mapping

Address N2HET1	Address N2HET2	Bits	
		[31:1]	**[0]**
0xFF46_2000	0xFF44_2000	Reads 0, Writes have no effect	Instruction 0 Program Field Parity Bit
0xFF46_2004	0xFF44_2004	Reads 0, Writes have no effect	Instruction 0 Control Field Parity Bit
0xFF46_2008	0xFF44_2008	Reads 0, Writes have no effect	Instruction 0 Data Field Parity Bit
0xFF46_200C	0xFF44_200C	Reads 0, Writes have no effect	Read 0
0xFF46_2010	0xFF44_2010	Reads 0, Writes have no effect	Instruction 1 Program Field Parity Bit
....	....	...	...

23.2.2.5 Initialization of Parity RAM

After device power up, the N2HET RAM contents including the parity bits cannot be guaranteed. In order to avoid false parity failures due to the random state in which RAM powers up, the RAM has to be initialized.

Before initializing the N2HET RAM, enable the N2HET parity logic by writing to HETPCR. Then the N2HET Instruction RAM should be initialized. With parity enabled, the N2HET parity RAM will be initialized automatically by N2HET at the same time that the N2HET instruction RAM is initialized by the CPU. Note that loading the N2HET program with parity enabled is also effective.

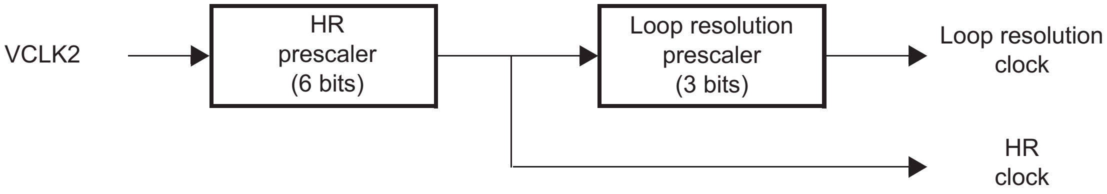
Another possibility to initialize the N2HET memory and its parity bits is, to use the system module to start the automatic initialization of all RAMs on the microcontroller. The RAMs will be initialized to ‘0’. Depending on the even/odd parity selection, the parity bit will be calculated accordingly.

23.2.3 Time Base

All N2HET timings are derived from VCLK2 (see Figure 23-7). Internally N2HET instructions execute at the VCLK2 rate; but the timer loop clock and the high-resolution hardware timer clock can be scaled down from VCLK2. Two prescalers are available to adjust the timer loop resolution clock for the program loop, and the high resolution (HR) clock for the HR I/O counters.

- **Time Slots:** The number of cycles available for instruction execution per loop. Time Slots is the number of VCLK2 cycles in a Loop Resolution Clock.
- **High Resolution Clock:** The high resolution clock is the smallest time increment with which a pin can change it’s state or can be measured in the case of input signals. A 6-bit prescaler dividing VCLK2 by a user-defined HR prescale divide rate (hr) stored in the 6-bit HR prescale factor code (HETPFR). See Table 23-5.
- **Loop Resolution Clock:** The loop resolution clock defines the timebase for executing all instructions in a N2HET program. Since instructions can be conditionally executed, the longest path through the N2HET program must fit into one loop resolution clock period (LRP).A 3-bit prescaler dividing the HR clock by a user-defined loop-resolution prescale divide rate (lr) stored in the 3-bit loop-resolution prescale factor code (HETPFR). See Table 23-5.

Figure 23-7. Prescaler Configuration



The following abbreviations and relations are used in this document:

1. hr: high resolution prescale factor (1, 2, 3, 4,..., 63, 64)
2. lr: loop resolution prescale factor (1, 2, 4, 8, 16, 32, 64,128)
3. ts: Time slots (cycles) available for instruction execution per loop. ts = hr x lr
4. HRP = high resolution clock period HRP = hr × T_{VCLK2} (ns)
5. LRP = loop resolution clock period LRP = lr × HRP (ns)

The loop resolution period (LRP) must be selected to be larger than the number of Time slots (VCLK2 cycles) required to complete the worst-case execution path through the N2HET program. Otherwise a program overflow condition may occur (see Section 23.2.1.4). Because of the relationship of time slots to the hr and lr prescalers as described in item 3 above, increasing either hr or lr increases the number of time slots available for program execution. However, lr would typically be increased first, since increasing hr results in a decrease in timer resolution since it reduces the clock to the High Resolution IO structures.

The divide rates hr and lr can be defined in the HETPFR register. Table 23-5 lists the bit field encodings for the prescale options.

Table 23-5. Prescale Factor Register Encoding

LRPFC - Loop Resolution		HRPFC - High Resolution	
HETPFR[10:8]	Prescale Factor lr	HETPFR[5:0]	Prescale Factor hr
000	/1	000000	/1
001	/2	000001	/2
010	/4	000010	/3
011	/8	000011	/4
100	/16	:	:
101	/32	111101	/62
110	/64	111110	/63
111	/128	111111	/64

23.2.3.1 Determining Loop Resolution

As an example, consider an application that requires high resolution of HRP = 62.5 ns, and loop resolution of LRP = 8 µs, and needs at least 250 time slots for the N2HET application program.

Assuming VCLK2 = 32 MHz, the following shows which divide-by rates and which value in the Prescale Factor Register (HETPFR) is required for the above requirements:

$$\mathrm{hr} = 2 \rightarrow \mathrm{HRP} = \frac{\mathrm{hr}}{\mathrm{VCLK2}} = \frac{2}{32\mathrm{MHz}} = 62.5\mathrm{ns}$$

$$\mathrm{lr} = 128 \rightarrow \mathrm{lr} \times \mathrm{HRP} = 128 \times 62.5\mathrm{ns} = 8\ \mu\mathrm{s}$$

$$\mathrm{ts} = \mathrm{hr} \times \mathrm{lr} = 2 \times 128 = 256$$

$$\mathrm{hr} = 2,\ \mathrm{lr} = 128 \rightarrow \mathrm{HETPFR[31:0]} = 0\mathrm{x}00000701 \tag{33}$$

In the example above, if the loop resolution period needs to decrease from 8 µs to 4 µs, then only 128 time slots will be available for program execution. The program may need to be restructured as suggested in Section 23.2.1.6.

23.2.3.2 The 7-Bit HR Data Field

The instruction execution examples of ECMP (Section 23.2.5.9), MCMP (Section 23.2.5.10), PCNT (Section 23.2.5.12), PWCNT (Section 23.2.5.11), and WCAP (Section 23.2.5.13) show that the 7-bit HR data field can generate or measure high resolution delays (HR delay) relative to the start of an LRP within one N2HET loop LRP. The last section showed that:

LRP = lr × HRP

There are lr high resolution clock periods (HRP) within the N2HET loop resolution clock period (LRP). If lr = 128 then the HR delay can range from 0 to127 HRP clocks within LRP and all 7 bits of the HR data field are needed. Instead of being limited to measuring and triggering events based on the loop resolution clock period (LRP) the HR extension allows measurements and events to be described in terms fractions of an LRP (down to 1/128 of an LRP). The only limitation is that a maximum of one HR delay can be specified per pin during each loop resolution period.

Table 23-6 shows which bits of the HR data field are not used by the high resolution IO structures if lr is less than 128. In this case the non-relevant bits (LSBs) of the HR data fields will be one of the following:

- Written as 0 for HR capture (for PCNT, WCAP)
- Or interpreted as 0 for HR compare (for ECMP, MCMP. PWCNT)

Table 23-6. Interpretation of the 7-Bit HR Data Field

Loop Resolution Prescale divide rate (lr)	Bits of the HR data field [1]							HRP Cycles delay range
	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]	
1	X X X X X X X							0
2	1/2	X X X X X X						0 to 1
4	1/2	1/4	X X X X X					0 to 3
8	1/2	1/4	1/8	X X X X				0 to 7
16	1/2	1/4	1/8	1/16	X X X			0 to 15
32	1/2	1/4	1/8	1/16	1/32	X X		0 to 31
64	1/2	1/4	1/8	1/16	1/32	1/64	X	0 to 63
128	1/2	1/4	1/8	1/16	1/32	1/64	1/128	0 to 127

[1] X = Non-relevant bit (treated as '0')

23.2.3.2.1 *Example:*

Prescale Factor Register (HETPFR) = 0x0300

—> lr = 8 —> LRP = 8 × HRP

Assumption: HR data field = 0x50 = 1010000b

lr = 8 —> Bits D[3:0] are ignored —> HR delay = 101b = 5 HRPs

or by using the calculation with weight factors:

HR Delay
= lr · (D[6] · 1/2 + D[5] · 1/4 + D[4] · 1/8 + D[3] · 1/16 + D[2] · 1/32 + D[1] · 1/64 + D[0] · 1/128)
= 8 · (1 · 1/2 + 0 · 1/4 + 1 · 1/8 + 0 · 1/16 + 0 · 1/32 + 0 · 1/64 + 0 · 1/128)
= 5 HRPs

23.2.4 Host Interface

The host interface controls all communications between timer-RAM and masters accessing the N2HET RAM. It includes following components:

23.2.4.1 Host Accesses to N2HET RAM

The host interface supports the following types of accesses to N2HET RAM:

- Read accesses of 8, 16, or 32 bits
- Read accesses of 64-bits that follow the shadow register sequence described in Section 23.2.4.2.
- Write accesses of 32 bits

Writes of 8 or 16 bits to N2HET RAM by an external host are not supported.

23.2.4.2 64-bit Read Access

The consecutive read of a control field CF(n) and a data field DF(n) of the same instruction (n) performed by the same master (for example, CPU, DMA, or any other master) is always done as a simultaneous 64-bit read access. This means that at the same time CF(n) is read, DF(n) is loaded in a shadow register. So the second access will read DF(n) from the shadow register instead of the N2HET RAM.

In general a 64-bit read access of one master could be interrupted by a 64-bit read access of another master. A total of three shadow registers are available. Therefore up to three masters can perform 64-bit reads in an interleaved manner (Master1 CF, Master2 CF, Master3 CF, Master1 DF, Master2 DF, Master3 DF).

If all three shadow registers are activated and a 4th master performs a CF or DF read it will result in an address error and the RAM access will not happen. Other access types by a fourth master (reads from the PF field or writes to any of the fields) will occur because these access types do not require an available shadow register resource to complete.

23.2.4.3 Automatic Read Clear Feature

The N2HET provides a feature allowing to automatically clear the data field immediately after the data field is read by the external host CPU (or DMA). This feature is implemented via the *control bit*, which is located in the control field (bit C26). This is a static bit that can be used by any instruction, and specified in the N2HET program by adding the option (control = ON) to the N2HET instruction. The automatic read clear feature works for both 32 and 64 bit reads that follow the sequence described in Section 23.2.4.2.

When the host CPU reads the data field of that instruction, the current data value is returned to the host CPU but the field is cleared automatically as a side effect of the read. In case the master reads data from an instruction currently executing, any new capture result is stored and this takes priority over the automatic read clear feature, so that the new capture result is not lost.

As an example of where the automatic read clear feature is useful, consider the PCNT instruction. If this instruction is configured for automatic read clear, then when the host CPU reads the PCNT data field it will be cleared automatically. The host CPU can then poll the PCNT data field again, and as long as the field returns a value of zero the host CPU program knows a new capture event has not occurred. If the data field were not cleared, it would be impossible for the host CPU to determine whether the data field holds data from the previous capture event, or if it happens to be data from a new capture event with the same value.

23.2.4.4 Emulation Mode

Emulation mode, used by the software debugger, is specified in the global configuration register. When the host CPU debugger hits a breakpoint, the CPU sends a suspend signal to the modules. Two modes of operation are provided: suspend and ignore suspend.

- Suspend

When a suspend is issued, the timer operation stops at the end of the current timer instruction. However, the CPU accesses to the timer RAM or control registers are freely executed.

- Ignore suspend

The timer RAM ignores the suspend signal and operates real time as normal.

23.2.4.5 Power-Down

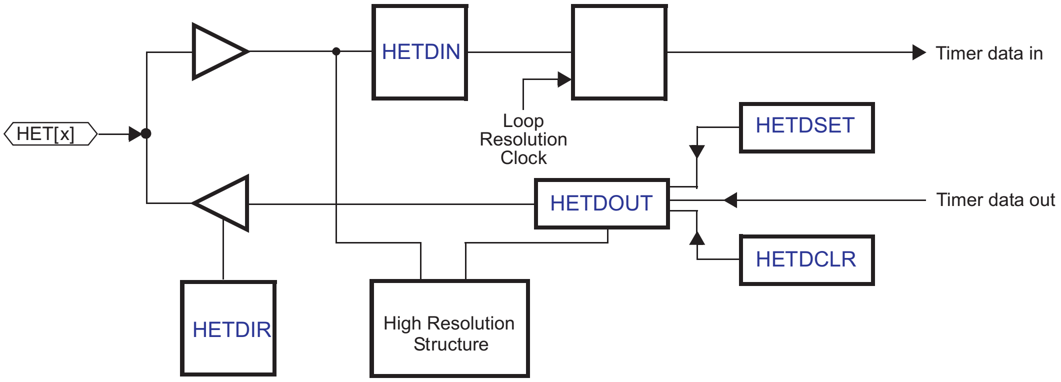
After setting the turn-off bit in the Global Configuration Register (HETGCR), it is required to delay until the end of the timer program loop before putting the N2HET in power-down mode. This can be done by waiting until the N2HET Current Address (HETADDR) becomes zero, before disabling the N2HET clock source in the device's Global Clock Module (GCM).

23.2.5 I/O Control

The N2HET has up to 32 pins. Refer to device specific data sheets for information concerning the number of N2HETIO available. All of the N2HET pins available are programmable as either inputs or outputs.

These 32 I/Os have an identical structure connected to pins HET[31] to HET[0]. See Figure 23-8 for an illustration of the I/O control. In addition all 32 I/Os have a special HR structure based on the HR clock. This structure allows any N2HET instruction to use any of these I/Os with an accuracy of either loop resolution or high resolution accuracy.

Figure 23-8. I/O Control



Pins N2HET [31] to N2HET [0] can be used by the CPU as general-purpose inputs or outputs using the N2HET Data Input Register (HETDIN) for reading and N2HET Data Output Register (HETDOUT), N2HET Data Set Register (HETDSET) or N2HET Data Clear Register (HETDCLR) for writing, depending on the type of action to perform. The N2HET pins used as general-purpose inputs are sampled on each VCLK2 period.

23.2.5.1 Using General-Purpose I/O Data Set and Clear Registers

The N2HET Data Clear Register (HETDCLR) and N2HET Data Set Register (HETDSET) can be used to minimize the number of accesses to the peripheral to modify the output register and output pins. When the application needs to set or to reset some N2HET pins without changing the value of the others pins, the first possibility is to read N2HET Data Output Register (HETDOUT), modify the content (AND, OR, and so on), and write the result into N2HET Data Output Register (HETDOUT). However, this read-modify-write sequence could be interrupted by a different function modifying the same register which will result in a data coherency problem.

Using the N2HET Data Set Register (HETDSET) or N2HET Data Clear Register (HETDCLR), the application program must write the mask value (same mask value for the first option) to the register to set or reset the desired pins. Any bits written as 0 to HETDSET and HETDCLR are left unchanged, which avoids the possible coherency problem of the read-modify-write approach.

```
Coding Example (C program): Set pins using the 2 methods.

unsigned int MASK;                              /* Variable that content the bit mask  */
volatile unsigned int *HETDOUT,*HETDSET;        /* Pointer to HET registers            */
...
*HETDOUT = *HETDOUT | MASK;                     /* Read-modify-write of HETDOUT        */
*HETDSET = MASK;                                /* Set the pin without reading HETDOUT */
```

23.2.5.2 Loop Resolution Structure

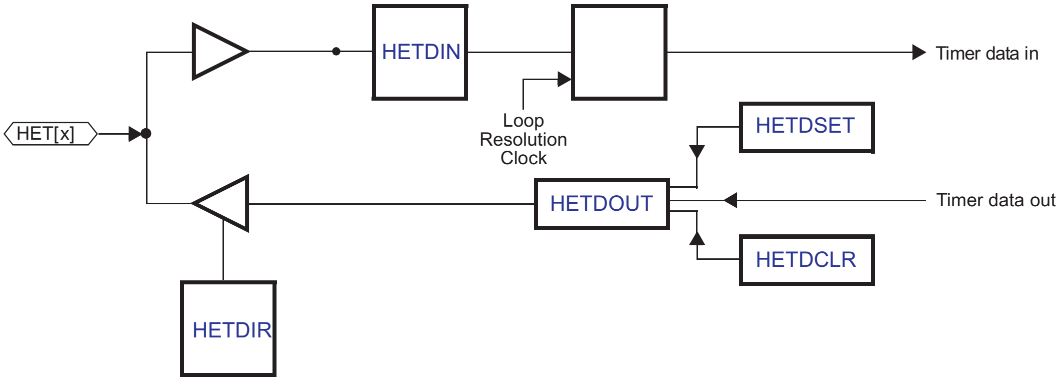
The N2HET uses the pins N2HET [31:0] as input and/or output by the way of the instruction set. Actually, each pin could monitor the N2HET program or could be monitored by the N2HET program. By using the I/O register of the N2HET, the CPU is able to interact with the N2HET program flow.

When an action (set or reset) is taken on a pin by the N2HET program, the N2HET will modify the pin at the rising edge of the next loop resolution clock.

When an event occurs on a N2HET I/O pin, it is taken into account at the next rising edge of the loop resolution clock.

The structure of each pin is shown in Figure 23-9.

Figure 23-9. N2HET Loop Resolution Structure for Each Bit

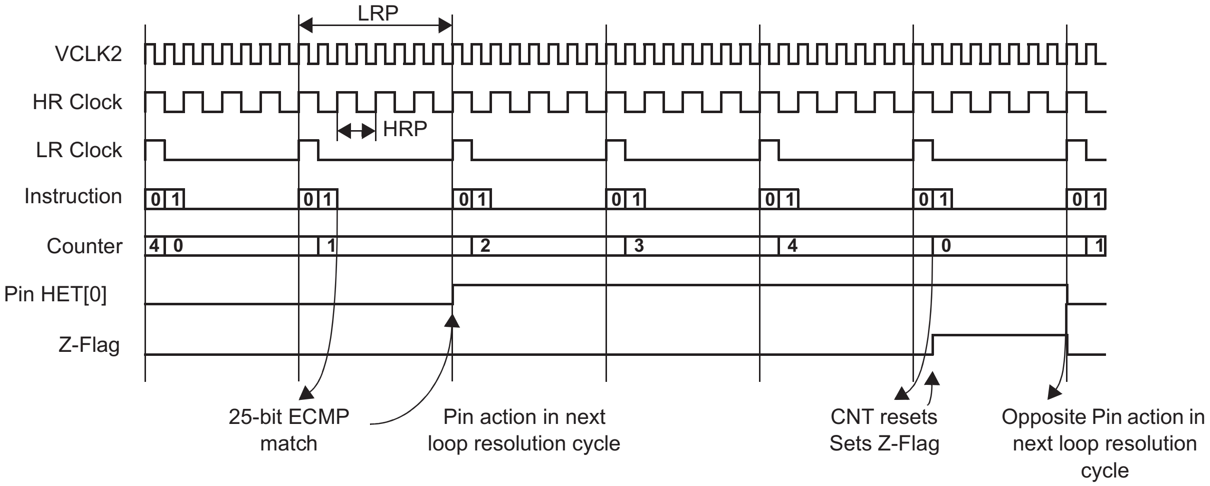


The example in Figure 23-10 shows a simple PWM generation with loop resolution accuracy. The corresponding program is:

```
HETPFR[31:0] register = 0x201 --> lr=4 and hr=2 --> ts = 8

N2HET Program:

L00  CNT   { next= L01, reg=A, irq=OFF, max = 4 }
L01  ECMP  { next= L00, cond_addr= L00, hr_lr=LOW, en_pin_action=ON, pin=0,
             action=PULSEHI, reg=A, irq=OFF, data= 1, hr_data = 0x0 }

; 25 bit compare value is 1 and the 7-bit HR compare value is 0
```

The CNT and ECMP instructions are executed once each loop resolution cycle. When the CNT instruction is executed, the specified register (A) and the CNT instruction data field are both incremented by one. Next the ECMP is executed and the data field of the ECMP is compared with the specified register (A). If both values match, then the pin action (PULSEHI in this case) will be performed in the next loop resolution cycle. The CNT continues incrementing each loop resolution cycle. When the data field overflows (max + 1), then the Z-flag is set by the CNT instruction. In the next loop resolution cycle, the Z-flag is evaluated and the opposite pin action is performed if it is set. The Z-flag will only be active for one loop resolution cycle.

Figure 23-10. Loop Resolution Instruction Execution Example



23.2.5.3 High Resolution Structure

All 32 I/Os provide the HR structure based on the HR clock. The HR clock frequency is programmed through the Prescale Factor Register (HETPFR). In addition to the standard I/O structure, all pins have HR hardware so that these pins can be used as HR input captures (using the HR instructions PCNT or WCAP) or HR output compares (using the HR instructions ECMP, MCMP, or PWCNT).

All five HR instructions (PCNT, WCAP, ECMP, MCMP, and PWCNT) have a dedicated hr_lr bit (high resolution/low resolution; program field bit 8) allowing operation either in HR mode or in standard resolution mode by ignoring the HR field. By default, the hr_lr bit value is 0 which implies HR operation mode. However, setting this bit to one allows the use of several HR instructions on a single HR pin. Only one instruction is allowed to operate in HR mode (bit cleared to 0), but the other instructions can be used in standard resolution mode (bit set to 1).

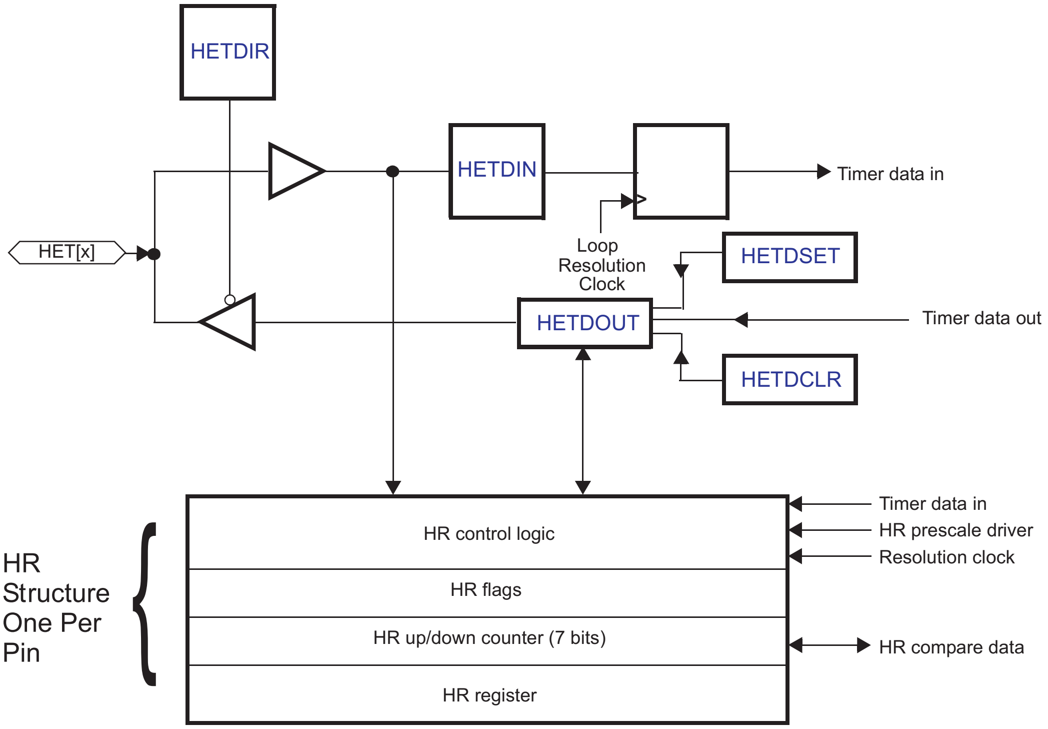
23.2.5.4 HR Block Diagram

Each time an HR instruction is executed on a given pin, the HR structure for that pin is programmed and synchronized to the next loop-resolution cycle (which HR function to perform and on which edges it should take an action) with the information given by the instruction. The HR structure for each pin decodes the pin select field of the instruction and programs its HR structure if it matches.

NOTE: For each N2HET pin, only one instruction specifying a high resolution operation (hr_lr = HIGH) is allowed to execute per loop resolution period. This includes any instructions where (hr_lr = HIGH) but (en_pin_action = OFF).

The first high resolution instruction that executes and specifies a particular pin locks out subsequent high resolution instructions from operating on the same pin until the end fo the current loop resolution period.

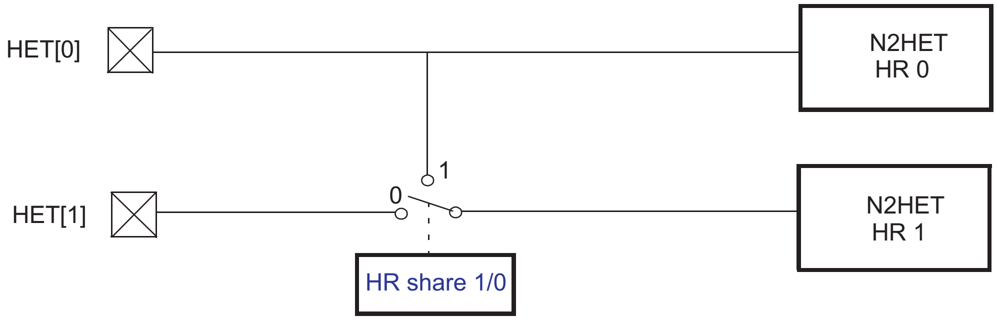
Figure 23-11. HR I/O Architecture



23.2.5.5 HR Structures Sharing (Input)

The HR Share Control Register (HETHRSH) allows two HR structures to share the same pin **for input capture only**. If these bits are set, the HR structures N and N+1 are connected to pin N. In this structure, pin N+1 remains available for general-purpose input/output. See Figure 23-12.

Figure 23-12. Example of HR Structure Sharing for N2HET Pins 0/1



The following program gives an example how the HR share feature (HET[0] HR structure and HET[1] HR structure shared) can be used for the PCNT instruction:

```
L00 PCNT { next=L01, type=rise2fall, pin=0 }
L01 PCNT { next=L00, type=fall2rise, pin=1 }
```

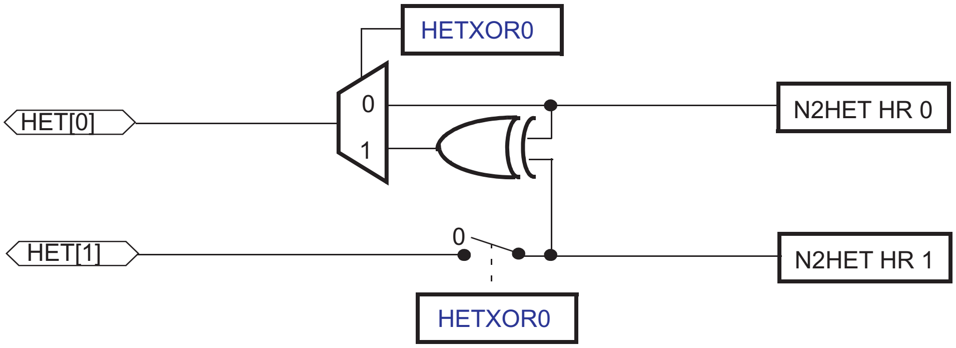
The HET[1] HR structure is also connected to the HET[0] pin. The L00_PCNT data field is able to capture a high pulse and the L01_PCNT captures a low pulse on the **same** pin (N2HET [0] pin).

23.2.5.6 AND / XOR-shared HR Structure (Output)

Usually the N2HET design allows only one HR structure to generate HR edges on a pin configured as output pin. The HETXOR register allows a logical XOR of the output signals of two consecutive HR structures N (even) and N+1 (odd). See Figure 23-13. In this way, it is possible to generate pulses smaller than the loop resolution clock since both edges can be generated by two independent HR structures. This is especially required for symmetrical PWM. See Figure 23-14.

The hardware provides a XOR gate that is connected to the outputs of the HR structure of two consecutive pins. In this structure, pin N+1 remains available for general-purpose input/output.

Figure 23-13. XOR-shared HR I/O



The following N2HET program gives an example for **one** channel of the symmetrical PWM. The generated timing is given in Figure 23-14.

```
MAXC .equ 22
A_   .equ 0  ; HR structure HR0
B_   .equ 1  ; HR structure HR1

CN CNT   { next=EA, reg=A, max=MAXC }

EA ECMP  { next=EB, cond_addr=MA, hr_lr=HIGH, en_pin_action=ON, pin=A_,
          action=PULSELO, reg=A, data=17, hr_data=115 }

MA MOV32 { next=EB, remote=EA, type=IMTOREG&REM, reg=NONE, data=17, hr_data=19 }

EB ECMP  { next=CN, cond_addr=MB, hr_lr=HIGH, en_pin_action=ON, pin=B_,
          action=PULSELO, reg=A, data=5, hr_data=13 }

MB MOV32 { next=CN, remote=EB, type=IMTOREG&REM, reg=NONE, data=5, hr_data=13 }
```

N2HET Settings and output signal calculation for this example program:

- Pin HET[0] and HET[1] are XOR-shared.
- HETPFR[31:0] register = 0x700: lr=128, hr=1, time slots ts = 128
- PWM period (determined by CNT_max field) = (22+1) · LRP = 2944 HRP
- Length of high pulse of (HET[0] XOR HET[1]) =

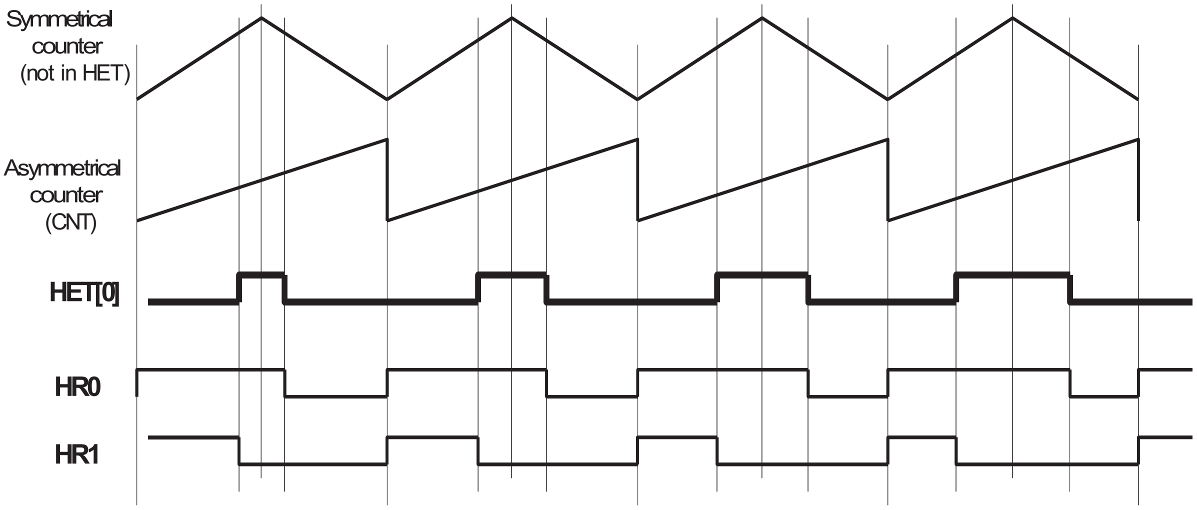
  LH = (17·LRP+115·HRP) - (5·LRP+13·HRP)

  With lr=128 there is LRP = 128 · HRP, so

  LH = (2291 - 653) · HRP = 1638 HRP
- Duty cycle = DC = LH / PWM_period = 1638 HRP / (2944·HRP) = 55.6 %

Figure 23-14 graphically shows the implementation of the XOR-shared feature. The first 2 waveforms (symmetrical counter and CNT) show a symmetric counter and asymmetric counter. The symmetric counter is shown only to highlight the axis of symmetry and is not implemented in the N2HET. The asymmetric counter, which is implemented with a CNT instruction, needs to be set to the period of the symmetric counter. The next two waveforms (HR [0] and HR [1]) show the output of the HR structures, which are the inputs for the XOR gate to create the PWM output on pin HET[0]. Notice that the pulses of signal HET[0] are centered about the axis of symmetry.

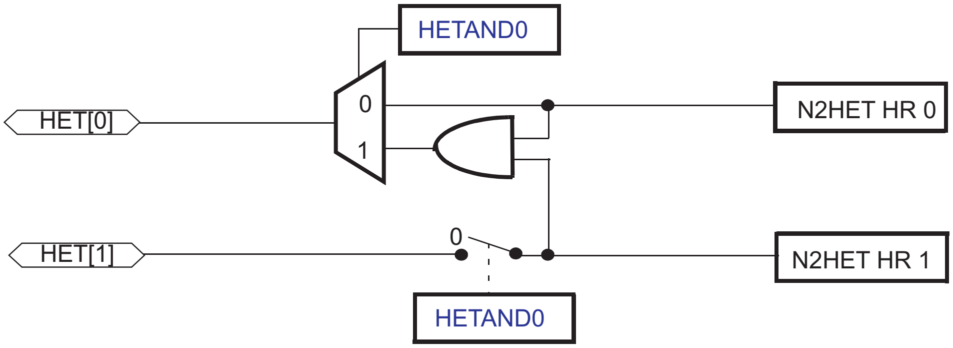
Figure 23-14. Symmetrical PWM with XOR-sharing Output



As an alternative, HR structures may be shared using a logical AND function to combine the effects of the pin structures. The HETAND allows sharing two consecutive HR structures N (even) and N+1 (odd). See Figure 23-15. In this structure, pin N+1 remains available for general-purpose input/output.

NOTE: Setting both the HETAND bit and HETXOR bits at the same time for a given pair of N2HET pins is not supported, must be avoided by the application program.

Figure 23-15. AND-shared HR I/O



23.2.5.7 Loop Back Mode

The loop back feature can be used by the application to monitor an N2HET output signal. For example, if a PWM is generated by HR structure 0, then a PCNT instruction assigned to HR structure 1 can measure back the pulse length or periods of the PWM output signal.

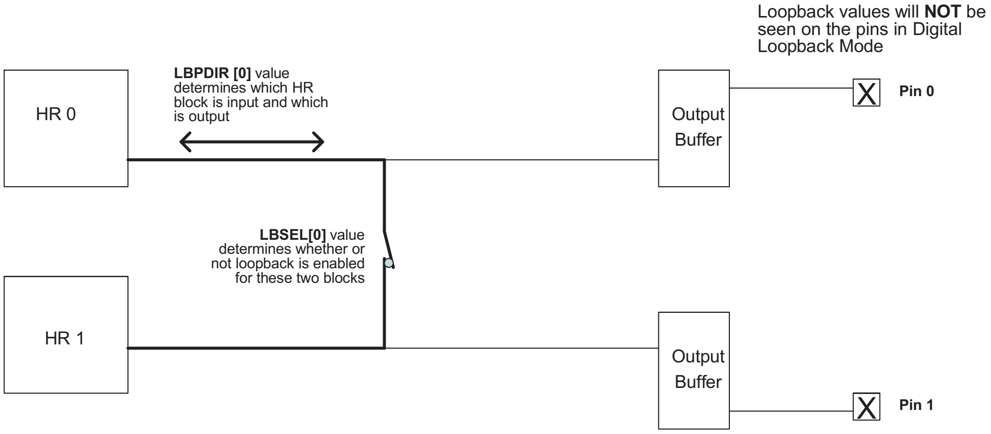
Loopback mode is activated between two high resolution structures by setting LBPSEL[x] to 1 in the HETLBPSEL register for the corresponding structure pair. The **direction** of the loopback between the two structures in the structure pair is determined by the value of LBPDIR[x] in the HETLBPDIR Register.

For example, if bit LBPSEL[0] is set to 1, then HR structures 0 and 1 will be internally connected in loop back mode. If bit LBPDIR[0] is set to 0, then structure 0 will be the input and structure 1 will be the output.

Digital Loopback

Digital loopback mode is enabled by setting LBPTYPE[x] to 0 in the HETLBPSEL register for the corresponding structure pairs. In digital loopback mode, the structure pairs are connected directly and the output buffers are bypassed. Therefore, the loopback values will NOT be seen on the corresponding pins. Figure 23-16 shows an example of digital loopback between structures HR0 and HR1. LBSEL[0] has been set to 1 to enable loopback between the two structures. LBTYPE[0] has been set to 0 to select digital mode for the loopback pair. The LPBDIR[0] value will determine the direction of the loopback by selecting which of the HR blocks is output, and which is input. The bold lines show the digital loopback path.

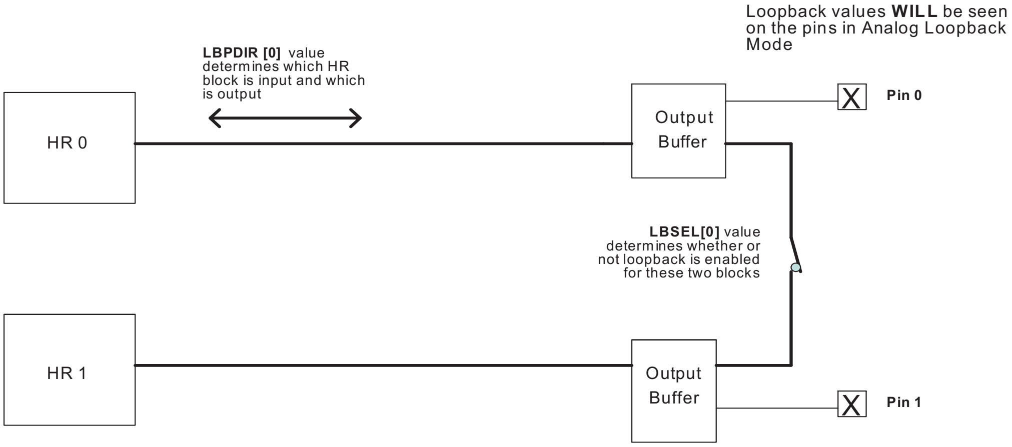
Figure 23-16. HR0 to HR1 Digital Loopback Logic: LBTYPE[0] = 0


Loopback values will **NOT** be seen on the pins in Digital Loopback Mode
HR 0
LBPDIR [0] value determines which HR block is input and which is output
Output Buffer
Pin 0
LBSEL[0] value determines whether or not loopback is enabled for these two blocks
HR 1
Output Buffer
Pin 1

Analog Loopback

Analog loopback mode is enabled by setting LBPTYPE[x] to 1 in the HETLBPSEL register for the corresponding structure pairs. In analog loopback mode, the structure pairs are connected outside of the output buffers. Therefore, the loopback values WILL be seen on the corresponding pins. Figure 23-17 shows an example of analog loopback between structures HR0 and HR1. LBSEL[0] has been set to 1 to enable loopback between the two structures. LBTYPE[0] has been set to 1 to select analog mode for the loopback pair. The LPBDIR[0] value will determine the direction of the loopback by selecting which of the HR blocks is output, and which is input. The bold lines show the analog loopback path.

Figure 23-17. HR0 to HR1 Analog Loop Back Logic: LBTYPE[0] = 1


Loopback values **WILL** be seen on the pins in Analog Loopback Mode
LBPDIR [0] value determines which HR block is input and which is output
HR 0
Output Buffer
Pin 0
LBSEL[0] value determines whether or not loopback is enabled for these two blocks
HR 1
Output Buffer
Pin 1

Note:

- The loop back direction can be selected independent of the HETDIR register setting.
- The pin that is not driven by the N2HET output pin actions can still be used as normal GIO pin.

23.2.5.8 Edge Detection Input Timing

There are several timing requirements for input signals in order to be captured correctly by N2HET. Figure 23-18 illustrates these requirements, with min and max values described in Table 23-7 (Loop Resolution) and Table 23-8 (High Resolution).

Figure 23-18. N2HET Input Edge Detection

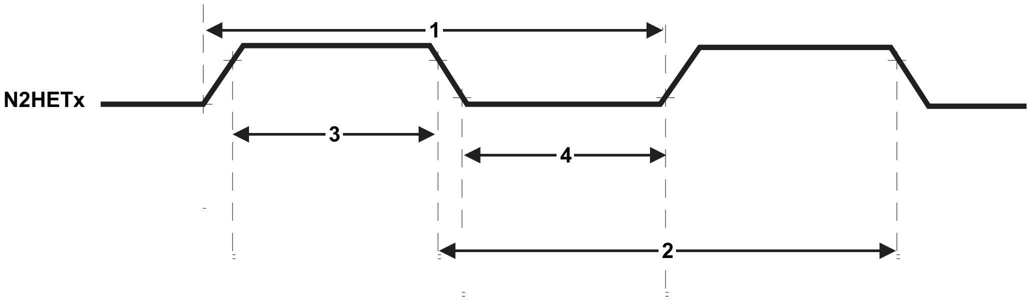


Table 23-7. Edge Detection Input Timing for Loop Resolution Instructions

Parameter #	Description	min	max
1	Input Signal Period, rising edge to rising edge	> 2 (hr) (lr) $t_{c(VCLK2)}$	< 2^{25} (hr) (lr) $t_{c(VCLK2)}$
2	Input Signal Period, falling edge to falling edge		
3	Input Signal, high phase	> (hr) (lr) $t_{c(VCLK2)}$	
4	Input Signal, high phase		

Table 23-8. Edge Detection Input Timing for High Resolution Instructions

Parameter #	Description	min	max
1	Input Signal Period, rising edge to rising edge	> (hr) (lr) $t_{c(VCLK2)}$	< 2^{25} (hr) (lr) $t_{c(VCLK2)}$
2	Input Signal Period, falling edge to falling edge		
3	Input Signal, high phase	> 2 (hr) $t_{c(VCLK2)}$	
4	Input Signal, high phase		

These are the N2HET architectural limitations. Actual limitations will be slightly different due to on chip routing and IO buffer delays, usually by several nanoseconds. Be sure to consult the device datasheet for actual timings that apply to that device. Also, certain devices place additional restrictions on which pins support the high resolution timings of Table 23-8, if present these additional limitations will also be called out in the device datasheet.

Note that the max limit in Table 23-7 and Table 23-8 is based on the counter range of a single N2HET instruction. The max value could be extended by employing an additional N2HET instruction to keep track of counter overflows of the input counter / capture instruction.

23.2.5.9 PWM Generation Example 1 (in HR Mode)

The following example shows how an ECMP instruction works in high resolution mode. The example assumes a VCLK2 of 32 MHz and the following values for the prescale divide rates (hr and lr), number of time slots (ts), high and loop resolution period (HRP and LRP):

hr = 2, lr = 4, ts = hr × lr = 8

HRP = hr / VCLK2 = 2 / 32 MHz = 62.5 ns

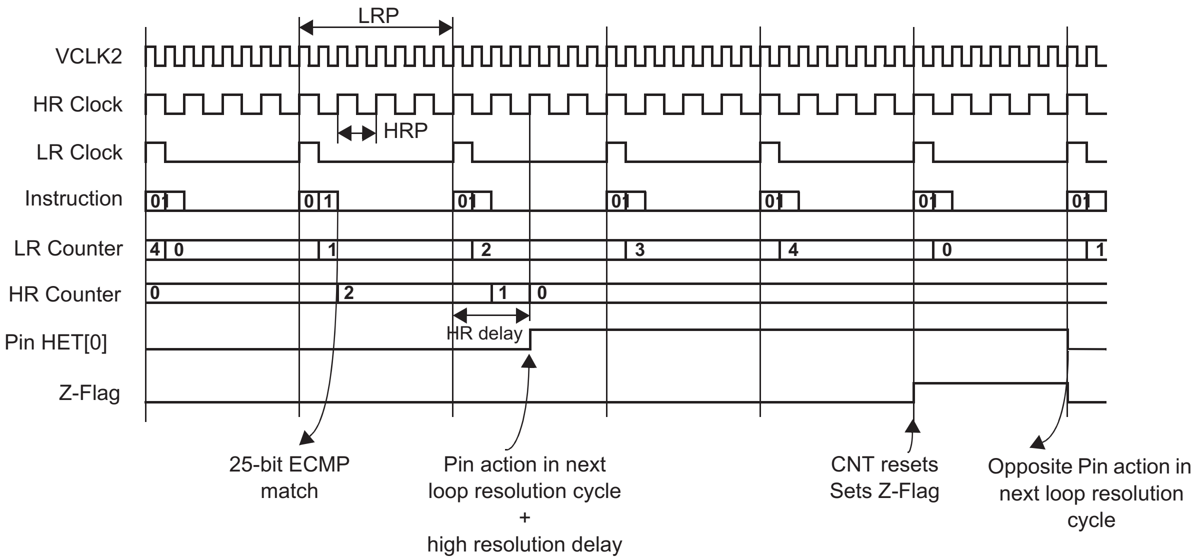
LRP = (hr × lr) / VCLK2 = 8 / 32 MHz = 250 ns

With ts = 8, there are eight time slots available for the program execution, which in this case will consist of one CNT and one ECMP instruction as shown below. The data field of the ECMP instruction is the 32-bit compare value, whereby the lower 7 bits represent the high resolution compare field.

When the 25-bit (loop resolution) compare matches, the HR compare value will be loaded from the 7 lower bits of the instruction data field to the HR counter. At the next loop resolution clock, the HR counter will count down at the HR clock frequency and perform the pin action when it reaches zero.

In the example illustrated by Figure 23-19, the 25-bit compare value is 1 and the 7-bit HR compare value is 2. According to Section 23.2.3.2, depending on the loop resolution divide rate (lr), only certain bits of the 7-bit HR compare value are valid. In this example only the upper 2 bits (D[6:5]) are taken into account. The example program below has a setting of hr_data = 100000b. Shifting this value right by 5 bits, results in 10b which equals the two HR clock cycles delay mentioned above.

Figure 23-19. ECMP Execution Timings



```
HETPFR[31:0] register = 0x201 --> lr=4 and hr=2 --> ts = 8

N2HET Program:

L00   CNT   { next= L01, reg=A, irq=OFF, max = 4 }
L01   ECMP  { next= L00, cond_addr= L00, hr_lr=HIGH, en_pin_action=ON, pin=0,
              action=PULSEHI, reg=A, irq=OFF, data= 1, hr_data = 0x40 }

; 25 bit compare value is 1 and the 7-bit HR compare value is 2
; (Because of lr=4 the D[4:0] of the 7-bit HR field are ignored )
```

NOTE: ECMP Opposite Actions

ECMP opposite pin actions are always synchronized to the loop resolution clock.

Changing the duty cycle of a PWM generated by an ECMP instruction, can lead to a missing pulse if the data field of the instruction is updated directly. This can happen when it is changed from a high value to a lower value while the CNT instruction has already passed the new updated lower value. To avoid this a synchronous duty cycle update can be performed with the use of an additional instruction (MOV32). This instruction is only executed when the compare of the ECMP matches. For this the cond_addr of the ECMP needs to point to the MOV32. On execution of the MOV32, it moves its data field into the data field of the ECMP. The update of the duty cycle has to be made to the MOV32 data field instead of the ECMP data field.

23.2.5.10 PWM Generation Example 2 (in HR Mode)

The MCMP instruction can also be used in HR mode. In this case operation is exactly the same as for the ECMP instruction except that the 25-bit low resolution is now the result of a magnitude compare (greater or equal) rather than an equality compare. When the 25-bit (loop resolution) magnitude compare matches, the HR compare value will be loaded from the 7 lower bits of the instruction data field to the HR counter. At the next loop resolution clock, the HR counter will count down at the HR clock frequency and perform the pin action when it reaches zero.

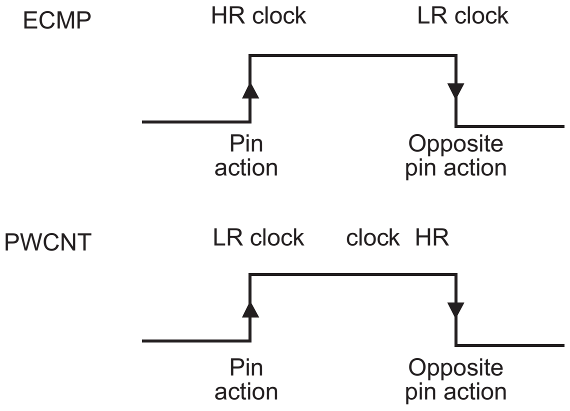
The MCMP instruction avoids the missing pulse problem of the ECMP instruction (see previous example), however the duty cycle of the signal might not be exact for one PWM period. The benefit of the MCMP is that it avoids adding another instruction to do the duty cycle update synchronously.

23.2.5.11 Pulse Generation Example (in HR Mode)

The PWCNT instruction may also be used in HR mode to generate pulse outputs with HR width. It generates a single pulse when the data field of the instruction is non-zero. It remains at the opposite pin action when the data field is zero.

The PWCNT instruction operates conversely to the ECMP instruction. See Figure 23-20. For PWCNT, the opposite pin action is synchronous with the HR clock and for ECMP the pin action is synchronous with the HR clock. The PWCNT pin action is synchronous with the loop resolution clock.

Figure 23-20. High/Low Resolution Modes for ECMP and PWCNT



23.2.5.12 Pulse Measurement Example (in HR Mode)

The PCNT instruction captures HR measurement of the high/low pulse time or periods of the input. As shown in Figure 23-21, at marker (1) the input goes HIGH and the HR counter immediately begins to count. The counter increments and rolls over until the falling edge on the input pin, where it captures the counter value into the HR capture register (marker (2)). The PCNT instruction begins counting when the synchronized input signal goes HIGH and captures both the 25-bit data field and the HR capture register into RAM when the synchronized input falls (marker (3)).

NOTE: The HR capture value written into RAM is shifted appropriately depending on the loop resolution prescale divide rate (lr). (See also Section 23.2.3.2).

Figure 23-21 shows what happens when the capture edge arrives *after* the HR counter overflows. This causes the incremented value to be captured by the PCNT instruction.

Figure 23-21. PCNT Instruction Timing (With Capture Edge After HR Counter Overflow)

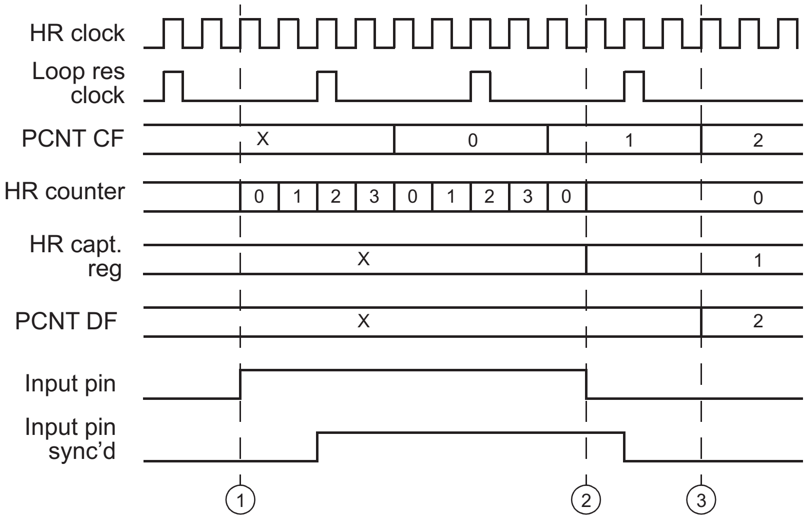
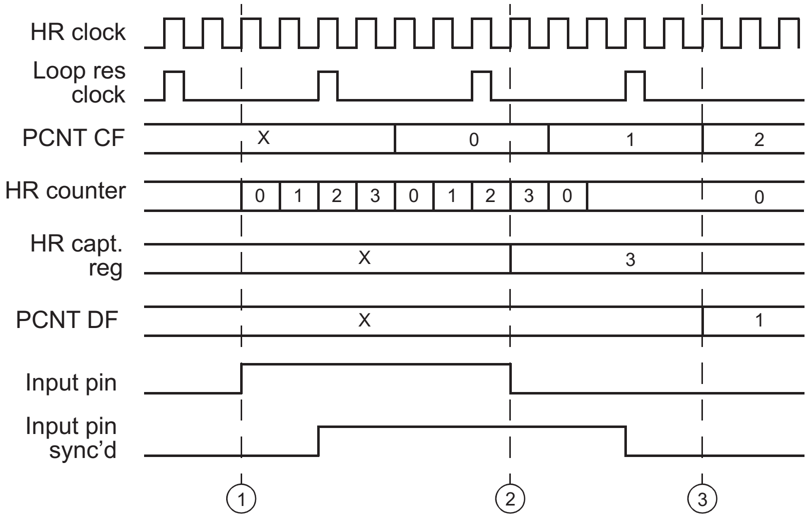


Figure 23-22 shows what happens when the capture edge arrives *before* the HR counter overflows. This causes the non-incremented value to be captured by the PCNT instruction.

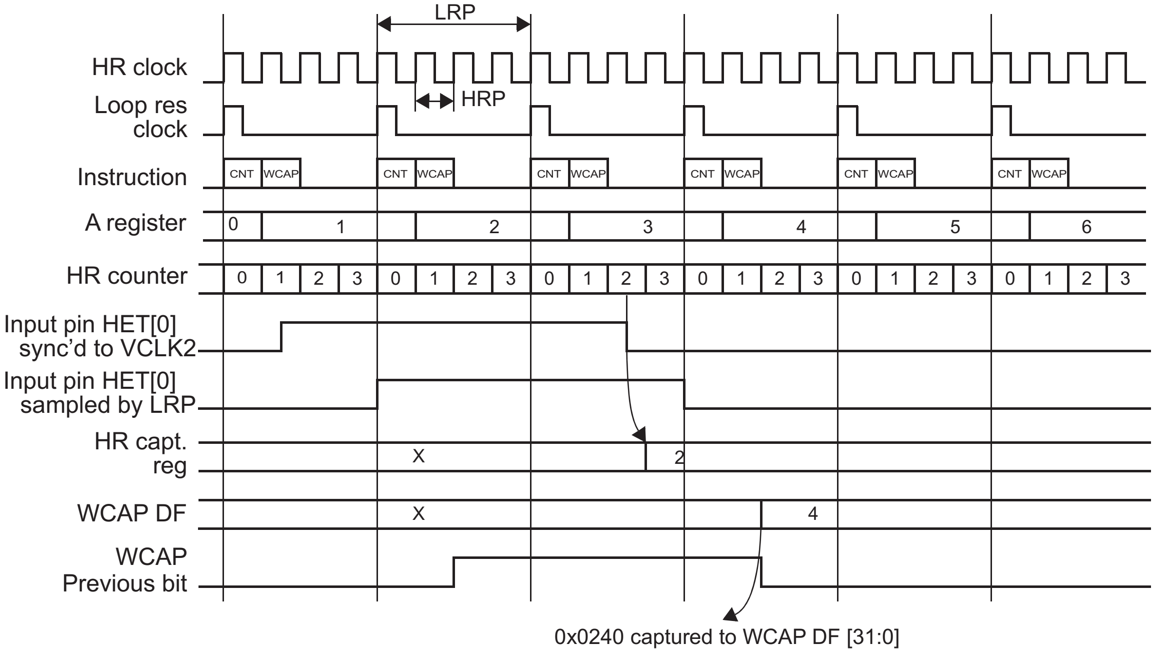
Figure 23-22. PCNT Instruction Timing (With Capture Edge Before HR Counter Overflow)



23.2.5.13 WCAP Execution Example (in HR Mode)

The HR capability is enabled for WCAP, if its hr_lr bit is zero. In this case the HR counter is always enabled and is synchronized with the resolution loop. When the specified edge is detected, the current value of the HR counter is captured in the HR capture register and written into the RAM after the next WCAP execution. The WCAP instruction effectively time stamps the free running timer saved in a register (for example, register A shown in Figure 23-23).

Figure 23-23. WCAP Instruction Timing



```
HETPFR_register = 0x0200 --> lr = 4, hr = 1, ts = 4
```

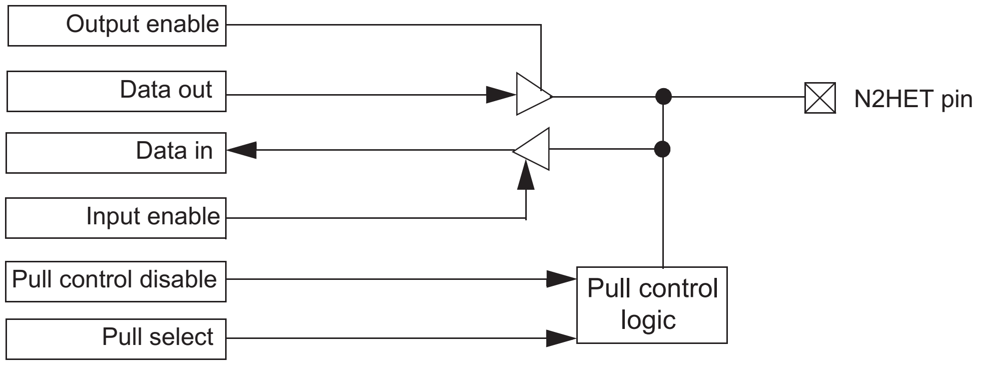
N2HET Program:

```
L00 CNT  {reg=A, max=01ffffffh}
L01 WCAP {next=L00, cond_addr=L00, hr_lr=high, reg=A, event= FALL, pin=0,
          data=0}
```

In the example, the WCAP is configured to capture the counter when a **falling** edge occurs. The WCAP data field (WCAP_DF) is updated in the loop succeeding the loop in which the edge occurred. The WCAP instruction evaluates an edge by comparing its Previous bit with the sync'd input signal. In Figure 23-23, the current value of the counter (4) is captured to WCAP_DF[31:7] and the value of the HR capture register (2) is transferred to the valid bits (according the lr prescaler) of WCAP_DF[6:0]. Therefore, in the example 0x0240 is captured in WCAP_DF[31:0].

23.2.5.14 I/O Pull Control Feature

Figure 23-24. I/O Block Diagram Including Pull Control Logic



The following apply if the device is under reset:

- Pull control: The reset pull control on the pins is enabled and a pulldown is configured.
- Input buffer: The input buffer is enabled.
- Output buffer: The output buffer is disabled.

The following apply if the device is out of reset:

- Pull control: The pull control is enabled by clearing the corresponding bit in the N2HET Pull Disable Register (HETPULDIS). In this case, if the corresponding bit in the N2HET Pull Select Register (HETPSL) is set, the pin will have a pull-up; if the bit in the N2HET Pull Select Register (HETPSL) is cleared, the pin will have a pull-down. If the bit in the N2HET Pull Disable Register (HETPULDIS) is set, there is no pull-up or pull-down on the pin.
- Input buffer: The input buffer is disabled only if the pin direction is set to input AND the pull control is disabled AND pull down is selected as the pull bias. In all other cases, the input buffer is enabled.

NOTE: The pull-disable logic depends on the pin direction. If the pin is configured as output, then the pulls are disabled automatically. If the pin is configured as input, the pulls are enabled or disabled depending on the pull disable register bit.

- Output buffer: A pin can be driven as an output pin if the corresponding bit in the N2HET Direction Register (HETDIR) is set AND the open-drain feature (N2HET Open Drain Register (HETPDR)) is not enabled. See Section 23.2.5.15 for more details.

The behavior of the input buffer, output buffer, and the pull control is summarized in Table 23-9. When an input buffer is disabled, it appears as a logic low to on-chip logic.

Table 23-9. Input Buffer, Output Buffer, and Pull Control Behavior

Device under Reset?	Pin Direction (DIR)[1]	Pull Disable (PULDIS)[1]	Pull Select (PULSEL)[1]	Pull Control	Output Buffer	Input Buffer
Yes	X	X	X	Enabled	Disabled	Enabled
No	0	0	0	Pull down	Disabled	Enabled
No	0	0	1	Pull up	Disabled	Enabled
No	0	1	0	Disabled	Disabled	Disabled
No	0	1	1	Disabled	Disabled	Enabled
No	1	X	X	Disabled	Enabled	Enabled

[1] X = Don't care

23.2.5.15 Open-Drain Feature

The following apply if the open-drain feature is enabled on a pin, that is, the corresponding bit in the N2HET Open Drain Register (HETPDR) is set:

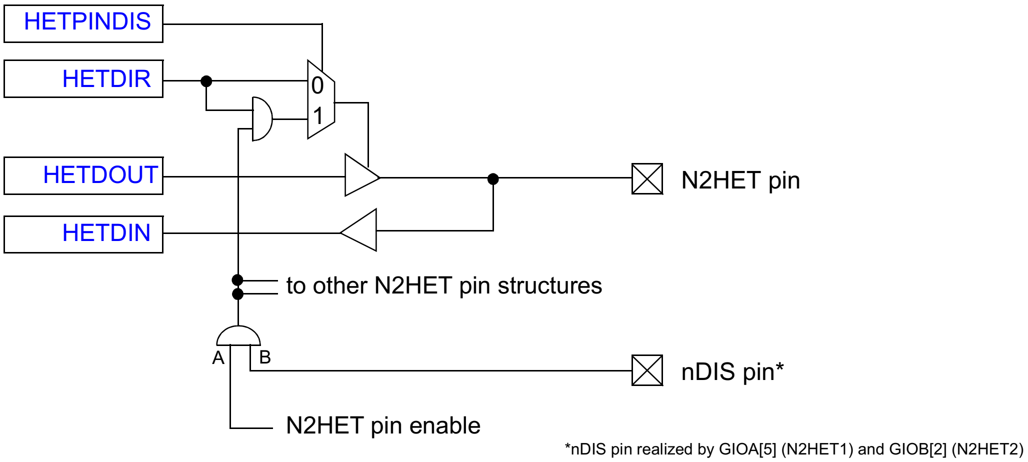
- Output buffer is enabled if a low signal is being driven internally to the pin.
- The output buffer is disabled if a high signal is being driven internally to the pin.

23.2.5.16 N2HET Pin Disable Feature

This feature is provided for the safe operation of systems such as power converters and motor drives. It can be used to inform the monitoring software of motor drive abnormalities such as over-voltage, over-current, and excessive temperature rise.

Table 23-10 shows the conditions for the output buffer to be enabled/disabled.

Figure 23-25. N2HET Pin Disable Feature Diagram



*nDIS pin realized by GIOA[5] (N2HET1) and GIOB[2] (N2HET2)

Table 23-10. N2HET Pin Disable Feature

HETPINDIS.x	nDIS Pin (Input)	HET_PIN_ENA (HETGCR.24)	HETDIR.x	Output Buffer
0	X	X	0	Disabled
0	X	X	1	Enabled
1	0	X	0	Disabled
1	0	X	1	Disabled
1	1	X	0	Disabled
1	1	0	1	Disabled
1	1	1	1	Enabled

An interrupt capable device I/O pin can share the same pin as the N2HET nDIS signal. Normally GIOA[5] serves as nDIS for N2HET1 and GIOB[2] as nDIS for N2HET2. Check the device datasheet for the actual implementation. Sharing a pin with a GIO pin that is Interrupt capable allows the N2HET nDIS input to also generate an interrupt to the CPU. An active low level on nDIS is intended to signal an abnormal situation as described above. All N2HET pins, which are selected with the N2HET Pin Disable Register (HETPINDIS), will be put in the high-impedance state by hardware immediately after the nDIS signal is pulled low. At this time a CPU interrupt is issued, if it is enabled in the GIO pin logic.

The bit HET_PIN_ENA is automatically cleared in the failure condition and this state remains as long as the software explicitly sets the bit again. The steps to do this are:

- Software detects, by reading the HETDIN register of the GIO pin, that the level on nDIS is inactive (high).
- Software sets bit HET_PIN_ENA to deactivate the high impedance state of the pins.

23.2.6 Suppression Filters

Each N2HET pin is equipped with a suppression filter. If the pin is configured as an input it enables to filter out pulses shorter than a programmable duration. Each filter consists of a 10-bit down counter, which starts counting at a programmable preloaded value and is decremented using the VCLK2 clock.

- The counter starts counting when the filter input signal has the opposite state of the filter output signal. The output signal is preset to the same input signal state after reset, in order to ensure proper operation after device reset.
- Once the counter reaches zero without detecting an opposite pin state on the filter input signal, the output signal is set to the opposite state.
- When the counter detects an opposite pin action on the filter input signal before reaching zero, the counter is loaded with it's preload value and the opposite pin action on the filter output signal does not take place. The counter resumes at the preload value until it detects an opposite pin action on the input signal again.
- Therefore the filter output signal is delayed compared to the filter input signal. The amount of delay depends on the counter clock frequency (VCLK2) and the programmed preload value.
- The accuracy of the output signal is +/- the counter clock frequency.

Table 23-11 gives examples for a 100 MHz VCLK2 frequency.

Figure 23-26. Suppression Filter Counter Operation

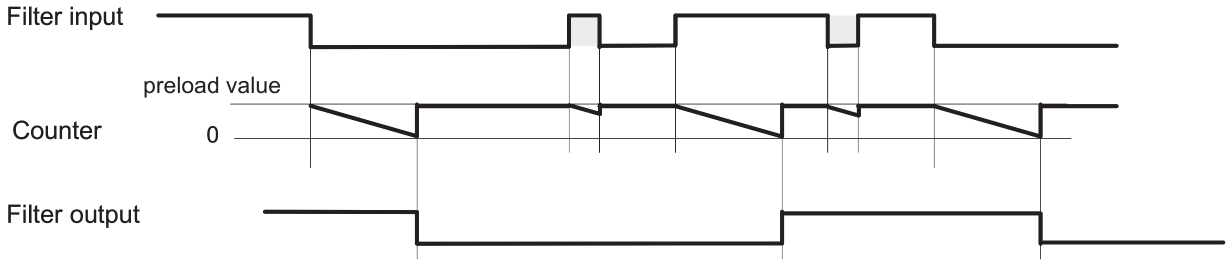


Table 23-11. Pulse Length Examples for Suppression Filter

Divider CCDIV	VCLK2	Possible values for the suppressed pulse length / frequency resulting from the programmable 10 bit preload value (0,1,..,1023)	
1	100.0 MHz	10 ns, 20 ns, …, 10.22 µs, 10.23 µs	50 MHz, 25 MHz, …, 48.924 kHz, 48.876 kHz
2	50.0 MHz	20 ns, 40 ns, …, 20.44 µs, 20.48 µs	25 MHz, 12.5 MHz, …, 24.462 kHz, 24.414 kHz
3	33.3 MHz	30 ns, 60 ns, …, 30.66 µs, 30.69 µs	16.7 MHz, 8.3 MHz, …, 16.308 kHz, 16.292 kHz

23.2.7 Interrupts and Exceptions

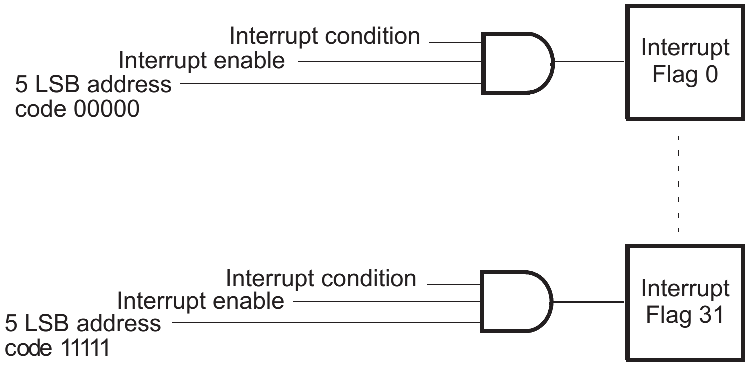
N2HET interrupts can be generated by any instruction that has an interrupt enable bit in its instruction format. When the interrupt condition in an instruction is true and the interrupt enable bit of that instruction is set, an interrupt flag is then set in the N2HET Interrupt Flag Register (HETFLG). The address code for this flag is determined by the five LSBs of the current timer program address. The flag in the N2HET Interrupt Flag Register (HETFLG) is set even if the corresponding bit in the N2HET Interrupt Enable Set Register (HETINTENAS) is 0. To generate an interrupt, the corresponding bit in the N2HET Interrupt Enable Set Register (HETINTENAS) must be 1. In the N2HET interrupt service routine, the main CPU must first determine which source inside the N2HET created the interrupt request. This operation is accelerated by the N2HET Offset Index Priority Level 1 Register (HETOFF1) or N2HET Offset Index Priority Level 2 Register (HETOFF2) that automatically provides the number of the highest priority source within each priority level. Reading the offset register will automatically clear the corresponding N2HET interrupt flag that created the request. However, if the offset registers are not used by the N2HET interrupt service routine, the flag should be cleared explicitly by the CPU once the interrupt has been serviced.

Table 23-12. Interrupt Sources and Corresponding Offset Values in Registers HETOFFx

Source No.	Offset Value
no interrupt	0
Instruction 0, 32, 64...	1
Instruction 1, 33, 65...	2
:	:
Instruction 31, 63, 95...	32
Program Overflow	33
APCNT underflow:	34
APCNT overflow	35

The instructions capable of generating interrupts are listed in Table 23-75.

Figure 23-27. Interrupt Functionality on Instruction Level



Each interrupt source is associated with a priority level (level 1 or level 2). When multiple interrupts with the same priority level occur during the same loop resolution the lowest flag bit is serviced first.

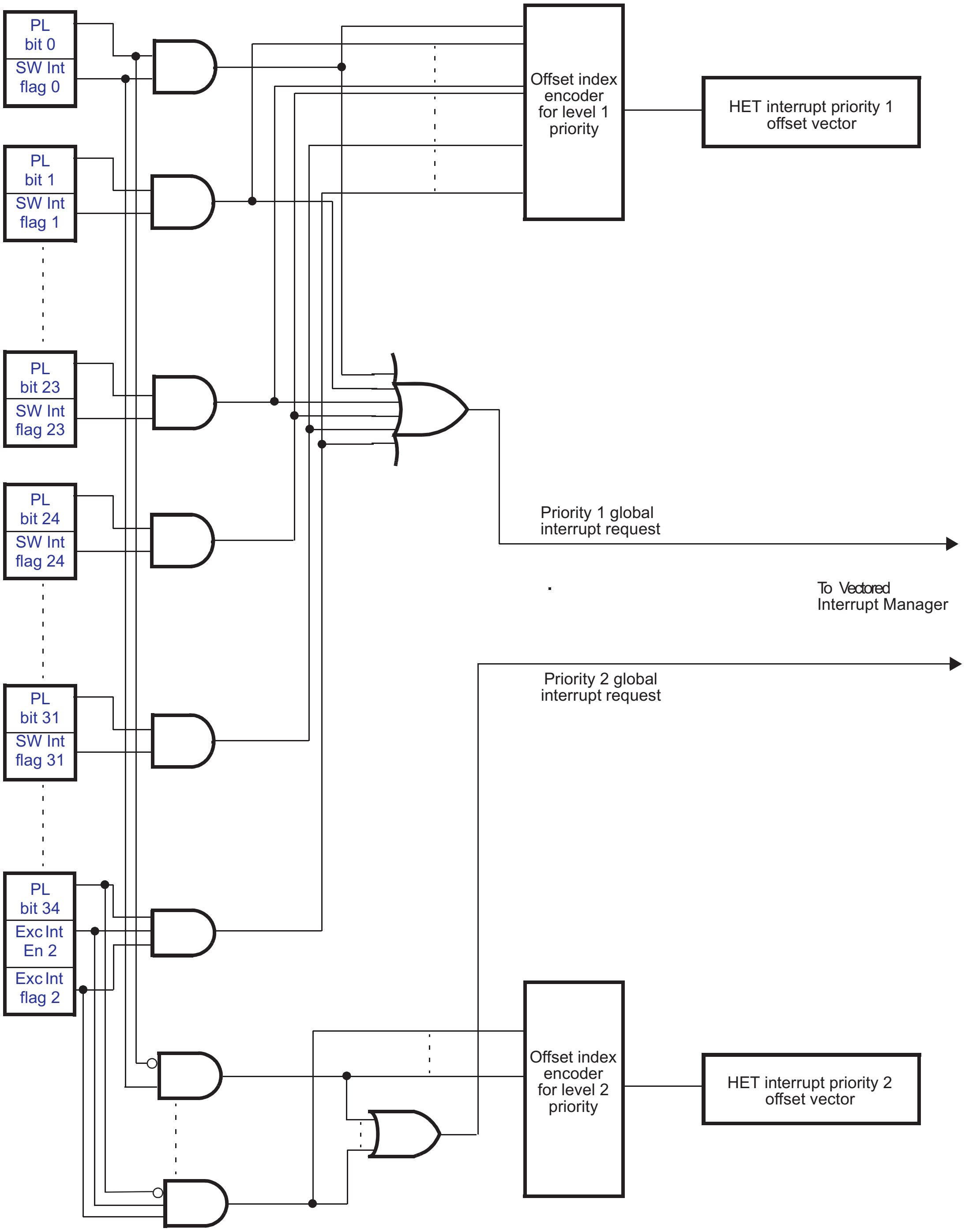
In addition to the interrupts generated by the instructions the N2HET can generate three additional exceptions:

- Program overflow
- APCNT underflow (see Section 23.3.1.2)
- APCNT overflow (see Section 23.3.1.3)

23.2.8 Hardware Priority Scheme

If two or more software interrupts are pending on the same priority level, the offset value will show the one with the highest priority. The interrupt with the highest priority is the one with the lower offset value. This scheme is hard-wired in the offset encoder. See Figure 23-28.

Figure 23-28. Interrupt Flag/Priority Level Architecture

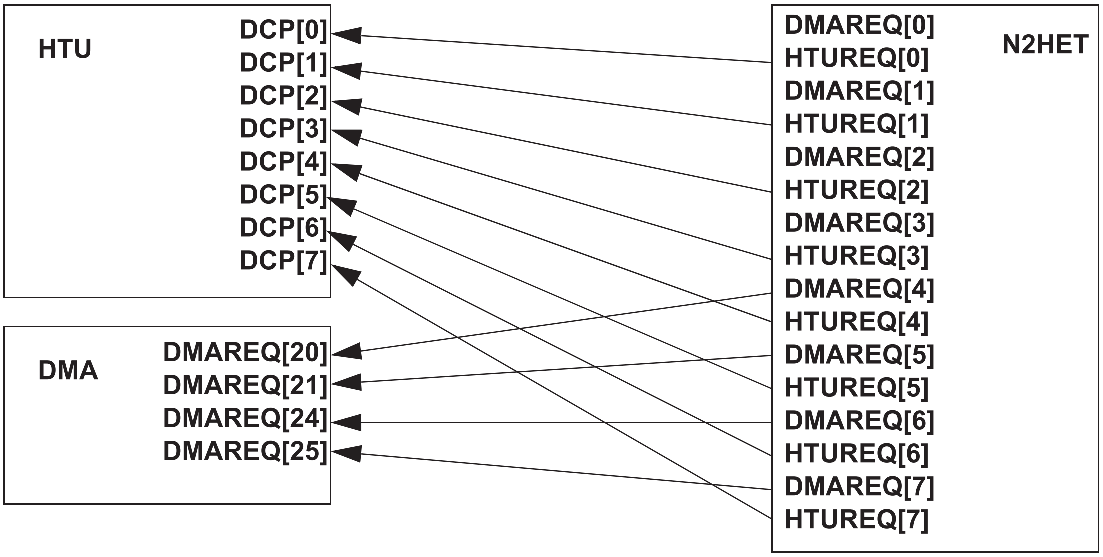

PL
bit 0
SW Int
flag 0
PL
bit 1
SW Int
flag 1
PL
bit 23
SW Int
flag 23
PL
bit 24
SW Int
flag 24
PL
bit 31
SW Int
flag 31
PL
bit 34
Exc Int
En 2
Exc Int
flag 2
Offset index encoder for level 1 priority
HET interrupt priority 1 offset vector
Priority 1 global interrupt request
To Vectored Interrupt Manager
Priority 2 global interrupt request
Offset index encoder for level 2 priority
HET interrupt priority 2 offset vector

23.2.9 N2HET Requests to DMA and HTU

As described in Section 23.6.3, the majority of the N2HET instructions are able to generate a transfer request to the High-End Timer Transfer Unit (HTU) and/or to the DMA module when an instruction-specific condition is true. One N2HET instruction can select one of 8 request lines by programming the “reqnum” parameter. The “request” field in an instruction is used to enable, disable, or to generate a quiet request (see Section 23.6.2) on the selected request line. Quiet requests can be used by the HTU, but not by the DMA. For quiet request, refer to the *High-End Timer Transfer Unit (HTU) Module* chapter (see Section 24.2.4.1).

The configuration of the N2HET Request Destination Select Register (HETREQDS) bits determines if a request line triggers an HTU-DCP, a DMA channel or both. This means the register bits will determine whether an N2HET instruction triggers DMAREQ[x], HTUREQ[x] or both signals (shown in Figure 23-29). The request line number x corresponds to the “reqnum” parameter used in the instruction.

Figure 23-29. Request Line Assignment Example



23.3 Angle Functions

Engine management systems require an angle-referenced time base to synchronize signals to the engine toothed wheel. The N2HET has a method to provide such a time base for low-end engine systems. The reference is created by the N2HET using three dedicated instructions with fractional angle steps equal to /8, /16, /32, /64.

23.3.1 Software Angle Generator

The N2HET provides three specialized count instructions to generate an angle referenced time base synchronized to an external reference signal (the toothed wheel signal) that defines angular reference points.

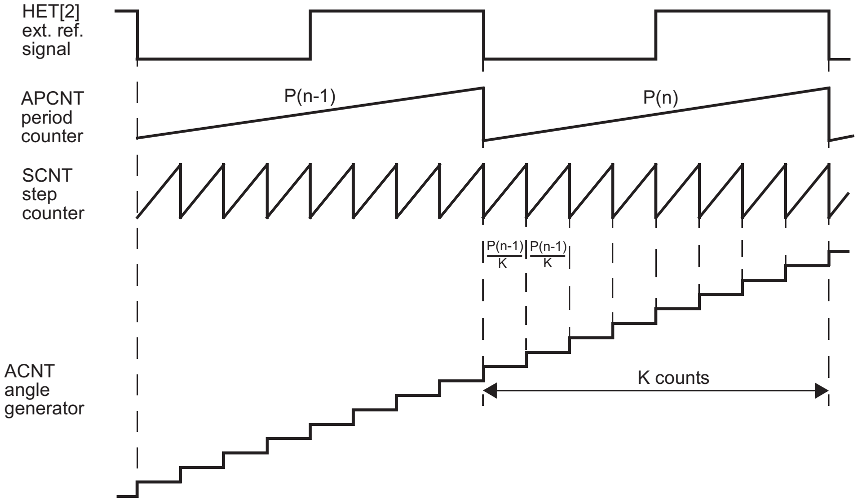
The time base is used to generate fractional angle steps between the reference points. The step width K (= 8, 16, 32, or 64) programmed by the user defines the angle accuracy of the time base. These fractional steps are then accumulated in an angle counter to form the absolute angle value.

The first counter, APCNT, incremented on each loop resolution clock measures the periods P(n) of the external signal. The second counter SCNT counts by step K up to the previous period value P(n-1), measured by APCNT, and then recycles. The resulting period of SCNT is the fraction P(n-1) / K. The third counter ACNT accumulates the fractions generated by SCNT.

Figure 23-30 illustrates the basic operation of APCNT, SCNT, and ACNT.

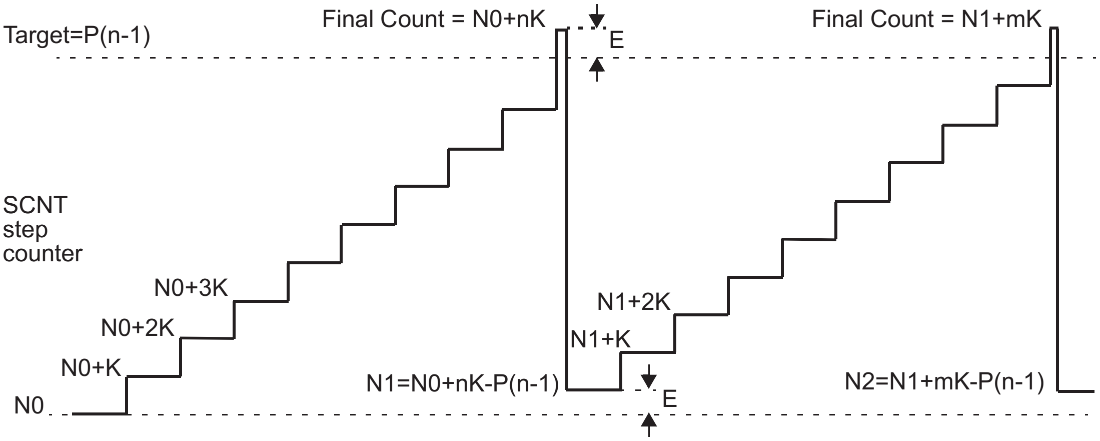
A N2HET timer program can only have one angle generator.

Figure 23-30. Operation of N2HET Count Instructions



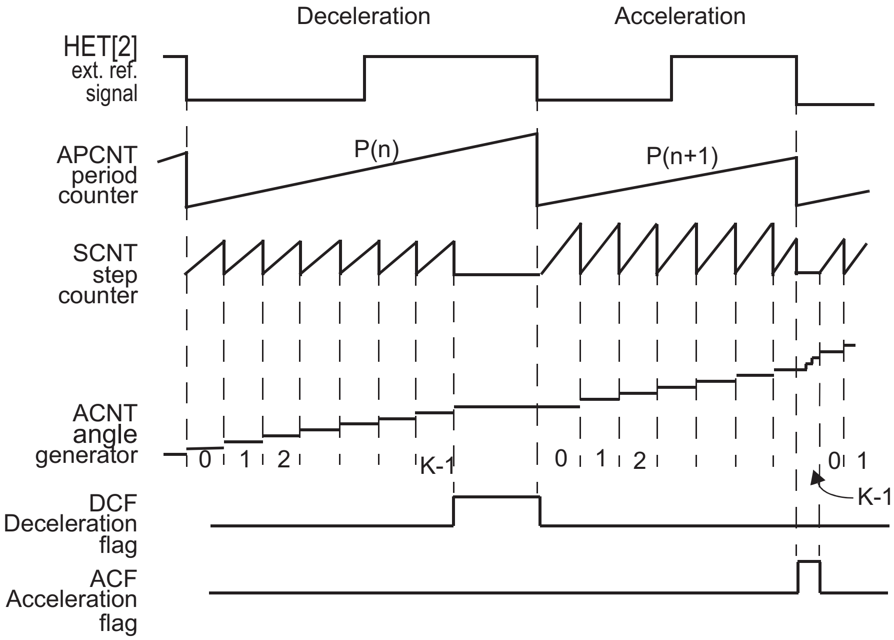
Due to stepping, the final count of SCNT does not usually exactly match the target value P(n-1). Figure 23-31 illustrates how SCNT compensates for this feature by starting each cycle with the remainder (final count - target) of the previous cycle.

Figure 23-31. SCNT Count Operation



ACNT detects period variations of the external signal measured by APCNT and compensates related counting errors. A period increase is flagged in the deceleration flag. A period decrease is flagged in the acceleration flag. If no variation is flagged, ACNT increments the counter value each time SCNT reaches its target. If acceleration is detected, ACNT increments the counter value on each timer resolution (fast mode). If deceleration is detected, ACNT is stopped. Figure 23-32 illustrates how the compensations for acceleration and deceleration operate.

Figure 23-32. ACNT Period Variation Compensations



23.3.1.1 Singularities

Singularities (gaps, in this case, from missing teeth in a toothed wheel) in the external reference signal can be masked. The start and end of singularities are defined by gap start and gap end values specified in SCNT and ACNT. When ACNT reaches gap start or gap end, it sets/resets the gap flag.

While the gap flag is set, new periods of the external reference signal are ignored for angle computation. SCNT uses the last period measured by APCNT just before gap start.

Figure 23-33 and Figure 23-34 illustrate the behavior of the angle generator during a gap after a deceleration or acceleration of the N2HET.

Figure 23-33. N2HET Timings Associated with the Gap Flag (ACNT Deceleration)

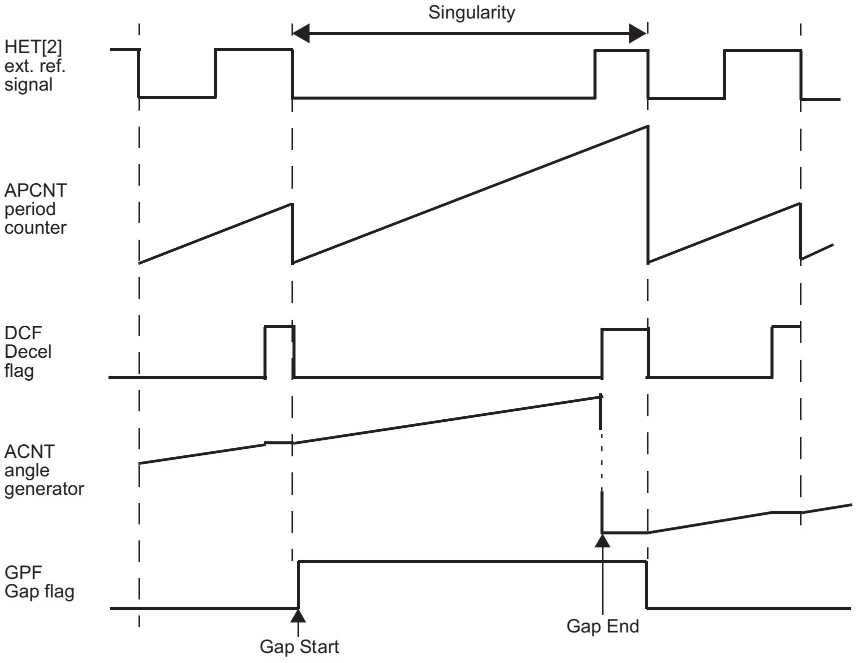
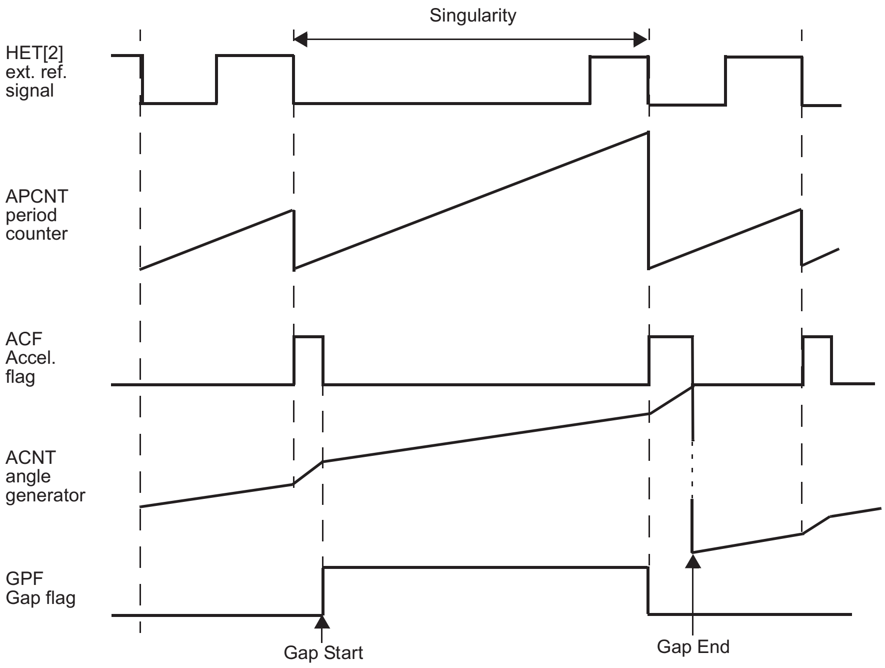


Figure 23-34. N2HET Timings Associated with the Gap Flag (ACNT Acceleration)



23.3.1.2 APCNT Underflow

The fastest valid external signal APCNT can accept must satisfy the following condition:

Step Width K < Period Min. Resolution (LRP)

This condition fixes the maximum possible step width once the minimum period and the resolution of an application are specified.

If a period value accidentally falls below the minimum allowed, APCNT stops the capture of these periods and sets the APCNT underflow interrupt flag located in the exceptions interrupt control register. In such a situation, SCNT and ACNT continue to be executed using the last valid period captured by APCNT.

23.3.1.3 APCNT Overflow

The slowest valid external signal APCNT can measure must satisfy the following condition:

Period Max Resolution < 33554431

When this limit is reached (APCNT Count equals all 1’s), APCNT stays at a maximum count (stops counting). APCNT remains in this position until the next specified capture edge is detected on the selected pin and sets the APCNT overflow interrupt flag located in the exceptions interrupt control register. In this situation, SCNT and ACNT continue to be executed using the maximum APCNT period count.

23.3.2 Hardware Angle Generator (HWAG)

23.3.2.1 Overview

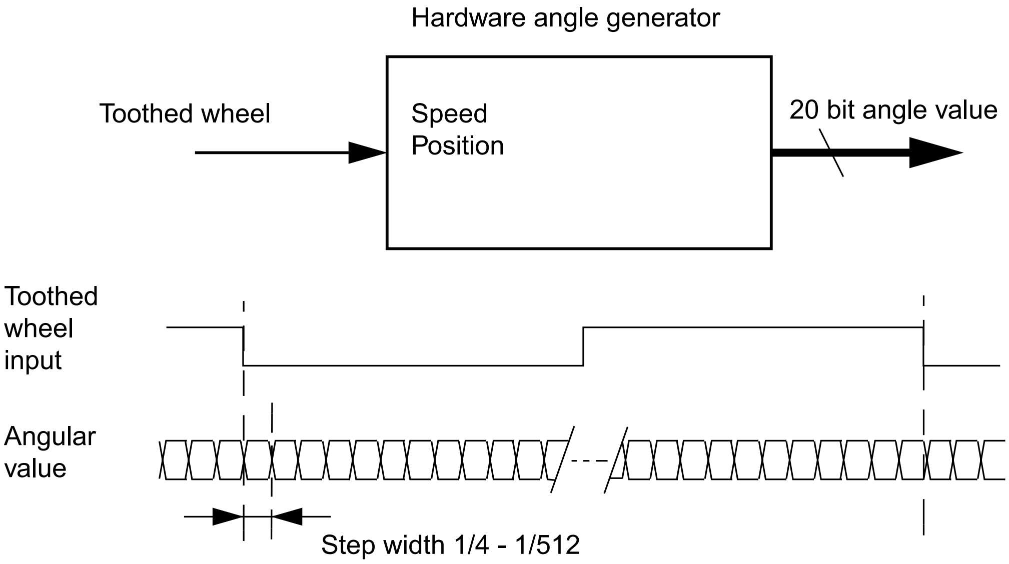
More engine control functions require powerful microcontrollers to process the timing. These controllers must generate signals such as dwell time, spark time, and fuel injection, at precise engine angles. These signals must be synchronized with the engine cycle.

The hardware angle generator (HWAG) generates angle value from toothed wheels. Because the toothed wheels are inaccurate (the most widely wheel used has 60 teeth with 6°/tooth), the period between two tooth edges (\) interpolates the angle value and the step width gives the number of interpolated angles. For an example of the angle generator principle, see Figure 23-35.

The HWAG can complement the high-end timer (NHET) to generate complex angle-angle or angle-time wave forms.

To work with the majority of toothed wheels, the HWAG provides registers to allow the CPU to configure step width, singularity, and filtering when initializing.

Figure 23-35. Angle Generator Principle



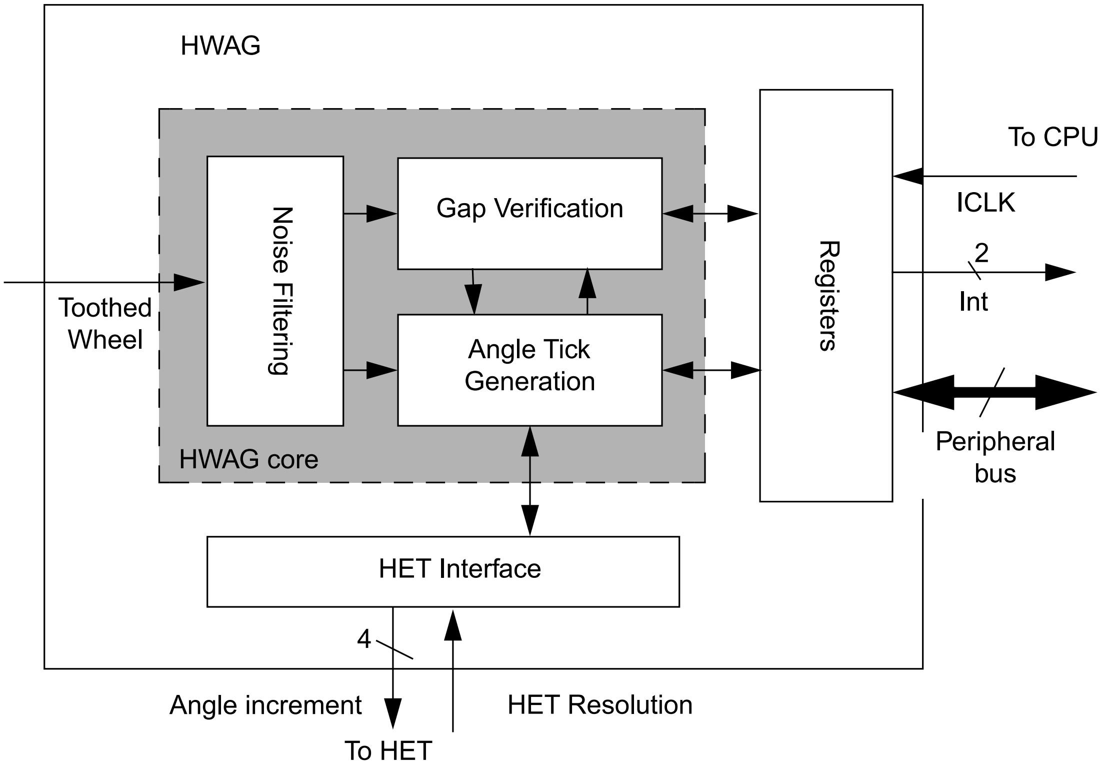
23.3.2.1.1 HWAG Features

The HWAG provides the following features:

- Programmable step width from 1/4 to 1/512
- Automatic synchronization check after first singularity synchronization
- Direct interface with the high-end timer
- 15 to 10,000 RPM range
- Programmable toothed-wheel input filter
- Programmable active edge on toothed-wheel
- Start bit synchronized to the tooth edge
- Pin selection capability for toothed-wheel input

23.3.2.1.2 Block Diagram

Figure 23-36. Hardware Angle Generator Block Diagram



23.3.2.2 HWAG Operation

23.3.2.2.1 Angle Tick Generation Algorithm

23.3.2.2.1.1 Angle Tick Generation Principle

The angle tick generator is the core kernel of this module. It uses the time-interpolation algorithm to generate angle ticks based on the last toothed wheel period. The angle counter is incremented at each new angle tick.

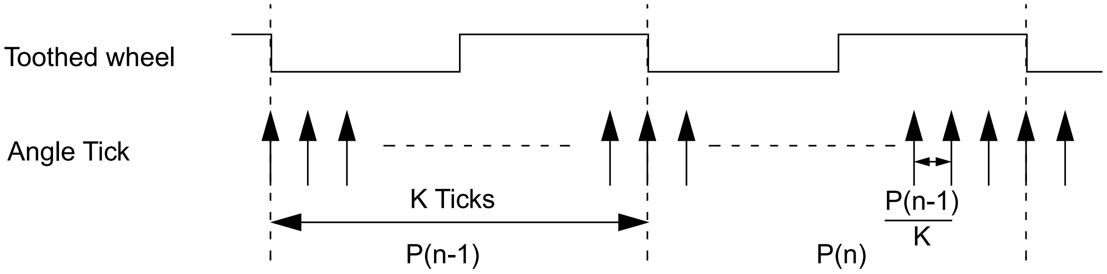
Because the toothed wheel is too inaccurate to fit with actual power-train applications, the algorithm is based on dividing the previous tooth period by K angle steps. The tooth period is the period between two active edges, which the HWAG global control register 2 (HWAGCR2) defines as the falling or the rising edge of the input signal. For an example of the angle tick generation principle, see Figure 23-37.

The speed of the toothed wheel varies. This variance in speed creates some discontinuities in the angle counter behavior.

When the toothed wheel accelerates, the current period becomes shorter than the previous one and the tooth edge arrives before the last tick has been generated. To compensate for any missed ticks, the HWAG adds them to the angle counter when the active edge of the tooth arrives. The angle value is updated and resynchronized at each new active tooth edge.

When the toothed wheel decelerates, the period becomes longer than the previous period and K ticks are already counted before the active edge tooth arrives. After the last tick has been generated, the HWAG generates a tick only after the active tooth edge arrives.

Figure 23-37. Angle Tick Generation Principle



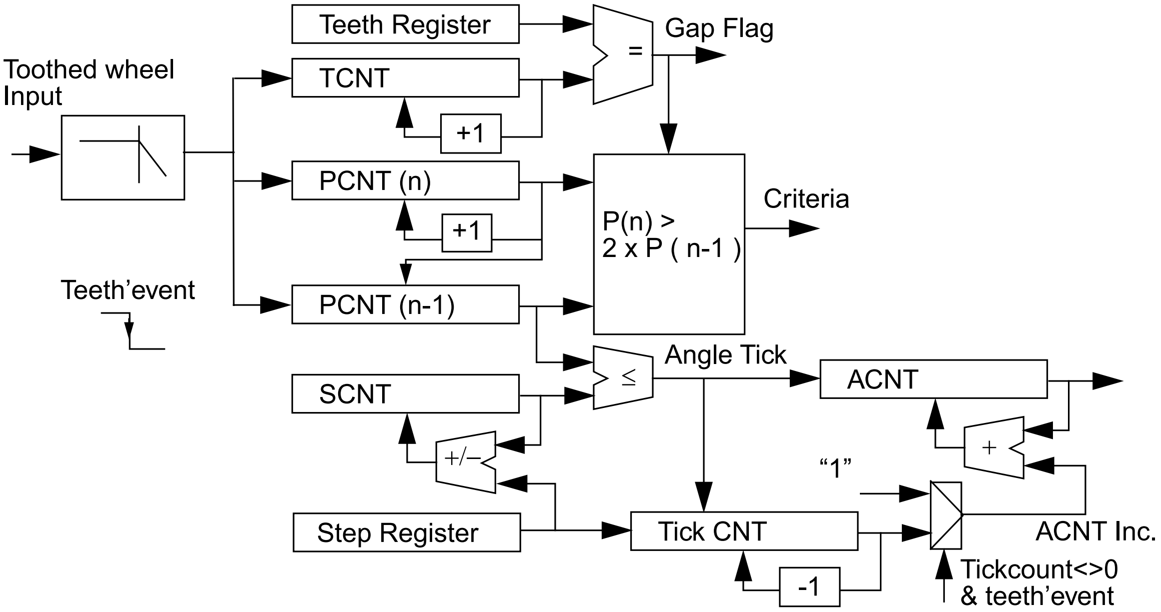
23.3.2.2.1.2 Angle Tick Generation Implementation

The time-interpolation algorithm, which generates ticks based on the toothed wheel tooth period, consists of the following five main counters linked together:

- Tooth counter (TCNT): Current tooth
- Period counter (PCNT): Period between two teeth
- Step counter (SCNT): Angle step
- Tick counter (TCKC): Angle ticks
- Angle counter (ACNT): Angle value

The algorithm also includes differences comparison, adder, and working registers as shown in Figure 23-38.

Figure 23-38. New Angle Tick Generation Architecture



The TCNT is an 8-bit counter. It counts teeth until it reaches the teeth register value then generates a gap flag signal. The gap flag signal which changes the behavior of the HWAG during the singularity and resets the TCNT on the next active edge of the toothed wheel input.

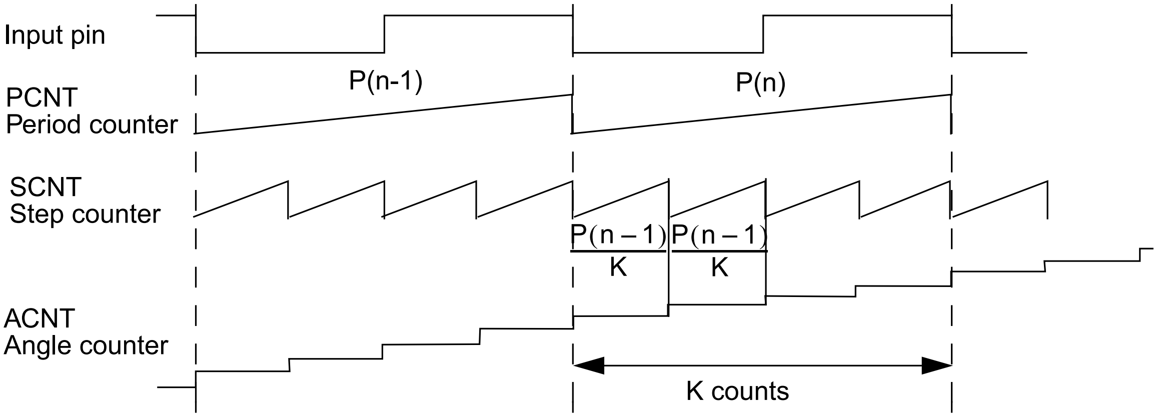
The PCNT calculates the period P(n) between two teeth (two active edges on the toothed wheel input). The active edge (falling or rising) is selected by setting the TED bit in the HWAG global control register 2 (HWAGCR2). On an active edge from the toothed wheel input, the PCNT is saved in the HWAG previous tooth period value register (HWAPCNT1).

The SCNT counts by K steps up to the previous period value, which is contained in the HWAPCNT1 register. When the SCNT overflows PCNT(n-1), an angle tick is generated and SCNT is reset to the remainder between the SCNT and PCNT(n-1). The resulting period of the SCNT is the fraction PCNT(n-1)/K.

The TCKC counts every angle tick until it reaches K and then stops the SCNT. If an active edge occurs before the TCKC has reached K, the remainder is added directly to the ACNT.

When encountering an earlier active edge, the ACNT accumulates the fractions (angle ticks) generated by the SCNT and the remainder of the TCKC. For an example of angle generation using the time-based algorithm, see Figure 23-39.

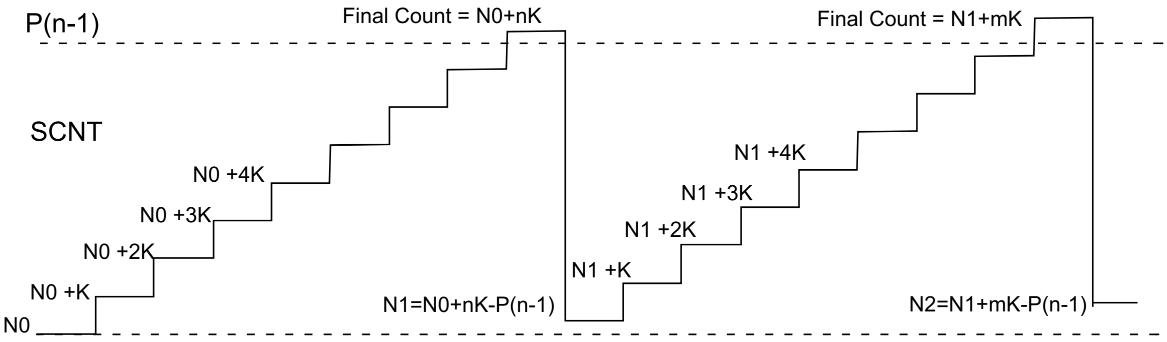
Figure 23-39. Angle Generation Using Time Based Algorithm



Because of stepping, the final count of the SCNT will usually be unequal to the target value PCNT(n-1) and then will overflow. To compensate for this error generated by the algorithm, reset the SCNT to the remainder of the difference between (SCNT - PCNT(n-1)).

To see how the SCNT and PCNT(n-1) generate angle ticks and compensate for the error due to the integer fractions, see Figure 23-40.

Figure 23-40. SCNT Stepping Compensation



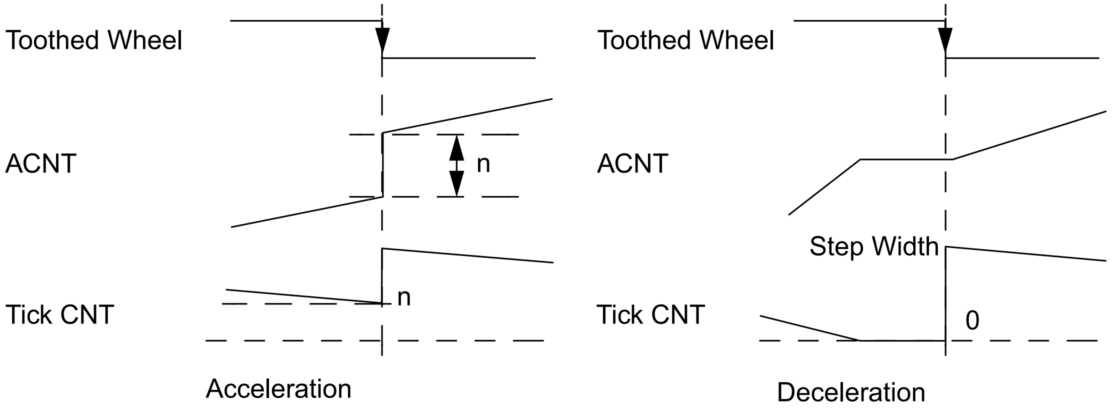
23.3.2.2.1.3 Acceleration and Deceleration

Because the toothed wheel speed is inconstant, it creates discontinuities in the angle counter behavior.

If the TCKC reaches zero before a new active tooth edge during a deceleration, the angle tick signal is no longer generated by the SCNT and PCNT(n-1). This halts the ACNT until the new active tooth arrives.

If the TCKC is unequal to zero when the new active tooth edge arrives during an acceleration (that is, the falling edge on the toothed wheel input in the example below), the rest of the tick counter increments the ACNT. For an example of the ACNT during acceleration and deceleration, see Figure 23-41.

Figure 23-41. ACNT During Acceleration and Deceleration



23.3.2.2.1.4 End of Cycle

The HWAG behaves differently during the singularity tooth period of the toothed wheel. During the singularity period, the HWAG counts three virtual teeth (that is, three times the step width is added to the ACNT) to ensure that the ACNT reaches the maximum value (that is, every angle step has been counted) before resetting it.

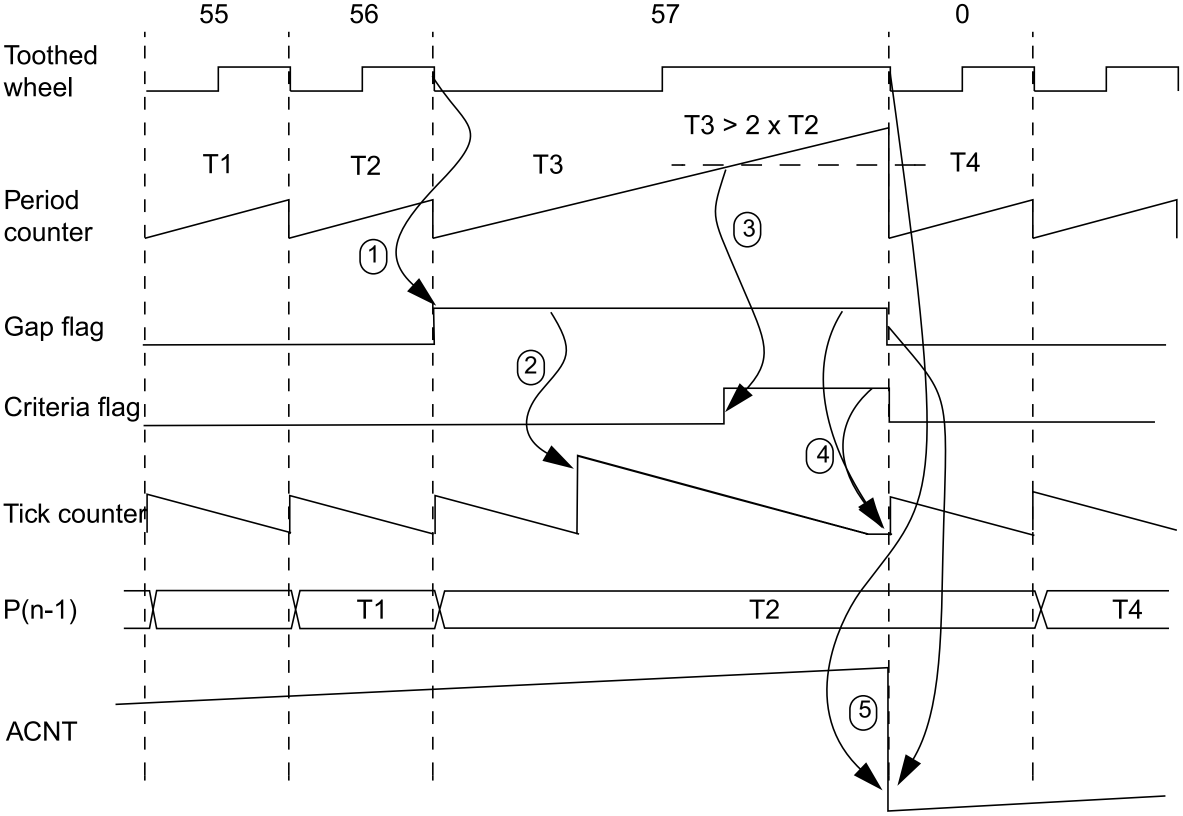
During the singularity period, the HWAG generates angle ticks like for a normal tooth but with three times the value. To generate these angle ticks, the HWAG uses a constant period based on the previous tooth period. Because the period is based on the previous tooth period, the HWAG must recover from a deceleration or acceleration of three teeth when realizing the active edge tooth at the end of the singularity tooth.

The HWAG must ensure that the singularity occurs where expected and must verify it. When the singularity tooth arrives, TCNT reaches the teeth register, sets the signal gap flag, and then keeps PCNT(n-1) until the first tooth of the next round has passed. Because of these conditions, angle ticks before the second tooth will be based on the previous singularity tooth period.

The tick counter is first loaded with a normal value. When the counter reaches zero, it is reloaded once with twice the step width value if the criteria flag is not set. PCNT(n) continues to be incremented and to check the criteria with PCNT(n-1). For more information on gap verification, see Section 23.3.2.2.4. The SCNT continues to generate angle ticks until the tick counter reaches zero the second time. The criteria flag validates the tooth in order to reset the counters. For an example of how the criteria flag validates the tooth to reset the counters, see Figure 23-42.

When the tooth active edge occurs, the ACNT is incremented with the remainder value if the tick counter is not equal to zero. When the ACNT contains a value equals to K times the teeth register, the PCNT, the TCNT and the ACNT are reset to begin a new revolution.

Figure 23-42. Singularity Check, ACNT Reset and Timing Associated



1. When TCNT = teeth register, the Gap flag is raised
2. Tick CNT reloads automatically with 2x the step-width because the Gap flag = ’1’
3. If PCNT (n) > 2 x PCNT (n-1) and the Gap flag = ’1’ then the Criteria flag is raised
4. The tick counter is not reloaded because the Criteria flag is raised
5. The Gap flag and tooth active edge reset, followed by ACNT

23.3.2.2.2 Angle Zero Initialization

Before any angle operation, initialize the HWAG and then initialize the angle zero as the singularity tooth. To initialize the angle zero as the singularity tooth, the HWAG can send an interrupt at each new tooth to help the software detect the first tooth if the interrupt is set. This allows you to decide which algorithm to apply to detect the zero degree tooth (by enabling the corresponding interrupt, you can also use the wired criteria).

When researching which algorithm to apply, the counters ACNT and TCNT are frozen and must be initialized to their start values. The ACNT value is equal to T times the step value (T is the tooth where the start will take effect and the initial value of the tooth counter). The counters PCNT(n) and PCNT(n-1) contain the current period and the previous period respectively. These counters allow you to set a detection criteria. When the application software sets the start bit, the software unfreezes the ACNT and TCNT counters. The counters count from the preloaded values at the next tooth active edge. The ACNT is preloaded with the value of 2 teeth and started synchronously with the next active edge of the toothed wheel. For an example of the HWAG start sequence, see Figure 23-43.

Figure 23-43. Example of HWAG Start Sequence

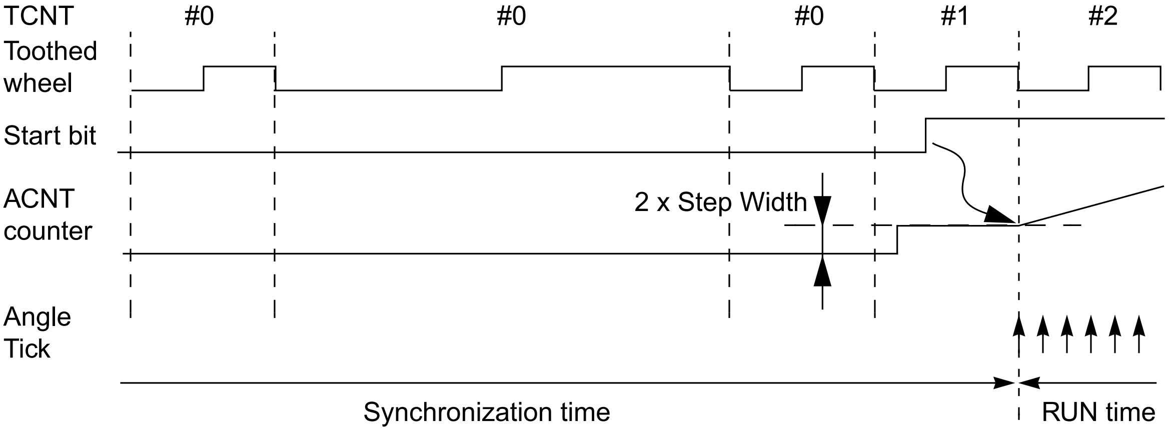
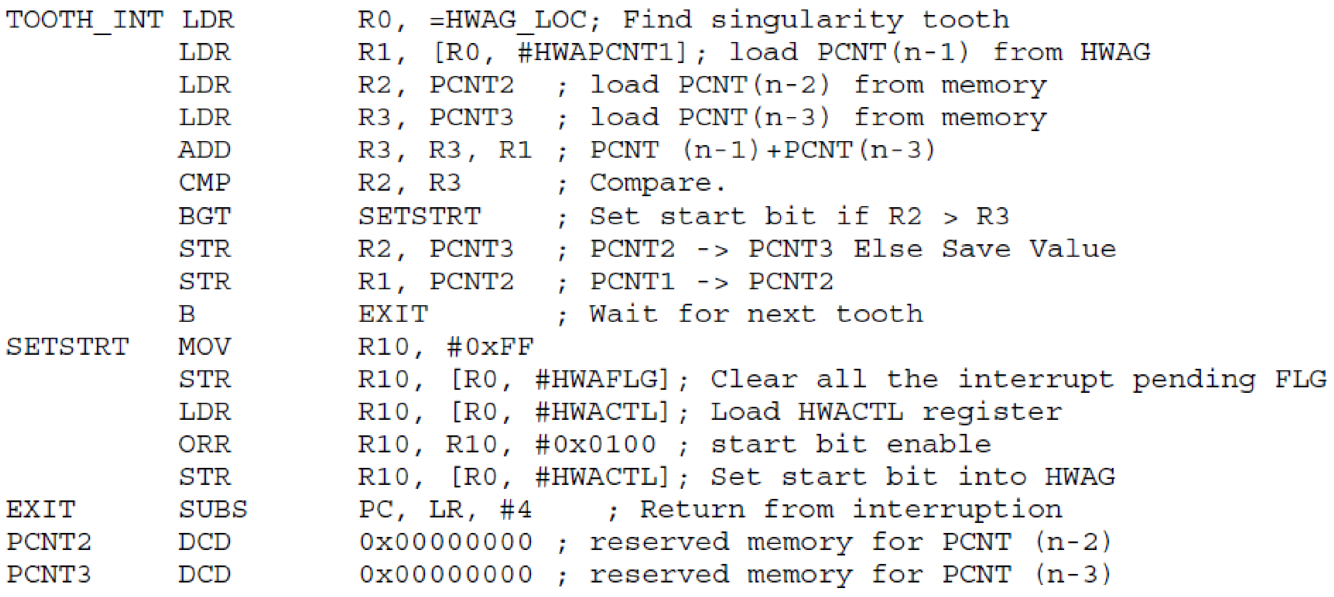


Figure 23-44 is an example of a singularity research initializing the HWAG at the second tooth to start synchronously with the third tooth. The HWAG angle value register (HWAACNT) contains 1024 (2 × 512) and the HWAG current teeth number register (HWATHVL) contains 2.

The code is executed in a tooth interrupt subroutine in code using the PCNT(n-2) > PCNT (n-3) + PCNT (n-1) algorithm.

Figure 23-44. Code



```
TOOTH_INT LDR       R0, =HWAG_LOC; Find singularity tooth
          LDR       R1, [R0, #HWAPCNT1]; load PCNT(n-1) from HWAG
          LDR       R2, PCNT2   ; load PCNT(n-2) from memory
          LDR       R3, PCNT3   ; load PCNT(n-3) from memory
          ADD       R3, R3, R1  ; PCNT (n-1)+PCNT(n-3)
          CMP       R2, R3      ; Compare.
          BGT       SETSTRT     ; Set start bit if R2 > R3
          STR       R2, PCNT3   ; PCNT2 -> PCNT3 Else Save Value
          STR       R1, PCNT2   ; PCNT1 -> PCNT2
          B         EXIT        ; Wait for next tooth
SETSTRT   MOV       R10, #0xFF
          STR       R10, [R0, #HWAFLG]; Clear all the interrupt pending FLG
          LDR       R10, [R0, #HWACTL]; Load HWACTL register
          ORR       R10, R10, #0x0100 ; start bit enable
          STR       R10, [R0, #HWACTL]; Set start bit into HWAG
EXIT      SUBS      PC, LR, #4     ; Return from interruption
PCNT2     DCD       0x00000000 ; reserved memory for PCNT (n-2)
PCNT3     DCD       0x00000000 ; reserved memory for PCNT (n-3)
```

23.3.2.2.3 Stopping the HWAG

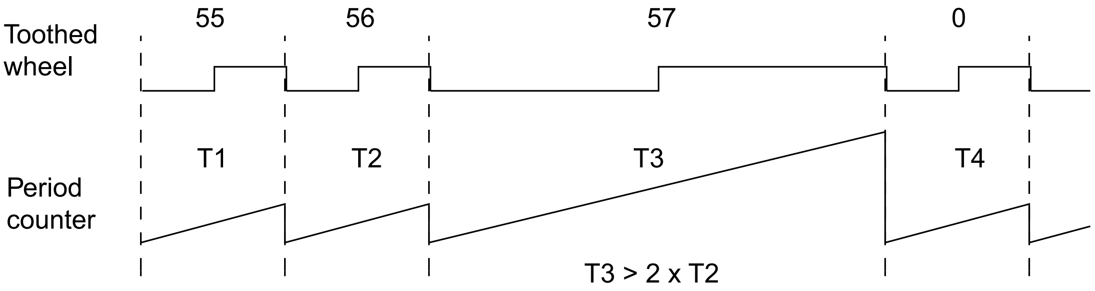
The HWAG starts synchronously with the active edge of the toothed wheel, but stops when the start (STRT) bit in the HWAG global control register 2 (HWAGCR2) is reset. Within a tooth, the HWAG can be stopped and parameters can be changed (that is, step width, angle counter, and so on) If this happens, the restart will take effect on the next active tooth edge.

NOTE: When stopping the HWAG, stop the angle increment delivered to the NHET and set it to zero. Reload the NHET counter with the same value of the angle counter (± corrections), if restarting the HWAG.

23.3.2.2.4 Gap Verification

After the CPU sets the synchronization and puts the HWAG into RUN time (that is, the start bit is set), the tooth counter counts until reaching the teeth register (the number of real teeth of a full wheel revolution). When the tooth counter reaches the teeth register, the gap flag signal is set. For more information on the end of the cycle, see Section 23.3.2.2.1.4. When the gap flag signal is set, it allows the HWAG to verify if the singularity is in the correct position (last tooth). The module then applies the PCNT(n) > 2 x PCNT(n-1) criteria by comparing PCNT(n) and PCNT(n-1) with one bit left shifted. If the criteria does not match when the tooth arrives, then the HWAG sends an interrupt to the CPU and does not reset the ACNT counter. The application software must recover from such an interrupt to keep the HWAG operating optimally. For an example of gap verification criteria for a 60-2 toothed wheel, see Figure 23-45.

Figure 23-45. Gap Verification Criteria For a 60-2 Toothed Wheel



If the hardware criteria is not enabled, you must set the angle reset (ARST) bit in the HWAG global control register 2 (HWAGCR2) to validate the singularity. The HWAGCR2 register must validate the singularity before the active edge of the singularity tooth. If the HWAGCR2 register fails to validate the singularity, the HWAG generates an interrupt and does not clear the ACNT counter when the tooth edge occurs.

NOTE: For a 60-2 toothed wheel, set the ARST flag after the reload of the tick counter(when PCNT(n) = PCNT(n-1)). By verifying the criteria, the application software can set the ARST bit after this point.

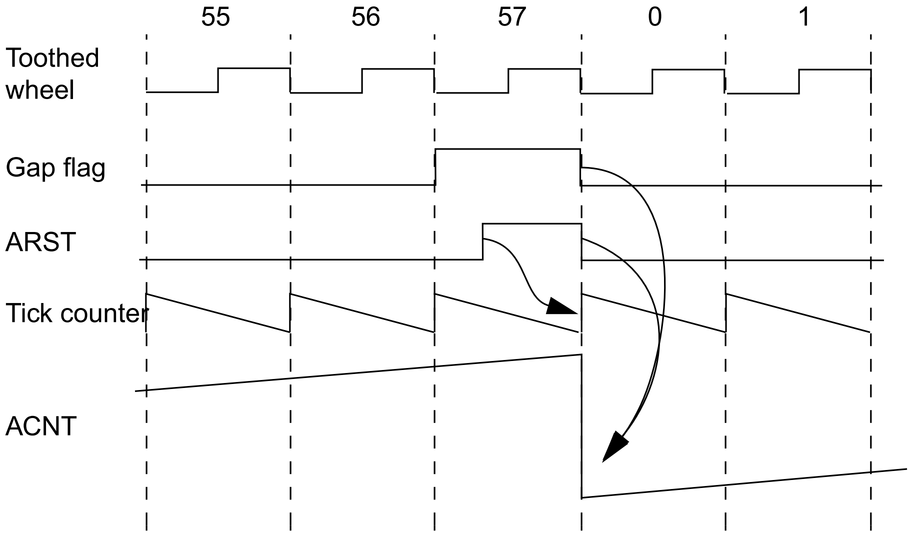
The CPU can read the PCNT counter and make a custom criteria set the ARST bit on time for the HWAG. The application software can use the gap flag interrupt to find the singularity tooth. Alternately, the CPU can verify the validity of the singularity in the second tooth with a more accurate criteria by using the HWAG previous tooth period value register (HWAPCNT1).

23.3.2.2.4.1 Use of the ARST Bit In Case of a Toothed Wheel Without Singularity

If a toothed wheel has no singularity (that is, no missing teeth), the ACNT must be reset when it reaches the angle zero point. To reset the ACNT when it reaches the angle zero point, set the ARST bit to 1.

Setting the ARST bit before the reload of the tick counter will cause the HWAG to fail to reload the tick counter. The HWAG will act like a normal tooth but the next active edge on the toothed wheel input will reset the ACNT and TCNT and clear the ARST bit. For an example of using the ARST bit in a toothed wheel without singularity, see Figure 23-46.

Figure 23-46. Using the ARST Bit in a Toothed Wheel Without Singularity



23.3.2.2.5 Input Noise Filtering

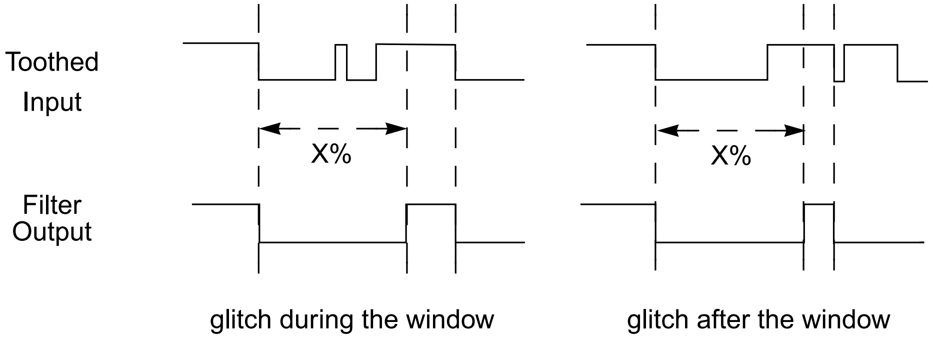
The toothed wheel input comes from an analog part and is sensitive to external noise. Due to this sensitivity, the input needs to be filtered because of glitches in the signal.

The HWAG digitally filters the toothed wheel input signal before it is used inside the core. The filter blocks the signal which negates the effect inside the HWAG. The HWAG provides two filter registers that filter the same way.

The filters validate the input signal after n angle ticks. The n angle ticks are like X% of the tick counter. The value of the remaining percentage of the tick counter (1- X%) need to be set because the tick counter is a down counter. Calculate the value to put into the filter registers from the step width value (or angle ticks value per tooth). The toothed wheel input is like a low pass filter with a cut-off frequency that functions like a toothed-wheel speed, but without acceleration and decelerations side effects. For an example of a windowing filter for a toothed wheel input on a falling active edge, see Figure 23-47.

NOTE: At any time, the CPU can modify the filter values to fine tune with the application.

Figure 23-47. Windowing Filter for Toothed Wheel Input on Falling Active Edge



To calculate this number:

Step Width × (1 – X%) = Filter Register Value

If the step width value is equal to 512 and you want to filter 75% of the tooth, calculate the filter register as follows:

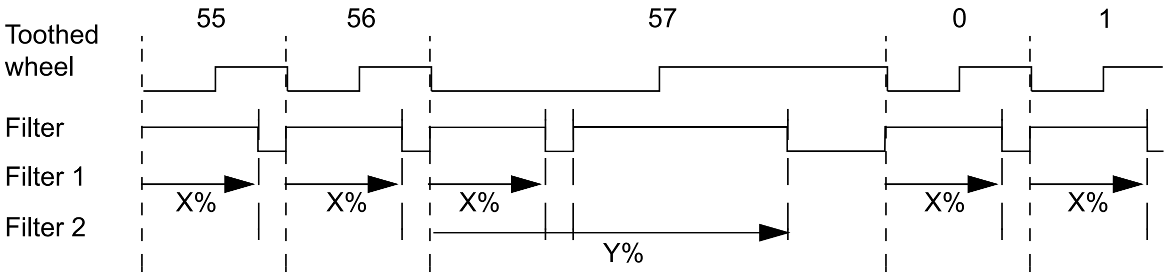
512 × (1 – 0.75) = 128

When the tick counter reaches the filter register value, the toothed wheel input is unblocked.

23.3.2.2.5.1 Filter During Singularity Tooth

During the singularity tooth, the filter acts differently than during a normal tooth. The filter releases the input for a normal tooth. When the tick counter is reloaded, a second filter value is applied to the toothed wheel input. For an example of filtering during a singularity tooth, see Figure 23-48.

Figure 23-48. Filtering During Singularity Tooth



The second filter value is set using the same equation as the first filter with the step width multiplied by 3.

To calculate this number:

$$(3 \times \text{Step Width}) \times (1 - Y\%) = \text{Second Filter Register Value} \quad (34)$$

If the step width value is equal to 512 and you want 70% of singularity tooth period to be filtered, calculate the filter register value as follows:

$$3 \times 512 \times (1 - 0.70) = 460$$

23.3.2.2.6 HWAG Interrupts

When conditions are set, the HWAG interrupts are generated.

When the interrupt condition is true, the corresponding flag is set in the HWAG interrupt flag register (HWAFLG). If the corresponding enable bit in the HWAG interrupt enable set register (HWAENASET) is also set, an interrupt request is sent to the CPU through one of the interrupt lines, depending on the priority of the interrupt (HWAG interrupt level set register (HWALVLSET)).

Because the HWAG can set interruptions, the CPU must determine which source created the interrupt request and then execute the interrupt service routine. The CPU reads the offset register (HWAOFFx) that gives the number of the source. If the CPU reads the offset register, it will automatically clear the source flag that created the request.

NOTE: If the corresponding enable bit is not set, a read in the offset register will not clear a flag. To set the bit, write a 1 in the corresponding bit within the HWAG interrupt flag register (HWAFLG).

The HWAG generates eight different interrupts:

- 0 = Overflow period
- 1 = Singularity not found
- 2 = Tooth interrupt
- 3 = ACNT overflow
- 4 = PCNT(n) > 2 × PCNT (n-1) during normal tooth
- 5 = Bad active edge tooth
- 6 = Gap flag
- 7 = Angle increment overflow

For more information on these interrupts, see Table 23-14. Each interrupt source is associated with a low or high priority. When one or more interrupts with the same priority occur, a fixed priority determines the offset vector if the corresponding enable bits are set.

The HWAG generates two interrupt request signals for the central interrupt module (CIM). For information on servicing interrupts, see Figure 23-49. For a list offset values, see Table 23-13.

Table 23-13. HWAG Interrupt Sources and Offset Values

Source Number	Offset Value
0	1
1	2
:	:
7	8

Figure 23-49. HWAG Interrupt Block Diagram

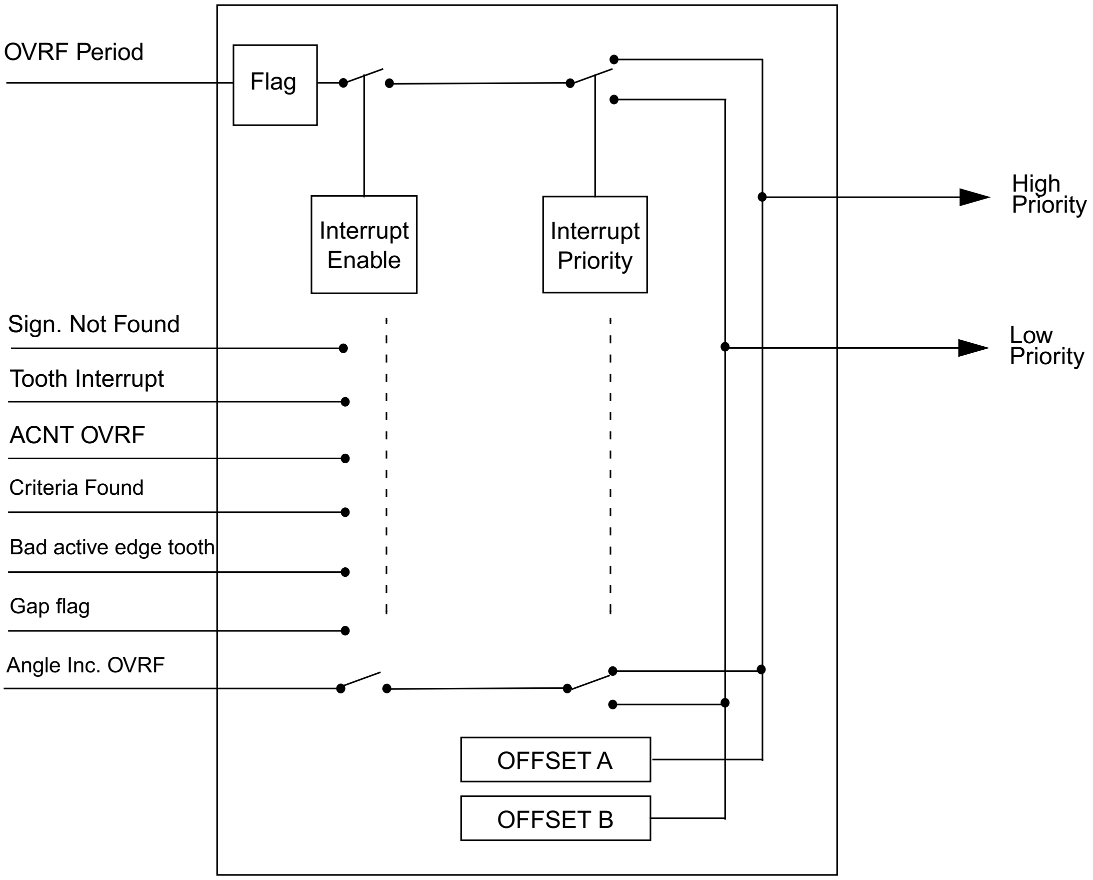


Table 23-14. HWAG Interrupt Descriptions

Interrupt Names	Interrupt Descriptions
Overflow period	Occurs when the PCNT (n) counter reaches the maximum value. Can occur if the toothed wheel input remains stable. May indicate failure of an engine stall or a toothed wheel sensor.
Singularity not found	When the TCNT counter sets the gap flag, the HWAG waits for the criteria flag to raise before the toothed wheel active edge. If the toothed wheel active edge occurs before the criteria flag, the HWAG raises the singularity not found interrupt flag.
New edge tooth	This interrupt can sync or let you control the tick generation. This interrupt indicates the new active edge tooth. This interrupt could be filtered or unfiltered (Bit FIL in control register).
Angle counter (ACNT) overflow	This interrupt occurs when the singularity is unable to be found. The angle counter (ACNT) continues until overflow.
Singularity found during normal tooth	This interrupt indicates that the period of the current tooth is at least two times longer than the previous one when the HWAG expects a normal tooth. This interrupt can detect the singularity without bit manipulation by the CPU.
Bad active edge tooth	This interrupt indicates that an active edge has occurred before the end of the filtering (toothed wheel input blocked) but the HWAG remains inactive internally. This interrupt can detect glitches on the toothed wheel input.
Gap flag	When TCNT reaches the teeth register and the HWAG raises the gap flag , This interrupt is set when the gap flag is raised by the HWAG,
Angle increment overflow	This interrupt indicates that the number of the angle increment is more than 15 since the last resolution tick. This interrupt can prevent any discrepancies between the NHET and the HWAG.

NOTE: Before enabling any interruption, clear the HWAG interrupt flag register (HWAFLG) to ensure that any interrupts have finished. If interrupts are pending, the HWAG could generate an interrupt based on an unrealistic event.

23.3.2.3 Emulation

Because the HWAG is designed to synchronize with a real-time environment, the HWAG counters continue during emulation.

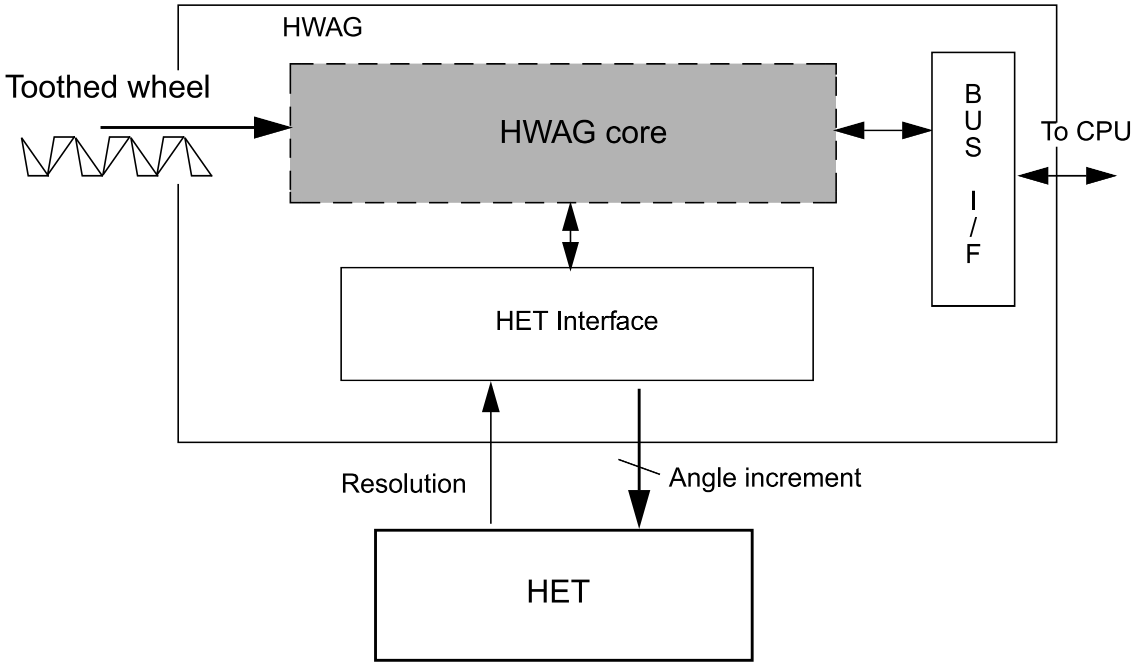
When the CPU is frozen, the HWAG continues to run and update registers. Only the offset registers remain uncleared when entering debug mode.

During debug mode, interrupts can occur and will wait until the CPU enters run mode again. If interrupts occur, they could affect synchronization with the toothed wheel

23.3.2.4 Hardware Angle Generator and High-End Timer

In the engine management application, the HWAG is usually connected to one or more high-end timers. This connection allows you to perform angle compare and angle/time compare. For an example of the hardware angle generator/high-end timer interface, see Figure 23-50.

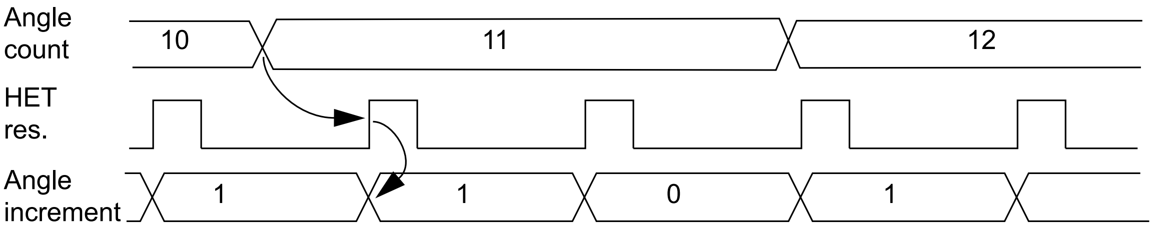
Figure 23-50. Hardware Angle Generator/High End Timer Interface



23.3.2.4.1 Signal Description

To perform a resynchronization, the HWAG interface provides to the NHET at every resolution clock an angle increment value that represents how much the angle counter of the HWAG has been incremented since the last NHET resolution clock. For an example of the angle count within the HWAG, see Figure 23-51.

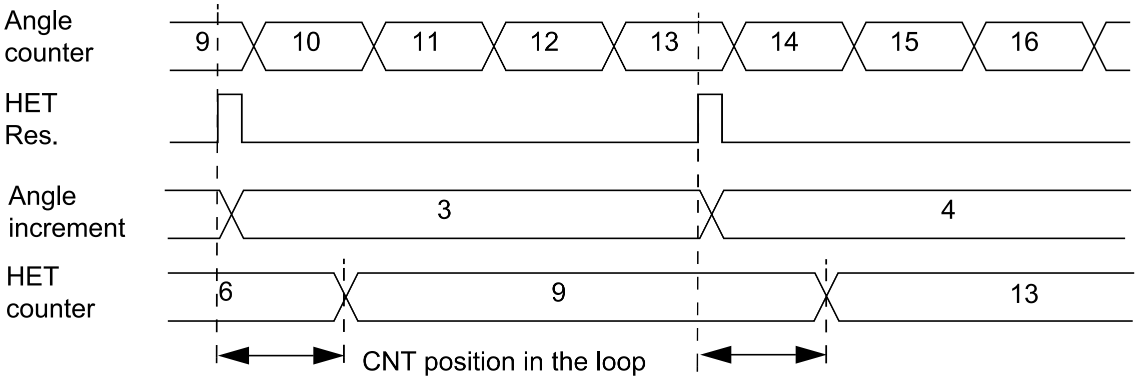
Figure 23-51. Angle Count Within the HWAG at Resolution Clock



When the engine speed increases, the angle count can increment by more than one in a NHET resolution but the HWAG will continue to provide the angle increment value at every resolution..

The NHET can then implement its own angle counter (using a CNT instruction in angle mode) which will be incremented once per resolution by the value given by the angle increment. For an example of an angle count within the NHET with increments, see Figure 23-52.

Figure 23-52. Angle Count Within the NHET With Increments

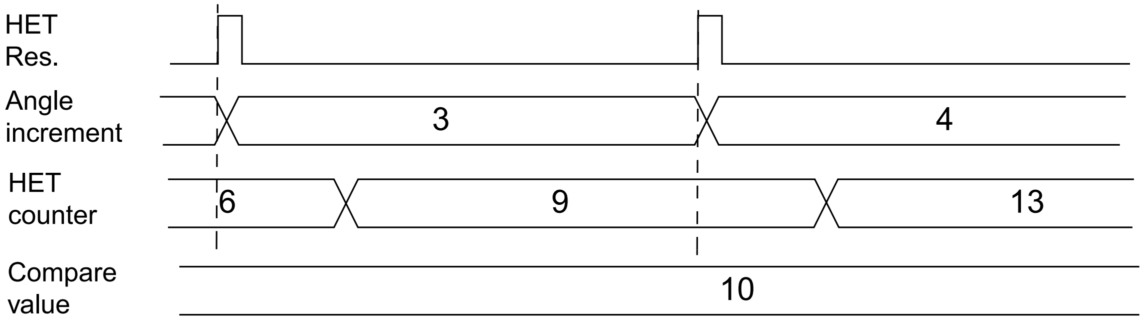


23.3.2.4.2 NHET Operation on Angle Functions (ACMP, CNT)

23.3.2.4.2.1 State of the Art

Because the angle value can be increased by more than one, the compare value could be in-between the old angle value and the new angle value of the NHET angle counter (where new angle value = old angle value + angle increment). To perform an angle compare that ensures not to miss a compare value, the NHET provides the ACMP instruction. For an example of a compare without ACMP instruction, see Figure 23-53.

Figure 23-53. Compare Without ACMP Instruction

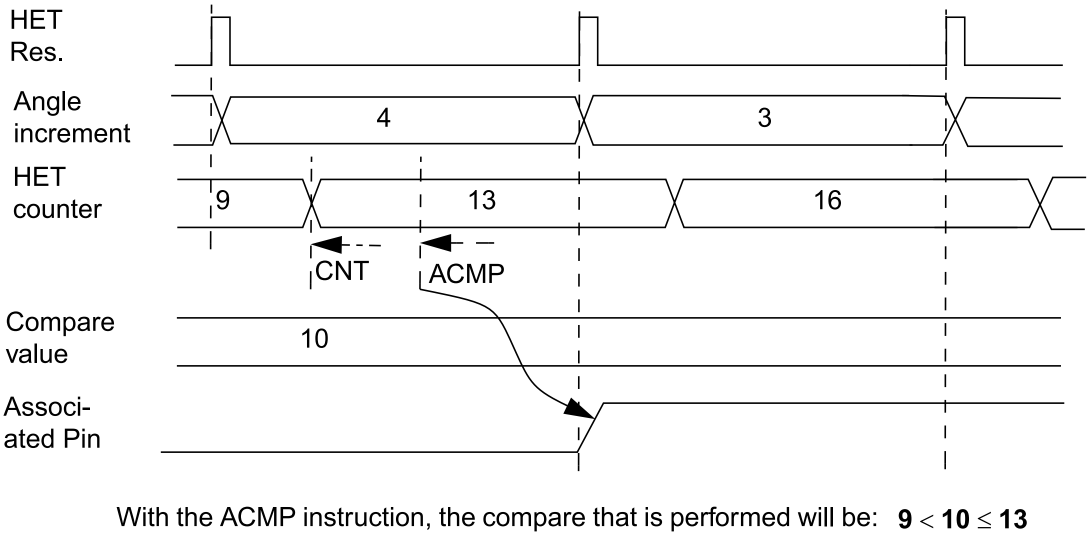


When the HET counter passes from 9 to 13, the equality compare can not match the compare value 10. Consequently, the angle position is missed!

23.3.2.4.2.2 ACMP Instruction Advantage

The ACMP instruction is more than an equality compare. ACMP instruction performs an in-between comparison (old angle value < compare value ≤ new angle value) to match the position of the toothed wheel. This instruction, where an equality compare executes every resolution, may miss a compare match. For an example of ACMP compare within the NHET, see Figure 23-54.

Figure 23-54. Example of ACMP Compare Within the NHET


HET
Res.
Angle
increment
4
3
HET
counter
9
13
16
CNT
ACMP
Compare
value
10
Associ-
ated Pin
With the ACMP instruction, the compare that is performed will be: 9 < 10 ≤ 13

With the ACMP instruction, the compare is: 9 < 10 ≤ 13

NOTE: To avoid multiple matches, the ACMP only matches during a single resolution.

Performing the following equations at the same time implements this compare:

CMP > NHET angle counter – Angle increment

CMP ≤ NHET angle counter

23.3.2.4.3 NHET Interface

23.3.2.4.3.1 Input Signal Selection

The input pin of the toothed-wheel signal is software selectable. In previous generations of NHET/HWAG, this was fixed to HET[2]. On this device, the input pin is programmable to provide more flexibility for the system implementation. However, the implementation is done in a way to be backward compatible.

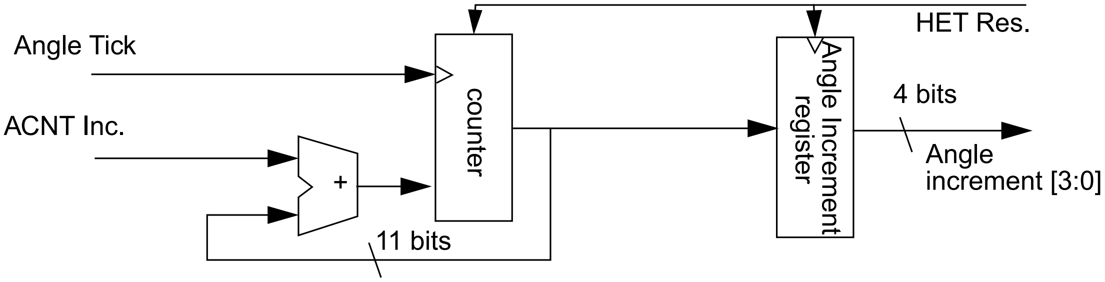
A separate register, HWAG pin select register (HWAPINSEL), is implemented to allow this selection functionality. The HWAPINSEL register should be programmed before the HWAG is turned on. The default selection will be HET[2] (PINSEL = 2h) after reset. The signals will be derived from the input buffer of each pin. This will allow configuring the pin as an output and measure back the output signal with the HWAG.

You can change the HWAPINSEL register at any time, but the proper functionality of the HWAG is not assured if the selection is changed while the HWAG is already operational. It is recommended that the input selection is done before the STRT bit in the HWAG global control register 2 (HWAGCR2)) is programmed to 1.

23.3.2.4.3.2 HWAG to NHET Interface

The NHET interface is a 11-bit counter sampled by the NHET and reset by the NHET resolution. The counter contains the value of ACNT incremented during the last resolution (see Section 23.3.2.4.1). For the NHET interface block diagram, see Figure 23-55.

Figure 23-55. NHET Interface Block Diagram



When the ACNT register is reset to zero, the angle increment register is not reset. The NHET software checks if its own angle register is higher than 360° and either clears it or continues to 720°. If ACNT is reset within the HWAG, the angle increment register gives the NHET the number of angle ticks from the last resolution.

During a strong acceleration after a tooth active edge, the number of angle ticks can exceed 15. If the number of ticks exceeds 15, the HWAG delivers to the NHET several angle increments at 15. This allow the NHET to follow without missing any angle positions from the HWAG. When the counter is below 15, the angle increment reflects the counter. When the angle increment overflows, sets to 15, and if the enable bit (bit 7 in the control register) is set, the HWAG can send an interrupt to the CPU.

During a strong deceleration, the angle increment can stay null for one or more NHET resolution clocks.

To minimize the error between the fly-wheel and NHET angle counter, the step width and the NHET resolution must be set to avoid any overflow of the 11-bit counter of the NHET interface. This can happen if the number of angle ticks always exceeds 15 during one resolution.

23.3.2.5 Range of Operations

23.3.2.5.1 Intrinsic HWAG Limitation

The following factors limit the HWAG:

- SYSCLK
- PCNT counter (overflow)
- Number of teeth
- Angle step

These factors will influence the engine speed range (RPM limitation) and the maximum accuracy of the angle steps (wheel limitation).

- RPM limitation

  The toothed wheel speed is limited by the period counter (PCNT) and the angle step for a given SYSCLK.

  RPM minimum is related to PCNT overflow and SYSCLK.

  Maximum PCNT value × SYSCLK = Maximum tooth period

  $$\text{RPM} = \frac{60}{\text{TeethNumber} \times \text{ToothPeriod}}$$

  PCNT is a 24-bit counter based on SYSCLK.

  RPM maximum is related to the angle step and SYSCLK.

  Minimum tooth period > Step Width × SYSCLK

  The angle ticks period could not be inferior to the SYSCLK.

  Example: The toothed wheel is a 60-2, SYSCLK is 50 Mhz (20 ns), and step width is 512:

  RPM minimum ≥ 16 777 215 × 20 ns = 335.5443 ms ≥ ~3 RPM

  RPM maximum ≥ 512 × 20 ns = 10.24 µs ≥ 97 656 RPM

  NOTE: With a 60-2 toothed wheel, the tooth period is the reverse of the RPM number.

- Wheel Limitation

  The HWAG is limited by the number of teeth and the increments in a revolution.

  The maximum number of teeth is 256. This limits the number of increments per revolution to 512 steps × 256 teeth = 131 072 angle increments.

23.3.2.5.2 HWAG-NHET Limitation

The maximum angle accuracy is a function of the angle step and the NHET loop resolution.

The increment per resolution limits the interface between the HWAG and the NHET. The maximum angle increment per NHET resolution is 15 increments/NHET_res, which is an angular speed. If the angle increment overflows 15 during a constant speed, the system is diverging.

In the HWAG, the angular speed is given by the relation:

$$\text{Angular Speed} = \frac{\text{Step Width}}{\text{Minimum Tooth Period}}$$

To ensure that the values are correct, they must satisfy the following equation:

$$\frac{\text{Max HET resolution} \times \text{Step Width}}{\text{Minimum Tooth Period}} < 15$$

Then,

$$\text{Max HET resolution} = \frac{15 \times \text{Min Tooth Period}}{\text{Step Width}}$$

Example: For a 60-2 at 10000 RPM, the tooth period is 100 µs and the step width is 512:

$$\text{Max HET resolution} = \frac{15 \times 100}{512} = 2.93\,\mu\text{s}$$

23.3.2.6 Tricks

23.3.2.6.1 Using HWAG Previous Tooth Period Value Register (HWAPCNT1)

The HWAG previous tooth period value register (HWAPCNT1) can compensate for errors because of acceleration or deceleration.

If there is a variation of the toothed wheel, the ACNT register will have a discontinuity . For an explanation of acceleration and deceleration, see Section 23.3.2.2.1.3. Avoid this discontinuity by giving the HWAPCNT1 register a smaller or larger value, depending of the variation. When HWAPCNT1 is modified, the angle tick period is also be modified which causes faster or slower tick generation and decreases the discontinuity on the next falling edge.

Because of this compensation, the NHET interface will not overflow and fewer errors will occur on the NHET angle counter in case of strong acceleration.

NOTE: Reading the angle increment will give the application the amount of the acceleration. However, adding the value directly to the NHET counter will result in a discontinuity in the compare sequence. Particularly angle based compare could be missed.

23.3.2.6.2 Using the Singularity During Normal Tooth Interrupt

This interrupt detects if the HWAG is desynchronized with the toothed wheel and resynchronizes the HWAG.

Because the criteria was set during a tooth other than the singularity tooth, the interrupt occurs. Because the criteria is based on PCNT > 2 × PCNT (n-1), this interrupt is likely due to the singularity.

The following steps explain how to resynchronize the HWAG with this interrupt:

1. Stop the HWAG
2. Reset ACNT
3. Reset tooth counter
4. Reset interrupt
5. Set start bit.

The HWAG will restart on the tooth zero.

23.4 N2HET Control Registers

Table 23-15 summarizes all the N2HET registers. The base address for the control registers is FFF7 B800h for N2HET1 and FFF7 B900h for N2HET2.

Table 23-15. N2HET Registers

Offset	Acronym	Register Description	Section
00h	HETGCR	Global Configuration Register	Section 23.4.1
04h	HETPFR	Prescale Factor Register	Section 23.4.2
08h	HETADDR	NHET Current Address Register	Section 23.4.3
0Ch	HETOFF1	Offset Index Priority Level 1 Register	Section 23.4.4
10h	HETOFF2	Offset Index Priority Level 2 Register	Section 23.4.5
14h	HETINTENAS	Interrupt Enable Set Register	Section 23.4.6
18h	HETINTENAC	Interrupt Enable Clear Register	Section 23.4.7
1Ch	HETEXC1	Exception Control Register 1	Section 23.4.8
20h	HETEXC2	Exception Control Register 2	Section 23.4.9
24h	HETPRY	Interrupt Priority Register	Section 23.4.10
28h	HETFLG	Interrupt Flag Register	Section 23.4.11
2Ch	HETAND	AND Share Control Register	Section 23.4.12
34h	HETHRSH	HR Share Control Register	Section 23.4.13
38h	HETXOR	HR XOR-Share Control Register	Section 23.4.14
3Ch	HETREQENS	Request Enable Set Register	Section 23.4.15
40h	HETREQENC	Request Enable Clear Register	Section 23.4.16
44h	HETREQDS	Request Destination Select Register	Section 23.4.17
4Ch	HETDIR	NHET Direction Register	Section 23.4.18
50h	HETDIN	NHET Data Input Register	Section 23.4.19
54h	HETDOUT	NHET Data Output Register	Section 23.4.20
58h	HETDSET	NHET Data Set Register	Section 23.4.21
5Ch	HETDCLR	NHET Data Clear Register	Section 23.4.22
60h	HETPDR	NHET Open Drain Register	Section 23.4.23
64h	HETPULDIS	NHET Pull Disable Register	Section 23.4.24
68h	HETPSL	NHET Pull Select Register	Section 23.4.25
74h	HETPCR	Parity Control Register	Section 23.4.26
78h	HETPAR	Parity Address Register	Section 23.4.27
7Ch	HETPPR	Parity Pin Register	Section 23.4.28
80h	HETSFPRLD	Suppression Filter Preload Register	Section 23.4.29
84h	HETSFENA	Suppression Filter Enable Register	Section 23.4.30
8Ch	HETLBPSEL	Loop Back Pair Select Register	Section 23.4.31
90h	HETLBPDIR	Loop Back Pair Direction Register	Section 23.4.32
94h	HETPINDIS	NHET Pin Disable Register	Section 23.4.33

23.4.1 Global Configuration Register (HETGCR)

N2HET1: offset = FFF7 B800h; **N2HET2:** offset = FFF7 B900h

Figure 23-56. Global Configuration Register (HETGCR) [offset = 00h]

31	25	24
Reserved		HET_PIN_ENA
R-0		R/W-1

23	22	21	20	19	18	17	16
Reserved	MP		Reserved		PPF	IS	CMS
R-0	R/W-0		R-0		R/W-0	R/W-0	R/W-0

15	1	0
Reserved		TO
R-0		R/W-0

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

Table 23-16. Global Configuration Register (HETGCR) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	HET_PIN_ENA		Enables the output buffers of the pin structures depending on the value of nDIS and DIR.x when PINDIS.x is set.
			Note: This bit will automatically get cleared when nDIS pin (input port) value is 0.
		0	No affect on the pin output buffer structure.
		1	Enables the pin output buffer structure when DIR = output, PINDIS.x is set and nDIS = 1.
23	Reserved	0	Reads return 0. Writes have no effect.
22-21	MP		Master Priority
			The NHET can prioritize master accesses to N2HET RAM between the HET Transfer Unit and another arbiter, which outputs the access of one of the remaining masters. The MP bits allow the following selections:
		0	The HTU has lower priority to access the N2HET RAM than the arbiter output.
		1h	The HTU has higher priority to access the N2HET RAM than the arbiter output.
		2h	The HTU and the arbiter output use a round robin scheme to access the N2HET RAM.
		3h	Reserved
20-19	Reserved	0	Reads return 0. Writes have no effect.
18	PPF		Protect Program Fields
			The PPF bit together with the Turn On/Off bit (TO) allows to protect the program fields of all instructions in N2HET RAM.
			When TO = 0:
		0	All masters can read and write the program fields.
		1	All masters can read and write the program fields.
			When TO = 1:
		0	All masters can read and write the program fields.
		1	The program fields are readable but not writable for all masters, which could access the N2HET RAM. Possible masters are the CPU, HTU, DMA and a secondary CPU (if available). Writes initiated by these masters are discarded.
17	IS		Ignore Suspend
			When Ignore Suspend = 0, the timer operation is stopped on suspend (the current timer instruction is completed). Timer RAM can be freely accessed during suspend. When set to 1, the suspend is ignored and the N2HET continues operating.
		0	N2HET stops when in suspend mode.
		1	N2HET ignores suspend mode and continues operation.

Table 23-16. Global Configuration Register (HETGCR) Field Descriptions (continued)

Bit	Field	Value	Description
16	CMS		Clk_master/slave
			This bit is used to synchronize multi-N2HETs. If set (N2HET is master), the N2HET outputs a signal to synchronize the prescalers of the slave N2HET. By default, this bit is reset, which means a slave configuration.
			Note: This bit must be set to one (1) for single-N2HET configuration.
		0	N2HET is configured as a slave.
		1	N2HET is configured as a master.
15-1	Reserved	0	Reads return 0. Writes have no effect.
0	TO		Turn On/Off
			TO does not affect the state of the pins. You must set/reset the timer pins when they are turned off, or re-initialize the timer RAM and control registers before a reset. After a device reset, the timer is turned off by default.
		0	N2HET is OFF. The timer program stops executing. Turn-off is automatically delayed until the current timer program loop is completed. Turn-off does not affect the content of the timer RAM, ALU registers, or control registers. Turn-off resets all flags.
		1	N2HET is ON. The timer program execution starts synchronously to the Loop clock. In case of multiple N2HETs configuration, the slave N2HETs are waiting for the loop clock to come from the master before starting execution. Then, the timer address points automatically address 00h (corresponding to program start).

23.4.2 Prescale Factor Register (HETPFR)

N2HET1: offset = FFF7 B804h; **N2HET2:** offset = FFF7 B904h

Figure 23-57. Prescale Factor Register (HETPFR)

<table>
<tr><td>31</td><td>17</td><td>16</td></tr>
<tr><td colspan="3">Reserved</td></tr>
<tr><td colspan="3">R-0</td></tr>
</table>

<table>
<tr><td>15</td><td>11</td><td>10</td><td>8</td><td>7</td><td>6</td><td>5</td><td>0</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="2">LRPFC</td><td colspan="2">Reserved</td><td colspan="2">HRPFC</td></tr>
<tr><td colspan="2">R-0</td><td colspan="2">R/WP-0</td><td colspan="2">R-0</td><td colspan="2">R/WP-0</td></tr>
</table>

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

Table 23-17. Prescale Factor Register (HETPFR) Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	0	Reads return 0. Writes have no effect.
10-8	LRPFC		Loop-Resolution Pre-scale Factor Code. LRPFC determines the loop-resolution prescale divide rate (lr).
		0	/1
		1h	/2
		2h	/4
		3h	/8
		4h	/16
		5h	/32
		6h	/64
		7h	/128
7-6	Reserved	0	Reads return 0. Writes have no effect.
5-0	HRPFC		High-Resolution Pre-scale Factor Code. HRPFC determines the high-resolution prescale divide rate (hr).
		0	/1
		1h	/2
		2h	/3
		3h	/4
		:	:
		3Dh	/62
		3Eh	/63
		3Fh	/64

23.4.3 N2HET Current Address Register (HETADDR)

N2HET1: offset = FFF7 B808h; **N2HET2:** offset = FFF7 B908h

Figure 23-58. N2HET Current Address (HETADDR)

31	16
Reserved	
R-0	

15	9	8	0
Reserved		HETADDR	
R-0		R-0	

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

Table 23-18. N2HET Current Address (HETADDR) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reads return 0. Writes have no effect.
8-0	HETADDR		N2HET Current Address Read: Returns the current N2HET program address. Write: Writes have no effect.

23.4.4 Offset Index Priority Level 1 Register (HETOFF1)

N2HET1: offset = FFF7 B80Ch; **N2HET2:** offset = FFF7 B90Ch

Figure 23-59. Offset Index Priority Level 1 Register (HETOFF1)

31	16
Reserved	
R-0	

15	6	5	0
Reserved		OFFSET1	
R-0		R-0	

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

Table 23-19. Offset Index Priority Level 1 Register (HETOFF1) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reads return 0. Writes have no effect.
5-0	OFFSET1		OFFSET1 indexes the currently pending high-priority interrupt. Offset values and sources are listed in Table 23-20. Read: Read of these bits determines the pending N2HET interrupt. Write: Writes have no effect. **Note:** In any read operation mode, the corresponding flag (in the HETFLG) is also cleared. In Emulation mode the corresponding flag is not cleared.

Table 23-20. Interrupt Offset Encoding Format

Offset Value	Source No.
0	No interrupt
1	Instruction 0, 32, 64...
2	Instruction 1, 33, 65...
:	:
32	Instruction 31, 63, 95...
33	Program Overflow
34	APCNT Underflow
35	APCNT Overflow

23.4.5 Offset Index Priority Level 2 Register (HETOFF2)

N2HET1: offset = FFF7 B810h; **N2HET2:** offset = FFF7 B910h

Figure 23-60. Offset Index Priority Level 2 Register (HETOFF2)

31	16
Reserved	
R-0	

15	6	5	0
Reserved		OFFSET2	
R-0		R-0	

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

Table 23-21. Offset Index Priority Level 2 Register (HETOFF2) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reads return 0. Writes have no effect.
5-0	OFFSET2		OFFSET2 indexes the currently pending low-priority interrupt. Offset values and sources are listed in Table 23-20. Read: Read of these bits determines the pending N2HET interrupt. Write: Writes have no effect. **Note:** In any read operation mode, the corresponding flag (in the HETFLG) is also cleared. In Emulation mode, the corresponding flag is not cleared.

23.4.6 Interrupt Enable Set Register (HETINTENAS)

N2HET1: offset = FFF7 B814h; **N2HET2:** offset = FFF7 B914h

Figure 23-61. Interrupt Enable Set Register (HETINTENAS)

31	16
HETINTENAS	
R/W-0	

15	0
HETINTENAS	
R/W-0	

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

Table 23-22. Interrupt Enable Set Register (HETINTENAS) Field Descriptions

Bit	Field	Value	Description
31-0	HETINTENAS[n]		Interrupt Enable Set bits. HETINTENAS is readable and writable in any operation mode. Writing a 1 to bit x enables the interrupts of the N2HET instructions at N2HET addresses x+0, x+32, x+64, and so on. Generating an interrupt requires to set bit x in HETINTENAS and to enable the interrupt bit in one of the instructions at addresses x+0, x+32, x+64, and so on. To avoid ambiguity, only one of the instructions x+0, x+32, x+64, and so on, should have the interrupt enable bit (inside the instruction) set. Writing a 0 to HETINTENAS has no effect. When reading from HETINTENAS bit x gives the information, if N2HET instructions x+0, x+32, x+64, and so on, have the interrupt enabled or disabled.
		0	Read: Interrupt is disabled. Write: Writes have no effect.
		1	Read: Interrupt is enabled. Write: Interrupt is enabled.

23.4.7 Interrupt Enable Clear Register (HETINTENAC)

N2HET1: offset = FFF7 B818h; **N2HET2:** offset = FFF7 B918h

Figure 23-62. Interrupt Enable Clear (HETINTENAC)

31	16
HETINTENAC	
R/W-0	

15	0
HETINTENAC	
R/W-0	

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

Table 23-23. NHET Interrupt Enable Clear (HETINTENAC) Field Descriptions

Bit	Field	Value	Description
31-0	HETINTENAC[n]		Interrupt Enable Clear bits. HETINTENAC is readable and writable in any operation mode. Writing a 1 to bit x disables the interrupts of the N2HET instructions at N2HET addresses x+0, x+32, x+64, and so on. (See also description in Table 23-22). Writing a 0 to HETINTENAC has no effect. When reading from HETINTENAC bit x gives the information, if N2HET instructions x+0, x+32, x+64, and so on, have the interrupt enabled or disabled.
		0	Read: Interrupt is disabled. Write: Writes have no effect.
		1	Read: Interrupt is enabled. Write: Interrupt is disabled.

23.4.8 Exception Control Register 1 (HETEXC1)

N2HET1: offset = FFF7 B81Ch; **N2HET2:** offset = FFF7 B91Ch

Figure 23-63. Exception Control Register (HETEXC1)

31–25	24
Reserved	APCNT_OVRFL_ENA
R-0	R/W-0

23–17	16
Reserved	APCNT_UNRFL_ENA
R-0	R/W-0

15–9	8
Reserved	PRGM_OVRFL_ENA
R-0	R/W-0

7–3	2	1	0
Reserved	APCNT_OVRFL_PRY	APCNT_UNRFL_PRY	PRGM_OVRFL_PRY
R-0	R/W-0	R/W-0	R/W-0

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

Table 23-24. Exception Control Register 1 (HETEXC1) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reads return 0. Writes have no effect.
24	APCNT_OVRFL_ENA		APCNT Overflow Enable
		0	APCNT overflow exception is not enabled.
		1	Enables the APCNT overflow exception.
23-17	Reserved	0	Reads return 0. Writes have no effect.
16	APCNT_UNRFL_ENA		APCNT Underflow Enable
		0	APCNT underflow exception is not enabled.
		1	Enables the APCNT underflow exception.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	PRGM_OVRFL_ENA		Program Overflow Enable
		0	The program overflow exception is not enabled.
		1	Enables the program overflow exception.
7-3	Reserved	0	Reads return 0. Writes have no effect.
2	APCNT_OVRFL_PRY		APCNT Overflow Exception Interrupt Priority
		0	Exception priority level 2.
		1	Exception priority level 1.
1	APCNT_UNRFL_PRY		APCNT Underflow Exception Interrupt Priority
		0	Exception priority level 2.
		1	Exception priority level 1.
0	PRGM_OVRFL_PRY		ProgramOverflow Exception Interrupt Priority
		0	Exception priority level 2.
		1	Exception priority level 1.

23.4.9 Exception Control Register 2 (HETEXC2)

N2HET1: offset = FFF7 B820h; **N2HET2:** offset = FFF7 B920h

Figure 23-64. Exception Control Register 2 (HETEXC2)

31	16
Reserved	
R-0	

15	9	8
Reserved		DEBUG_STATUS_FLAG
R-0		R/WC-0

7	3	2	1	0
Reserved		APCNT_OVRFL_FLAG	APCNT_UNRFL_FLAG	PRGM_OVRFL_FLAG
R-0		R/W1C-0	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 23-25. Exception Control Register 2 (HETEXC2) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reads return 0. Writes have no effect.
8	DEBUG_STATUS_FLAG		Debug Status Flag. This flag is set when N2HET has stopped at a breakpoint. Also generates a debug request to halt the ARM CPU.
		0	Read: N2HET is either running, or stopped, flag cleared but not yet restarted. Write: No effect.
		1	Read: N2HET is stopped at a breakpoint. Write: Clears the bit. To restart N2HET clear this bit and then restart the ARM CPU. The N2HET and ARM CPU will start synchronously.
7-3	Reserved	0	Reads return 0. Writes have no effect.
2	APCNT_OVRFL_FLAG		APCNT Overflow Flag
		0	Read: Exception has not occurred since the flag was cleared. Write: No effect.
		1	Read: Exception has occurred since the flag was cleared. Write: Clears the bit.
1	APCNT_UNDFL_FLAG		APCNT Underflow Flag
		0	Read: Exception has not occurred since the flag was cleared. Write: No effect.
		1	Read: Exception has occurred since the flag was cleared. Write: Clears the bit.
0	PRGM_OVERFL_FLAG		Program Overflow Flag
		0	Read: Exception has not occurred since the flag was cleared. Write: No effect.
		1	Read: Exception has occurred since the flag was cleared Write: Clears the bit.

23.4.10 Interrupt Priority Register (HETPRY)

N2HET1: offset = FFF7 B824h; **N2HET2:** offset = FFF7 B924h

Figure 23-65. Interrupt Priority Register (HETPRY)

31	16
HETPRY	
R/WP-0	

15	0
HETPRY	
R/WP-0	

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

Table 23-26. Interrupt Priority Register (HETPRY) Field Descriptions

Bit	Field	Value	Description
31-0	HETPRY[n]		HET Interrupt Priority Level Bits Used to select the priority of any of the 32 potential interrupt sources coming from N2HET instructions.
		0	Interrupt priority level 2 (low level).
		1	Interrupt priority level 1 (high level).

23.4.11 Interrupt Flag Register (HETFLG)

N2HET1: offset = FFF7 B828h; **N2HET2:** offset = FFF7 B928h

Figure 23-66. Interrupt Flag Register (HETFLG)

31	16
HETFLAG	
R/W1C-0	

15	0
HETFLAG	
R/W1C-0	

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset; X = Unknown

Table 23-27. Interrupt Flag Register (HETFLG) Field Descriptions

Bit	Field	Value	Description
31-0	HETFLAG[n]		Interrupt Flag Register Bits Bit x is set when an interrupt condition has occurred on one of the instructions x+0, x+32, x+64, and so on. The flag position x (in the register) is decoded from the five LSBs of the instruction address that generated the interrupt. The hardware will set the flag only if the interrupt enable bit (in the corresponding instruction) is set. The flag will be set even if bit x in the Interrupt Enable Set Register (HETINTENAS) is not enabled. Enabling bit x in HETINTENAS is required if an interrupt should be generated. Clearing the flag can be done by writing a one to the flag. Alternatively reading the corresponding Offset Index Priority Level 1 Register (HETOFF1) or Offset Index Priority Level 2 Register (HETOFF2) will automatically clear the flag.
		0	Read: No N2HET instruction with an interrupt has been reached since the flag was cleared. Write: No effect.
		1	Read: A N2HET instruction with an interrupt has been reached since the flag was cleared. Write: Clears the bit.

23.4.12 AND Share Control Register (HETAND)

N2HET1: offset = FFF7 B82Ch; **N2HET2:** offset = FFF7 B92Ch

Figure 23-67. AND Share Control Register (HETAND)

31–16
Reserved
R-0

15	14	13	12	11	10	9	8
AND SHARE31/30	AND SHARE29/28	AND SHARE27/26	AND SHARE25/24	AND SHARE23/22	AND SHARE21/20	AND SHARE19/18	AND SHARE17/16
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
AND SHARE15/14	AND SHARE13/12	AND SHARE11/10	AND SHARE9/8	AND SHARE7/6	AND SHARE5/4	AND SHARE3/2	AND SHARE1/0
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 23-28. AND Share Control Register (HETAND) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	ANDSHARE n+1 / n		AND Share Enable Enable the AND sharing of the same pin for two HR structures. For example, if bit ANDSHARE1/0 is set, the pin HET[0] will then be commanded by a logical AND of both HR structures 0 and 1. **Note:** If HR AND SHARE bits are used, pins not connected to HR structures (the odd number pin in each pair) can be accessed as general inputs/outputs.
		0	HR Output of HET[n+1] and HET[n] are not AND shared.
		1	HR Output of HET[n+1] and HET[n] are AND shared onto pin HET[n].

23.4.13 HR Share Control Register (HETHRSH)

N2HET1: offset = FFF7 B834h; **N2HET2:** offset = FFF7 B934h

Figure 23-68. HR Share Control Register (HETHRSH)

31 … 16
Reserved
R-0

15	14	13	12	11	10	9	8
HR SHARE31/30	HR SHARE29/28	HR SHARE27/26	HR SHARE25/24	HR SHARE23/22	HR SHARE21/20	HR SHARE19/18	HR SHARE17/16
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
HR SHARE15/14	HR SHARE13/12	HR SHARE11/10	HR SHARE9/8	HR SHARE7/6	HR SHARE5/4	HR SHARE3/2	HR SHARE1/0
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 23-29. HR Share Control Register (HETHRSH) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	HRSHARE n+1 / n		HR Share Bits Enables the share of the same pin for two HR structures. For example, if bit HRSHARE1/0 is set, the pin HET[0] will then be connected to both HR input structures 0 and 1. **Note:** If HR share bits are used, pins not connected to HR structures (the odd number pin in each pair) can be accessed as general inputs/outputs.
		0	HR Input of HET[n+1] and HET[n] are not shared.
		1	HR Input of HET[n+1] and HET[n] are shared; both measure pin HET[n].

23.4.14 XOR Share Control Register (HETXOR)

N2HET1: offset = FFF7 B838h; **N2HET2:** offset = FFF7 B938h

Figure 23-69. XOR Share Control Register (HETXOR)

31 … 16
Reserved
R-0

15	14	13	12	11	10	9	8
XOR SHARE31/30	XOR SHARE29/28	XOR SHARE27/26	XOR SHARE25/24	XOR SHARE23/22	XOR SHARE21/20	XOR SHARE19/18	XOR SHARE17/16
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
XOR SHARE15/14	XOR SHARE13/12	XOR SHARE11/10	XOR SHARE9/8	XOR SHARE7/6	XOR SHARE5/4	XOR SHARE3/2	XOR SHARE1/0
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 23-30. XOR Share Control Register (HETXOR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	XORSHARE n+1 / n		XOR Share Enable Enable the XOR-share of the same pin for two output HR structures. For example, if bit XORSHARE1/0 is set, the pin HET[0] will then be commanded by a logical XOR of both HR structures 0 and 1. **Note:** If XOR share bits are used, pins not connected to HR structures (the odd number pin in each pair) can be accessed as general inputs/outputs.
		0	HR Output of HET[n+1] and HET[n] are not XOR shared.
		1	HR Output of HET[n+1] and HET[n] are XOR shared onto pin HET[n].

23.4.15 Request Enable Set Register (HETREQENS)

N2HET1: offset = FFF7 B83Ch; **N2HET2:** offset = FFF7 B93Ch

Figure 23-70. Request Enable Set Register (HETREQENS)

31	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
REQENA7	REQENA6	REQENA5	REQENA4	REQENA3	REQENA2	REQENA1	REQENA0
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 23-31. Request Enable Set Register (HETREQENS) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	REQENAn		Request Enable Bits
		0	Read: Returns the information that request line n is disabled. Write: Writing a 0 has no effect.
		1	Read: Returns the information that request line n is enabled. Write: Writing a 1 to bit n enables the N2HET request line n. **Note:** The request line can trigger a DMA control packet (DMA channel), an HTU double control packet (DCP) or both simultaneously. The HETREQDS register determines to which module(s) the N2HET request line n is assigned. **Note:** A disabled request line does not memorize old requests. So there are no pending requests to service after enabling request line n.

23.4.16 Request Enable Clear Register (HETREQENC)

N2HET1: offset = FFF7 B840h; **N2HET2:** offset = FFF7 B940h

Figure 23-71. Request Enable Clear Register (HETREQENC)

31	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
REQDIS7	REQDIS6	REQDIS5	REQDIS4	REQDIS3	REQDIS2	REQDIS1	REQDIS0
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 23-32. Request Enable Clear Register (HETREQENC) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	REQDISn		Request Disable Bits
		0	Read: Returns the information that request line n is disabled. Write: Writing a 0 has no effect.
		1	Read: Returns the information that request line n is enabled. Write: Writing a 1 to bit n disables the N2HET request line n.

23.4.17 Request Destination Select Register (HETREQDS)

N2HET1: offset = FFF7 B844h; **N2HET2:** offset = FFF7 B944h

Figure 23-72. Request Destination Select Register (HETREQDS) [offset = FFF7 B844h]

31-24	23	22	21	20	19	18	17	16
Reserved	TDBS7	TDBS6	TDBS5	TDBS4	TDBS3	TDBS2	TDBS1	TDBS0
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

15-8	7	6	5	4	3	2	1	0
Reserved	TDS7	TDS6	TDS5	TDS4	TDS3	TDS2	TDS1	TDS0
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 23-33. Request Destination Select Register (HETREQDS) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23-16	TDBSn		HTU, DMA or Both Select Bits
		0	N2HET request line n is assigned to the module specified by TDS bit n.
		1	N2HET request line n is assigned to both DMA and HTU. TDS bit n is ignored in this case.
15-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	TDSn		HTU or DMA Select Bits **Note:** It must be ensured in the N2HET program, that one request line is triggered by only one N2HET instruction.
		0	N2HET request line n is assigned to HTU (TDBS bit n is zero).
		1	N2HET request line n is assigned to DMA (TDBS bit n is zero).

NOTE: Please refer to the device data sheet how each of the 8 N2HET request lines are connected to these modules. See also Section 23.2.9.

23.4.18 NHET Direction Register (HETDIR)

N2HET1: offset = FFF7 B84Ch; **N2HET2:** offset = FFF7 B94Ch

Figure 23-73. N2HET Direction Register (HETDIR)

31	16
HETDIR	
R/W-0	

15	0
HETDIR	
R/W-0	

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

Table 23-34. N2HET Direction Register (HETDIR) Field Descriptions

Bit	Field	Value	Description
31-0	HETDIR[n]		Data direction of NHET pins
		0	Pin HET[n] is an input (and its output buffer is tristated).
		1	Pin HET[n] is an output.

NOTE: Table 23-9 shows how the register bits of DIR, PULDIS and PULSEL are affecting the N2HET pins.

23.4.19 N2HET Data Input Register (HETDIN)

N2HET1: offset = FFF7 B850h; **N2HET2:** offset = FFF7 B950h

Figure 23-74. N2HET Data Input Register (HETDIN)

31	16
HETDIN	
R-x	

15	0
HETDIN	
R-x	

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

Table 23-35. N2HET Data Input Register (HETDIN) Field Descriptions

Bit	Field	Value	Description
31-0	HETDIN[n]		Data input. This bit displays the logic state of the pin.
		0	Pin HET[n] is at logic low (0).
		1	Pin HET[n] is at logic high (1).

23.4.20 N2HET Data Output Register (HETDOUT)

N2HET1: offset = FFF7 B854h; **N2HET2:** offset = FFF7 B954h

Figure 23-75. N2HET Data Output Register (HETDOUT)

31	16
HETDOUT	
R/W-0	

15	0
HETDOUT	
R/W-0	

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

Table 23-36. N2HET Data Output Register (HETDOUT) Field Descriptions

Bit	Field	Value	Description
31-0	HETDOUT[n]		Data out write. Writes to this bit will only take effect when the pin is configured as an output. The current logic state of the pin will be displayed by this bit even when the pin state is changed by writing to HETDSET or HETDCLR.
		0	Pin HET[n] is at logic low (0).
		1	Pin HET[n] is at logic high (1) if the HETPDR[n] bit = 0 or the output is in high-impedance state if the HETPDR[n] bit = 1.

23.4.21 NHET Data Set Register (HETDSET)

N2HET1: offset = FFF7 B858h; **N2HET2:** offset = FFF7 B958h

Figure 23-76. N2HET Data Set Register (HETDSET)

31	16
HETDSET	
R/WS-0	

15	0
HETDSET	
R/WS-0	

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

Table 23-37. N2HET Data Set Register (HETDSET) Field Descriptions

Bit	Field	Value	Description
31-0	HETDSET[n]		This register allows bits of HETDOUT to be set while avoiding the pitfalls of a read-modify-write sequence in a multitasking environment. Bits written as a logic 1 set the same bit in the HETDOUT register; while bits written as logic 0 leave the same bit in HETDOUT unchanged. Reads from this address return the value of the HETDOUT register.
		0	Write: HETDOUT[n] is unchanged.
		1	Write: HETDOUT[n] is set.

23.4.22 N2HET Data Clear Register (HETDCLR)

N2HET1: offset = FFF7 B85Ch; **N2HET2:** offset = FFF7 B95Ch

Figure 23-77. N2HET Data Clear Register (HETDCLR)

31	16
HETDCLR	
R/WC-0	

15	0
HETDCLR	
R/WC-0	

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

Table 23-38. N2HET Data Clear Register (HETDCLR) Field Descriptions

Bit	Field	Value	Description
31-0	HETDCLR[n]		This register allows bits of HETDOUT to be cleared while avoiding the pitfalls of a read-modify-write sequence in a multitasking environment. Bits written as a logic 1 clear the same bit in the HETDOUT register; while bits written as logic 0 leave the same bit in HETDOUT unchanged. Reads from this address return the value of the HETDOUT register.
		0	Write: HETDOUT[n] is unchanged.
		1	Write: HETDOUT[n] is cleared.

23.4.23 N2HET Open Drain Register (HETPDR)

Values in this register enable or disable the open drain capability of the data pins.
N2HET1: offset = FFF7 B860h; **N2HET2:** offset = FFF7 B960h

Figure 23-78. N2HET Open Drain Register (HETPDR)

31	16
HETPDR	
R/W-0	

15	0
HETPDR	
R/W-0	

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

Table 23-39. N2HET Open Drain Register (HETPDR) Field Descriptions

Bit	Field	Value	Description
31-0	HETPDR[n]		Open drain control for HET[n] pins
		0	The pin is configured in push/pull mode.
		1	The pin is configured in open drain mode. The HETDOUT register controls the state of the output buffer: HETDOUT[n] = 0 The output buffer of pin HET[n] is driven low. HETDOUT[n] = 1 The output buffer of pin HET[n] is tristated.

23.4.24 N2HET Pull Disable Register (HETPULDIS)

Values in this register enable or disable the pull-up/-down functionality of the pins.
N2HET1: offset = FFF7 B864h; **N2HET2:** offset = FFF7 B964h

Figure 23-79. N2HET Pull Disable Register (HETPULDIS)

31	16
HETPULDIS	
R/W-n	

15	0
HETPULDIS	
R/W-n	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset; n is device dependent, see device specific data sheet

Table 23-40. N2HET Pull Disable Register (HETPULDIS) Field Descriptions

Bit	Field	Value	Description
31-0	HETPULDIS[n]		Pull disable for N2HET pins
		0	The pull functionality is enabled on pin HET[n].
		1	The pull functionality is disabled on pin HET[n].

NOTE: See device data sheet for which pins provide programmable pullups/pulldowns.

Table 23-9 shows how the register bits of HETDIR, HETPULDIS, and HETPSL are affecting the N2HET pins.

23.4.25 N2HET Pull Select Register (HETPSL)

Values in this register select the pull-up or pull-down functionality of the pins.
N2HET1: offset = FFF7 B868h; **N2HET2:** offset = FFF7 B968h

Figure 23-80. N2HET Pull Select Register (HETPSL)

31	16
HETPSL	
R/W-0	

15	0
HETPSL	
R/W-0	

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

Table 23-41. N2HET Pull Select Register (HETPSL) Field Descriptions

Bit	Field	Value	Description
31-0	HETPSL[n]		Pull select for NHET pins
		0	The pull down functionality is enabled if corresponding bit in HETPULDIS is 0.
		1	The pull up functionality is enabled if corresponding bit in HETPULDIS is 0.

NOTE: See device data sheet for which pins provide programmable pullups/pulldowns.

Table 23-9 shows how the register bits of HETDIR, HETPULDIS and HETPSL are affecting the N2HET pins.

The information of this register is also used to define the pin states after a parity error:

After a parity error all N2HET pins, which are

1. Defined as output pins in the HETDIR register
2. Not defined as open drain pins (with the HETPDR register)
3. Selected with the HETPPR register, will remain outputs, but automatically change their levels in the following way:
   - If the HETPSL register specifies 0 for the pin, it will switch to low level.
   - If the HETPSL register specifies 1 for the pin, it will switch to high level.

This behavior is independent of the value, which register HETPULDIS specifies for the corresponding pin.

23.4.26 Parity Control Register (HETPCR)

N2HET1: offset = FFF7 B874h; **N2HET2:** offset = FFF7 B974h

Figure 23-81. Parity Control Register (HETPCR)

31–16
Reserved
R-0

15–9	8	7–4	3–0
Reserved	TEST	Reserved	PARITY_ENA
R-0	R/WP-0	R-0	R/WP-5h

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

Table 23-42. Parity Control Register (HETPCR) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reads return 0. Writes have no effect.
8	TEST		Test Bit. When this bit is set, the parity bits are mapped into the peripheral RAM frame to make them accessible by the CPU.
		0	Read: Parity bits are not memory mapped. Write: Disable mapping.
		1	Read: Parity bits are memory mapped. Write: Enable mapping.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	PARITY_ENA		Enable/disable parity checking. This bit field enables or disables the parity check on read operations and the parity calculation on write operations. If parity checking is enabled and a parity error is detected the N2HET_UERR signal is activated.
		5h	Read: Parity check is disabled. Write: Disable checking.
		Others	Read: Parity check is enabled. Write: Enable checking.

NOTE: It is recommended to write Ah to enable error detection, to guard against soft errors flipping PARITY_ENA to a disable state.

23.4.27 Parity Address Register (HETPAR)

N2HET1: offset = FFF7 B878h; **N2HET2:** offset = FFF7 B978h

Figure 23-82. Parity Address Register (HETPAR)

31 … 16
Reserved
R-0

15 … 13	12 … 2	1 … 0
Reserved	PAOFF	Reserved
R-0	R-X	R-0

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

Table 23-43. Parity Address Register (HETPAR) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reads return 0. Writes have no effect.
12-2	PAOFF		Parity Error Address Offset. This register holds the offset address of the first parity error, which is detected in N2HET RAM. This error address is frozen from being updated until it is read by the CPU. During emulation mode, this address is frozen even when read. In case of a N2HET RAM parity error, PAOFF will contain the offset address of the erroneous 32-bit N2HET RAM field counted from the beginning of the N2HET RAM. Examples: The 32-bit program field of instruction 0 will return 0, the 32-bit control field of instruction 0 will return 1, ..., the 32-bit control field of instruction 1 will return 5, and so on. Read: Returns the offset address of the erroneous 32-bit word in bytes from the beginning of the N2HET RAM. Write: Writes have no effect.
1-0	Reserved	0	Reads return 0. Writes have no effect.

NOTE: The Parity Error Address Register will not be reset, neither by PORRST nor by any other reset source.

23.4.28 Parity Pin Register (HETPPR)

N2HET1: offset = FFF7 B87Ch; **N2HET2:** offset = FFF7 B97Ch

Figure 23-83. Parity Pin Register (HETPPR)

31	16
HETPPR	
R/W-0	

15	0
HETPPR	
R/W-0	

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

Table 23-44. Parity Pin Register (HETPPR) Field Descriptions

Bit	Field	Value	Description
31-0	HETPPR[n]		NHET Parity Pin Select Bits. Allows HET[n] pins to be configured to drive to a known state when an N2HET parity error is detected.
		0	Pin HET[n] is not affected by the detection of an N2HET parity error.
		1	Pin HET[n] is driven to a known state when an N2HET parity error is detected. The known state is a function of bits HETDIR[n], HETPSL[n], HETPDR[n] as described in Table 23-45 (this state is also independent of HETPULDIS[n]).

Table 23-45. Known State on Parity Error

HETDIR[n]	HETPDR[n]	HETPSL[n]	Known State on Parity Error
0	x	x	High Impedance
1	0	0	Drive Logic 0
1	0	1	Drive Logic 1
1	1	x	High Impedance

23.4.29 Suppression Filter Preload Register (HETSFPRLD)

N2HET1: offset = FFF7 B880h; **N2HET2:** offset = FFF7 B980h

Figure 23-84. Suppression Filter Preload Register (HETSFPRLD)

31 - 18	17 - 16
Reserved	CCDIV
R-0	R/W-0

15 - 10	9 - 0
Reserved	CPRLD
R-0	R/W-0

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

Table 23-46. Suppression Filter Preload Register (HETSFPRLD) Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reads return 0. Writes have no effect.
17-16	CCDIV		Counter Clock Divider
			CCDIV determines the ratio between the counter clock and VCLK2.
		0	CCLK = VCLK2
		1h	CCLK = VCLK2 / 2
		2h	CCLK = VCLK2 / 3
		3h	CCLK = VCLK2 / 4
15-10	Reserved	0	Reads return 0. Writes have no effect.
9-0	CPRLD		Counter Preload Value
			CPRLD contains the preload value for the counter clock.

23.4.30 Suppression Filter Enable Register (HETSFENA)

N2HET1: offset = FFF7 B884h; **N2HET2:** offset = FFF7 B984h

Figure 23-85. Suppression Filter Enable Register (HETSFENA)

31 - 16
HETSFENA
R/W-0

15 - 0
HETSFENA
R/W-0

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

Table 23-47. Suppression Filter Enable Register (HETSFENA) Field Descriptions

Bit	Field	Value	Description
31-0	HETSFENA[n]		Suppression Filter Enable Bits
			Note: If the pin is configured as an output by the N2HET Direction Register (HETDIR), the filter is automatically disabled independent on the bit in HETSFENA.
		0	The input noise suppression filter for pin HET[n] is disabled.
		1	The input noise suppression filter for pin HET[n] is enabled.

23.4.31 Loop Back Pair Select Register (HETLBPSEL)

Refer to Section 23.2.5.7 for a description of loopback test functions.

N2HET1: offset = FFF7 B88Ch; **N2HET2:** offset = FFF7 B98Ch

Figure 23-86. Loop Back Pair Select Register (HETLBPSEL)

31	30	29	28	27	26	25	24
LBPTYPE31/30	LBPTYPE29/28	LBPTYPE27/26	LBPTYPE25/24	LBPTYPE23/22	LBPTYPE21/20	LBPTYPE19/18	LBPTYPE17/16
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
LBPTYPE15/14	LBPTYPE13/12	LBPTYPE11/10	LBPTYPE9/8	LBPTYPE7/6	LBPTYPE5/4	LBPTYPE3/2	LBPTYPE1/0
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
LBPSEL31/30	LBPSEL29/28	LBPSEL27/26	LBPSEL25/24	LBPSEL23/22	LBPSEL21/20	LBPSEL19/18	LBPSEL17/16
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
LBPSEL15/14	LBPSEL13/12	LBPSEL11/10	LBPSEL9/8	LBPSEL7/6	LBPSEL5/4	LBPSEL3/2	LBPSEL1/0
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 23-48. Loop Back Pair Select Register (HETLBPSEL) Field Descriptions

Bit	Field	Value	Description
31-16	LBPTYPE n+1 / n		Loop Back Pair Type Select Bits These bits are valid only when Loopback mode is enabled (HETLBPDIR[19:16] = 1010).
		0	Digital loopback is selected for HR structures on pins HET[n+1] and HET[n].
		1	Analog loopback is selected for HR structures on pins HET[n+1] and HET[n].
15-0	LBPSEL n+1 / n		Loop Back Pair Select Bits These bits are valid only when Loopback mode is enabled (HETLBPDIR[19:16] = 1010). If bit x is set, the HR structures on pins HET[n+1] and HET[n] are connected in a loop back mode. The direction is given by LBPDIR n+1/n and type is selected by LBPTYPE n+1/n. The pin which is not driven by the N2HET pin actions can still be used as normal GIO pin.

23.4.32 Loop Back Pair Direction Register (HETLBPDIR)

Refer to Section 23.2.5.7 for a description of loopback test functions.

N2HET1: offset = FFF7 B890h; **N2HET2:** offset = FFF7 B990h

Figure 23-87. Loop Back Pair Direction Register (HETLBPDIR)

31 20	19 16
Reserved	LBPTSTENA
R-0	R/WP-5h

15	14	13	12	11	10	9	8
LBPDIR31/30	LBPDIR29/28	LBPDIR27/26	LBPDIR25/24	LBPDIR23/22	LBPDIR21/20	LBPDIR19/18	LBPDIR17/16
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
LBPDIR15/14	LBPDIR13/12	LBPDIR11/10	LBPDIR9/8	LBPDIR7/6	LBPDIR5/4	LBPDIR3/2	LBPDIR1/0
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; WP = Write in privileged mode only; -*n* = value after reset

Table 23-49. Loop Back Pair Direction Register (HETLBPDIR) Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	LBPTSTENA		Loopback Test Enable Key
		5h	Loopback Test is disabled.
		Ah	Loopback Test is enabled.
		Others	Loopback Test is disabled.
15-0	LBPDIR n+1 / n		Loop Back Pair Direction Bits
		0	The HR structures on pins HET[n+1] and HET[n] are internally connected with HET[n] as input and HET[n+1] as output.
		1	The HR structures on pins HET[n+1] and HET[n] connected with HET[n] as output and HET[n+1] as input.

NOTE: The loop back direction can be selected independent on the HETDIR register setting.

23.4.33 N2HET Pin Disable Register (HETPINDIS)

N2HET1: offset = FFF7 B894h; **N2HET2:** offset = FFF7 B994h

Figure 23-88. N2HET Pin Disable Register (HETPINDIS)

31	16
HETPINDIS	
R/W-0	

15	0
HETPINDIS	
R/W-0	

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

Table 23-50. NHET Pin Disable Register (HETPINDIS) Field Descriptions

Bit	Field	Value	Description
31-0	HETPINDIS[n]		N2HET Pin Disable Bits
		0	Logic low: No affect on the output buffer enable of the pin (is controlled by the value of the HETDIR[n] bit).
		1	Logic high: Output buffer of the pin is enabled if pin nDIS = 1, HET_PIN_ENA = 1, and HETDIR = 1; or disabled if nDIS = 0, HETDIR = 0, or HET_PIN_ENA = 0.

23.5 HWAG Registers

Table 23-51 lists the HWAG registers.

Table 23-51. HWAG Registers

Offset	Acronym	Register Description	Section
9Ch	HWAPINSEL	HWAG Pin Select Register	Section 23.5.1
A0h	HWAGCR0	HWAG Global Control Register 0	Section 23.5.2
A4h	HWAGCR1	HWAG Global Control Register 1	Section 23.5.3
A8h	HWAGCR2	HWAG Global Control Register 2	Section 23.5.4
ACh	HWAENASET	HWAG Interrupt Enable Set Register	Section 23.5.5
B0h	HWAENACLR	HWAG Interrupt Enable Clear Register	Section 23.5.6
B4h	HWALVLSET	HWAG Interrupt Level Set Register	Section 23.5.7
B8h	HWALVLCLR	HWAG Interrupt Level Clear Register	Section 23.5.8
BCh	HWAFLG	HWAG Interrupt Flag Register	Section 23.5.9
C0h	HWAOFF0	HWAG Interrupt Offset Register 1	Section 23.5.10
C4h	HWAOFF1	HWAG Interrupt Offset Register 2	Section 23.5.11
C8h	HWAACNT	HWAG Angle Value Register	Section 23.5.12
CCh	HWAPCNT1	HWAG Previous Tooth Period Value Register	Section 23.5.13
D0h	HWAPCNT	HWAG Current Tooth Period Value Register	Section 23.5.14
D4h	HWASTWD	HWAG Step Width Register	Section 23.5.15
D8h	HWATHNB	HWAG Teeth Number Register	Section 23.5.16
DCh	HWATHVL	HWAG Current Teeth Number Register	Section 23.5.17
E0h	HWAFIL	HWAG Filter Register	Section 23.5.18
E8h	HWAFIL2	HWAG Filter Register 2	Section 23.5.19
F0h	HWAANGI	HWAG Angle Increment Register	Section 23.5.20

23.5.1 HWAG Pin Select Register (HWAPINSEL)

Figure 23-89. HWAG Pin Select Register (HWAPINSEL)

31–16
Reserved
R-0

15–5	4–0
Reserved	PINSEL
R-0	R/W-2h

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

Table 23-52. HWAG Pin Select Register (HWAPINSEL) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reads return 0. Writes have no effect.
4-0	PINSEL		HWAG Pin Select. Selects from which NHET pin input buffer the HWAG toothed-wheel signal is derived.
		0	Read: Pin HET[0] is selected. Write: Selects pin HET[0].
		1h	Read: Pin HET[1] is selected Write: Selects pin HET[1].
		2h	Read: Pin HET[2] is selected Write: Selects pin HET[2]. Default after reset for backwards compatibility
		:	:
		1Fh	Read: Pin HET[31] selected Write: Selects pin HET[31].

23.5.2 HWAG Global Control Register 0 (HWAGCR0)

Figure 23-90. HWAG Global Control Register 0 (HWAGCR0)

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 23-53. HWAG Global Control Register 0 (HWAGCR0) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	RESET		HWAG Module Reset.
		0	HWAG module is reset.
		1	HWAG module is not in reset.

23.5.3 HWAG Global Control Register 1 (HWAGCR1)

Figure 23-91. HWAG Global Control Register 1 (HWAGCR1)

31	16
Reserved	
R-0	

15	1	0
Reserved		PPWN
R-0		R/W-0

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

Table 23-54. HWAG Global Control Register 1 (HWAGCR1) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	PPWN	0	HWAG Module Power Down. This bit is implemented for legacy purposes, but has no functionality, however the HWAG module power down is controlled by the NHET power down. The HWAG cannot be powered down separately.

23.5.4 HWAG Global Control Register 2 (HWAGCR2)

Figure 23-92. HWAG Global Control Register 2 (HWAGCR2)

31	25	24	23	18	17	16
Reserved		ARST	Reserved		TED	CRI
R-0		R/W-0	R-0		R/W-0	R/W-0

15	9	8	7	1	0
Reserved		FIL	Reserved		STRT
R-0		R/W-0	R-0		R/W-0

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

Table 23-55. HWAG Global Control Register 2 (HWAGCR2) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	ARST		Angle Reset. This bit is used by the HWAG to validate the singularity when the hardware criteria is not used. The bit is cleared when the HWAG angle value register (HWAACNT) is cleared by the HWAG, when the last tooth edge occurs. If this bit is not set before the tooth edge during an singularity tooth, the HWAG generates an interruption “singularity not found”, if the interrupt is enabled.
		0	Do not reset ACNT once it reaches the angle zero point.
		1	Reset ACNT once it reaches the angle zero point.
23-18	Reserved	0	Reads return 0. Writes have no effect.
17	TED		Tooth Edge. This bit is used to select which edge of the tooth wheel must be considered as active.
		0	Falling edge
		1	Rising edge
16	CRI		Criteria enable. This bits is used to control whether the criteria are applied. You could set your own criteria filter by disabling the hardwired criteria.
		0	Criteria is disabled.
		1	Criteria is enabled.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	FIL		Input Filter Enable. This bit is used to enable the toothed wheel input filter.
		0	Filter is disabled.
		1	Filter is enabled.
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	STRT		Start bit. Put the HWAG into run time. Allows the HWAG to start counting ACNT, TCNT and criteria mechanism (if set). The HWAG starts at the next active edge from the toothed wheel, once set. If the start bit is cleared to 0, the HWAG is stopped immediately.
		0	Do not start counting.
		1	Start counting.

23.5.5 HWAG Interrupt Enable Set Register (HWAENASET)

Figure 23-93. HWAG Interrupt Enable Set Register (HWAENASET)

31-8
Reserved
R-0

7	6	5	4	3	2	1	0
SETINTENA7	SETINTENA6	SETINTENA5	SETINTENA4	SETINTENA3	SETINTENA2	SETINTENA1	SETINTENA0
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 23-56. HWAG Interrupt Enable Set Register (HWAENASET) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	SETINTENA[n]		Enable interrupt. See Table 23-57.
		0	Read: Corresponding interrupt is not enabled. Write: No effect.
		1	Read: Corresponding interrupt is enabled. Write: Enable corresponding interrupt.

Table 23-57. HWAG Interrupts

Bit	Interrupt
0	Overflow period
1	Singularity not found
2	Tooth interrupt
3	ACNT overflow
4	PCNT(n) > 2 x PCNT (n-1) during normal tooth
5	Bad active edge tooth
6	Gap flag
7	Angle increment overflow

23.5.6 HWAG Interrupt Enable Clear Register (HWAENACLR)

Figure 23-94. HWAG Interrupt Enable Clear Register (HWAENACLR)

31 … 8
Reserved
R-0

7	6	5	4	3	2	1	0
CLRINTENA7	CLRINTENA6	CLRINTENA5	CLRINTENA4	CLRINTENA3	CLRINTENA2	CLRINTENA1	CLRINTENA0
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 23-58. HWAG Interrupt Enable Clear Register (HWAENACLR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	CLRINTENA[n]		Disable interrupt. See Table 23-57.
		0	Read: Corresponding interrupt is not enabled. Write: No effect.
		1	Read: Corresponding interrupt is enabled. Write: Disable corresponding interrupt.

23.5.7 HWAG Interrupt Level Set Register (HWALVLSET)

Figure 23-95. HWAG Interrupt Level Set Register (HWALVLSET)

31 … 8
Reserved
R-0

7	6	5	4	3	2	1	0
SETINTLVL7	SETINTLVL6	SETINTLVL5	SETINTLVL4	SETINTLVL3	SETINTLVL2	SETINTLVL1	SETINTLVL0
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 23-59. HWAG Interrupt Level Set Register (HWALVLSET) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	SETINTLVL[n]		Set Interrupt Level. See Table 23-57.
		0	Read: Low-priority interrupt. Write: No effect.
		1	Read: High-priority interrupt. Write: Set interrupt priority to high.

23.5.8 HWAG Interrupt Level Clear Register (HWALVLCLR)

Figure 23-96. HWAG Interrupt Level Clear Register (HWALVLCLR)

31 … 8
Reserved
R-0

7	6	5	4	3	2	1	0
CLRINTLVL7	CLRINTLVL6	CLRINTLVL5	CLRINTLVL4	CLRINTLVL3	CLRINTLVL2	CLRINTLVL1	CLRINTLVL0
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 23-60. HWAG Interrupt Level Clear Register (HWALVLCLR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	CLRINTLVL[n]		Clear Interrupt Level. See Table 23-57.
		0	Read: Low-priority interrupt. Write: No effect.
		1	Read: High-priority interrupt. Write: Set interrupt priority to low.

23.5.9 HWAG Interrupt Flag Register (HWAFLG)

Figure 23-97. HWAG Interrupt Flag Register (HWAFLG)

31-8
Reserved
R-0

7	6	5	4	3	2	1	0
INTFLG7	INTFLG6	INTFLG5	INTFLG4	INTFLG3	INTFLG2	INTFLG1	INTFLG0
R/W1C-0	R/W1C-0	R/W1C-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; -*n* = value after reset

Table 23-61. HWAG Interrupt Flag Register (HWAFLG) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	INTFLG[n]		Interrupt Flag. These bit are set when an interrupt condition has occurred inside the HWAG. The interrupt is sent to the CPU if, and only if, the corresponding enable bit is set. HWAFLG is cleared by either reading the HWAOFF0 or HWAOFF1 register (if the corresponding bit is set) or by writing 1 to the bit. If HWAFLG is 1 but the corresponding interrupt is not enabled then it will not generate an interrupt, also the OFFSET index will not be generated for that particular HWAFLG bit. So, a read of HWAOFF registers will not clear a HWAFLG bit that is not enabled. See Table 23-57.
		0	Read: No interrupt is pending. Write: No effect.
		1	Read: Interrupt is pending. Write: Clear the corresponding interrupt flag.

23.5.10 HWAG Interrupt Offset Register 0 (HWAOFF0)

This register is a read-only register and provides a numerical value that represents the pending interrupt with a high priority. The index can be used to locate the interrupt routine position in the vector table. A read to this register clears the corresponding interrupt pending bit in the HWAG interrupt flag register (HWAFLG). An interrupt pending bit in the HWAFLG register is the bit for which the corresponding interrupt enable bit is set.

During suspend mode, a read to this register does not clear the corresponding interrupt bit.

Figure 23-98. HWAG Interrupt Offset Register 0 (HWAOFF0)

31 … 16
Reserved
R-0

15 … 8	7 … 0
Reserved	OFFSET1
R-0	R-0

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

Table 23-62. HWAG Interrupt Offset Register 0 (HWAOFF0) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	OFFSET1		High-Priority Interrupt Offset. These bits give the offset for the corresponding interrupts.
		0	Phantom interrupt
		1	Overflow period
		2	Singularity not found
		3	Tooth interrupt
		4	ACNT overflow
		5	PCNT(n) > 2 × PCNT (n-1) during normal tooth
		6	Bad active edge tooth
		7	Gap flag
		8	Angle increment overflow

23.5.11 HWAG Interrupt Offset Register 1 (HWAOFF1)

This register is a read-only register and provides a numerical value that represents the pending interrupt with a low priority. The index can be used to locate the interrupt routine position in the vector table. A read to this register clears the corresponding interrupt pending bit in the HWAG interrupt flag register (HWAFLG). An interrupt pending bit in the HWAFLG register is the bit for which the corresponding interrupt enable bit is set.

During suspend mode, a read to this register does not clear the corresponding interrupt bit.

Figure 23-99. HWAG Interrupt Offset Register 1 (HWAOFF1)

31 - 16
Reserved
R-0

15 - 8	7 - 0
Reserved	OFFSET2
R-0	R-0

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

Table 23-63. HWAG Interrupt Offset Register 1 (HWAOFF1) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	OFFSET2		Low-Priority Interrupt Offset.. These bits give the offset for the corresponding interrupts.
		0	Phantom interrupt
		1	Overflow period
		2	Singularity not found
		3	Tooth interrupt
		4	ACNT overflow
		5	PCNT(n) > 2 × PCNT (n-1) during normal tooth
		6	Bad active edge tooth
		7	Gap flag
		8	Angle increment overflow

23.5.12 HWAG Angle Value Register (HWAACNT)

Figure 23-100. HWAG Angle Value Register (HWAACNT)

31 24	23 16
Reserved	ACNT
R-0	R/W-0

15 0
ACNT
R/W-0

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

Table 23-64. HWAG Angle Value Register (HWAACNT) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23-0	ACNT	0-FF FFFFh	Angle Value. Provides the current angle value from the toothed wheel. This is equal to step width × teeth value.

23.5.13 HWAG Previous Tooth Period Value Register (HWAPCNT1)

Figure 23-101. HWAG Previous Tooth Period Value Register (HWAPCNT1)

31 24	23 16
Reserved	PCNT(n-1)
R-0	R/W-0

15 0
PCNT(n-1)
R/W-0

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

Table 23-65. HWAG Previous Tooth Period Value Register (HWAPCNT1) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23-0	PCNT(n-1)	0-FF FFFFh	Period (n-1) Value. Gives the period value of the previous tooth.

23.5.14 HWAG Current Tooth Period Value Register (HWAPCNT)

Figure 23-102. HWAG Current Tooth Period Value Register (HWAPCNT)

31 24	23 16
Reserved	PCNT(n)
R-0	R/W-0

15 0
PCNT(n)
R/W-0

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

Table 23-66. HWAG Current Tooth Period Value Register (HWAPCNT) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23-0	PCNT(n)	0-FF FFFFh	Period (n) Value. Provides the current period since the beginning of the last tooth active edge seen by the HWAG (PCNT (n)). This period would not be accurate due to the fact that the PCNT counter is running at VCLK2 and that the peripheral bus is running at VCLK. Then, the value will have changed when used.

23.5.15 HWAG Step Width Register (HWASTWD)

Figure 23-103. HWAG Step Width Register (HWASTWD)

31	16
Reserved	
R-0	

15	4	3	0
Reserved		STWD	
R-0		R/W-0	

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

Table 23-67. HWAG Step Width Register (HWASTWD) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved		Reads return 0. Writes have no effect.
3-0	STWD		Step Width. Sets the step width for the tick generation, dividing the period into K steps. (131072, 65536, ..., 8, 4). The step count is decoded from the three LSBs using the following encoding:
		0h	4 ticks per period
		1h	8 ticks per period
		2h	16 ticks per period
		:	:
		Eh	65536 ticks per period
		Fh	131072 ticks per period

23.5.16 HWAG Teeth Number Register (HWATHNB)

Figure 23-104. HWAG Teeth Number Register (HWATHNB)

31	16
Reserved	
R-0	

15 8	7 0
Reserved	THNB
R-0	R/W-0

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

Table 23-68. HWAG Teeth Number Register (HWATHNB) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	THNB	0-FFh	Teeth Number. Sets the teeth number with the maximum value of the toothed wheel. This must be equal to N-1 real teeth (that is, 57 for a 60-2 toothed wheel).

23.5.17 HWAG Current Teeth Number Register (HWATHVL)

Figure 23-105. HWAG Current Teeth Number Register (HWATHVL)

31	16
Reserved	
R-0	

15 8	7 0
Reserved	THVL
R-0	R/W-0

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

Table 23-69. HWAG Current Teeth Number Register (HWATHVL) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	THVL	0-FFh	Teeth Value. Provides the current teeth number.

23.5.18 HWAG Filter Register (HWAFIL)

Figure 23-106. HWAG Filter Register (HWAFIL)

31	16
Reserved	
R-0	

15	10	9	0
Reserved		FIL1	
R-0		R/W-0	

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

Table 23-70. HWAG Filter Register (HWAFIL) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return 0. Writes have no effect.
9-0	FIL1	0-3FFh	Filter Value. Contains the value to be compared to the tick counter. It allows the tooth signal to be taken into account by the HWAG. This function works only if the mode filtering is set. The value is calculated as shown in Section 23.3.2.2.5.

23.5.19 HWAG Filter Register 2 (HWAFIL2)

Figure 23-107. HWAG Filter Register 2 (HWAFIL2)

31	16
Reserved	
R-0	

15	12	11	0
Reserved		FIL2	
R-0		R/W-0	

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

Table 23-71. HWAG Filter Register 2 (HWAFIL2) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reads return 0. Writes have no effect.
11-0	FIL2	0-FFFh	Filter Value 2. Contains the value to be compared to the tick counter during the singularity tooth. It allows the tooth signal to be taken into account by the HWAG. This function works only if the mode filtering is set. The value is calculated as shown in Section 23.3.2.2.5.1.

23.5.20 HWAG Angle Increment Register (HWAANGI)

Figure 23-108. HWAG Angle Increment Register (HWAANGI)

31-16
Reserved
R-0

15-10	9-0
Reserved	ANGI
R-0	R-0

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

Table 23-72. HWAG Angle Increment Register (HWAANGI) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return 0. Writes have no effect.
9-0	ANGI	0-3FFh	Angle Increment Value. Provides the current angle increment value. The value is incremented by the tick counter and is decremented by the NHET resolution clock.

23.6 Instruction Set

23.6.1 Instruction Summary

Table 23-73 presents a list of the instructions in the N2HET instruction set. The pages following describe each instruction in detail.

Table 23-73. Instruction Summary

Abbreviation	Instruction Name	Opcode	Sub-Opcode	Cycles[1]
ACMP	Angle Compare	Ch	-	1
ACNT	Angle Count	9h	-	2
ADCNST	Add Constant	5h	-	2
ADC	Add with Carry and Shift	4h	C[25:23] = 011, C5 = 1	1-3
ADD	Add and Shift	4h	C[25:23] = 001, C5 = 1	1-3
ADM32	Add Move 32	4h	C[25:23] = 000, C5 = 1	1-2
AND	Bitwise AND and Shift	4h	C[25:23] = 010, C5 = 1	1-3
APCNT	Angle Period Count	Eh	-	1-2
BR	Branch	Dh	-	1
CNT	Count	6h	-	1-2
DADM64	Data Add Move 64	2h	-	2
DJZ	Decrement and Jump if -zero	Ah	P[7:6] = 10	1
ECMP	Equality Compare	0h	C[6:5] = 00	1
ECNT	Event Count	Ah	P[7:6] = 01	1
MCMP	Magnitude Compare	0h	C[6] = 1	1
MOV32	Move 32	4h	C[5] = 0	1-2
MOV64	Move 64	1h	-	1
OR	Bitwise OR	4h	C[25:23] = 100, C5 = 1	1-3
PCNT	Period/Pulse Count	7h	-	1
PWCNT	Pulse Width Count	Ah	P[7:6] = 11	1
RADM64	Register Add Move 64	3h	-	1
RCNT	Ratio Count	Ah	P[7:6] = 00, P[0] = 1	3
SBB	Subtract with Borrow and Shift	4h	C[25:23] =110, C[5] = 1	1-3
SCMP	Sequence Compare	0h	C[6:5] = 01	1
SCNT	Step Count	Ah	P[7:6] = 00, P[0] = 0	3
SHFT	Shift	Fh	C[3] = 0	1
SUB	Subtract and Shift	4h	C[25:23] = 101, C[5] = 1	1-3
WCAP	Software Capture Word	Bh	-	1
WCAPE	Software Capture Word and Event Count	8h	-	1
XOR	Bitwise Exclusive-Or and Shift	4h	C[25:23] = 111, C[5] = 1	1-3

[1] Cycles refers to the clock cycle of the N2HET module; which on most devices is VCLK2. (Check the device datasheet description of clock domains to confirm). If the high-resolution prescale value is set to /1, then this is also the same as the number of HR clock cycles.

Table 23-74. FLAGS Generated by Instruction

Abbreviation	Flag Name	Set/Reset by	Used by
C	Carry Flag	ADC, ADD, AND, OR, RCNT, SBB, SUB, XOR	ADC, BR, SBB
N	Negative Flag	ADC, ADD, AND, OR, SBB, SUB, XOR	BR
V	Overflow Flag	ADC, ADD, AND, OR, SBB, SUB, XOR	BR
Z	Zero flag	ACNT, ADC, ADD, AND, APCNT, CNT, OR, PCNT, SBB, SCNT, SHFT, SUB, XOR	ACMP, ACNT, BR, ECMP, MCMP, MOV32, RCNT, SCMP, SHFT
X	Angle Compare Match Flag	ACMP	SCMP
SWF 0-1	Step Width flags	SCNT	ACNT
NAF	New Angle Flag	ACNT	NAF_global
NAF_global	New Angle Flag (global)	HWAG or NAF	ACMP, BR, CNT, ECMP, ECNT
ACF	Acceleration Flag	ACNT	,ACNT, SCNT
DCF	Deceleration Flag	ACNT	,ACNT, SCNT
GPF	Gap Flag	ACNT	ACNT, APCNT

The instructions capable of generating software interrupts are listed in Table 23-75.

Table 23-75. Interrupt Capable Instructions

Interrupt Capable Instructions	Non Interrupt Capable Instructions
ACMP	ADC
ACNT	ADCNST
APCNT	ADD
BR	ADM32
CNT	AND
DJZ	DADM32
ECMP	MOV32
ECNT	MOV64
MCMP	OR
PCNT	RADM64
PWCNT	RCNT
SCMP	SBB
SHFT	SCNT
WCAP	SUB
WCAPE	XOR

23.6.2 Abbreviations, Encoding Formats and Bits

Abbreviations marked with a star (*) are available only on specific instructions.

U Reading a bit marked with U will return an indeterminate value.

BRK Defines the software breakpoint for the device software debugger.
Default: OFF
Location: Program field [22]

next Defines the program address of the next instruction in the program flow. This value may be a label or an 9-bit unsigned integer.
Default: Current instruction + 1
Location: Program field [21:13]

reqnum* Defines the number of the request line (0,1,..,7) to trigger either the HTU or the DMA.
Default: 0
Location: Program field [25:23]

request* Allows to select between no request (NOREQ), request (GENREQ) and quiet request (QUIET). See Section 23.2.9.
Default: No request
Location: Control Field [28:27]

Request	C[28]	C[27]	To HTU	To DMA
NOREQ	0	0	no request	no request
	1	0		
GENREQ	0	1	request	request
QUIET	1	1	quiet request	no request

remote* Determines the 9-bit address of the remote address for the instruction.
Default: Current instruction + 1
Location: Program field [8:0]

control Determines whether the immediate data field [31:0] is cleared when it is read. When the bit is not set, reads do not clear the immediate data field.
Default: OFF
Location: Control field [26]

en_pin_action* Determines whether the selected pin is ON so that the action occurs on the chosen pin
Default: OFF
Location: Control field [22]

Cond_addr* Conditional address (optional): Defines the address of the next instruction when the condition occurs.
Default: Current address + 1
Location: Control field [21:13]

Pin* Pin Select: Selects the pin on which the action occurs. Enter the pin number.
Default: pin 0
Location: Control field [12:8] except PCNT

The format CC{pin number} is also supported.

MSB				LSB	Description
0	0	0	0	0	Select HET[0]
0	0	0	0	1	Select HET[1]
(Each pin may be selected by writing its number in binary)					
1	1	1	1	0	Select HET[30]
1	1	1	1	1	Select HET[31]

Reg* Register select: Selects the register for data comparison and storage

Default: No register (None)

Location: Control field [2:1] except for CNT instruction.

Extended Register Select C[7] is available for ACMP, ADC, ADD, ADM32, AND, DADM64, ECMP, ECNT, MCMP, MOV32, MOV64, OR, RADM64, SBB, SHFT, SUB, WCAP, WCAPE instructions.

Register	Ext Reg. C[7]	C[2]	C[1]
A	0	0	0
B	0	0	1
T	0	1	0
None	0	1	1
R	1	0	0
S	1	0	1
Reserved	1	1	0
Reserved	1	1	1

Action* (2 Action Option) Either sets or clears the pin

Default: Clear

Location: Control Field [4]

Action	C[4]
Clear	0
Set	1

Action* (4 Action Option) Either sets, clears, pulse high or pulse low on the pin. Set/clear are single pin actions, pulse high/low include the opposite pin action.

Default: Clear

Location: Control Field [4:3]

Action	Action Type	C[4]	C[3]
Clear	Set low on match	0	0
Set	Set high on match	1	0
Pulse Low	Set low on match + reset to high on Z=1 (opposite action)	0	1
Pulse High	Set high on match + reset to low on Z=1 (opposite action)	1	1

hr_lr* Specifies HIGH/LOW data resolution. If the hr_lr field is HIGH, the instruction uses the hr_data field. If the hr_lr field is LOW, the hr_data field is ignored.

Default: HIGH

Location: Program Field [8]

hr_lr	Prog. field [8]
LOW	1
HIGH	0

prv* Specifies the initial value defining the previous bit (see Section 23.2.5.8). A value of ON sets the previous pin-level bit to 1. A value of OFF sets the initial value of the previous (prv) bit to 0. The prv bit is overwritten (set or reset) by the N2HET the first time the instruction is executed.

Default: OFF

Location: Control Field [25]

cntl_val* Available for DADM64, MOV64, and RADM64, this bit field allows the user to specify the replacement value for the remote control field.

comp_mode* Specifies the compare mode. This field is used with the 64-bit move instructions. This field ensures that the sub-opcodes are moved correctly.

Default: ECMP

Location: Control Field [6:5]

Action	C[6]	C[5]	Order
ECMP	0	0	
SCMP	0	1	
MCMP1	1	0	REG_GE_DATA
MCMP2	1	1	DATA_GE_REG

23.6.3 Instruction Description

The following sections provide information for individual instructions.

Parameters in [] are optional. Refer to the N2HET assembler user guide for the default values when parameters are omitted.

23.6.3.1 ACMP (Angle Compare)

Syntax

ACMP {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}]
[request={NOREQ | GENREQ | QUIET}]
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
[cond_addr={label | 9-bit unsigned integer}]
pin={pin number}
[action={CLEAR | SET}]
reg={A | B | R | S | T | NONE}
[irq ={OFF | ON}]
data={25-bit unsigned integer}
}

Figure 23-109. ACMP Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8 0
0	Request Number	BRK	Next program address	1100	Reserved
6	3	1	9	4	9

Figure 23-110. ACMP Control Field (C31:C0)

31 29	28 27	26	25	24 23	22	21 16
Reserved	Request type	Control	Cout prv	Reserved	En. pin action	Conditional address
3	2	1	1	2	1	9

15 13	12 8	7	6 5	4	3	2 1	0
Conditional address	Pin select	Ext. Reg	Reserved	Pin action	Res.	Register select	Int. ena
9	5	1	2	1	1	2	1

Figure 23-111. ACMP Data Field (D31:D0)

31 7	6 0
Data	Reserved
25	7

Cycles One

Register modified Selected register (A, B, R, S, or T)

The purpose of the comparison is to assert pin action when the angle compare value lies between the old counter value and the new counter value (held in the selected register). Since the angle increment varies from one loop resolution clock to another, an exact equality test cannot be applied. Instead, the following inequality is used to determine the occurrence of a match:

Old counter value < Angle compare value ≤ New counter value

This is done by performing following comparisons:

Selected register value minus angle increment < angle compare value
Angle compare value ≤ Selected register value

register Register B is recommended for typical applications with ACMP.

irq Specifies whether or not an interrupt is generated. Specifying ON generates an interrupt when the edge state is satisfied and the gap flag is set. Specifying OFF prevents an interrupt from being generated.

Default: OFF.

data Specifies the 25-bit angle compare value.

Execution

```
X = 0;
If (Data <= Selected Register)
      Cout = 0;
else
      Cout = 1;
If (Z == 0 AND (Selected Register - Angle Inc. < Data ) AND Cout == 0) OR

 (Z == 1 AND (Cout_prv == 1 OR Cout == 0)))
{
      X = 1;
      If (Enable Pin Action == 1)
          Selected Pin = Pin Action AT next loop resolution clock;

      If (Interrupt Enable == 1)
          HETFLG[n] = 1;   /* n depends on address */
      If ([C28:C27] == 01)
          Generate request on request line [P25:P23];
      If ([C28:C27] == 11)
          Generate quiet request on request line [P25:P23];

      Jump to Conditional Address;
 }

 else
      Jump to Next Program Address;

Cout_prv = Cout (always executed)
```

NOTE: Carry-Out Signal (Cout)

Cout is the carry-out signal of the adder. Even if it is not a flag, it is valid all along ACMP instruction execution.

Angle inc. = NAF_global or hardware angle generator 11-bit input.

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 23.2.7.

23.6.3.2 ACNT (Angle Count)

Syntax

ACNT {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}]
[request={NOREQ | GENREQ | QUIET}]
edge={RISING | FALLING}
[irq ={OFF | ON}]
[control={OFF | ON}]
[prv={OFF | ON}]
gapend ={25-bit unsigned integer}
data={25-bit unsigned integer}
}

Figure 23-112. ACNT Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8	7 1	0
0	Request Number	BRK	Next program address	1001	Edge select	Reserved	Int. ena
6	3	1	9	4	1	7	1

Figure 23-113. ACNT Control Field (C31:C0)

31 29	28 27	26	25	24 0
Res.	Request type	Control	Prv.	Gap End
3	2	1	1	25

Figure 23-114. ACNT Data Field (D31:D0)

31 7	6 0
Data	Reserved
25	7

Cycles

Two, as follows:

- First cycle: Angle increment condition and gap end comparison.
- Second cycle: Gap start comparison.

Register modified

Register B (angle value)

Description

This instruction defines a specialized virtual timer used after SCNT and APCNT to generate an angle-referenced time base that is synchronized to an external signal (that is, a toothed wheel signal). ACNT uses pin HET[2] exclusively. The edge select must be the same as the HET[2] edge which was selected in the previous APCNT.

ACNT refers to the same step width selection that the previous SCNT saved in flags SWF0 and SWF1 (see information on SCNT).

ACNT detects period variations of the external signal measured by APCNT and compensates related count errors.

A period increase is flagged in the deceleration flag (DCF). A period decrease is flagged in the acceleration flag (ACF). If no variation is detected, ACNT increments the counter value each time SCNT reaches its target.

If acceleration is detected, ACNT increments the counter value on each timer resolution. If deceleration is detected ACNT does not increment and is thus saturated.

ACNT also specifies the gap end angle value defining the end value of a gap range in ACNT where period measurements in APCNT are temporarily stopped to mask singularities in the external signal. ACNT uses register A containing gap start and register B to store the counter value.

Edge Specifies the edge for the input capture pin (HET[2]).

Action	P8	Edge Select
Rising	1	Detects a rising edge of HET[2]
Falling	0	Detects a falling edge of HET[2]

irq ON generates an interrupt when the edge state is satisfied and the gap flag is set. OFF prevents an interrupt from being generated.

Default: OFF.

gapend Defines the 25-bit end value of a gap range. The start value is defined in the SCNT instruction.

GAPEND = (Step Value * (# of teeth on the toothed wheel + # of missing teeth)) - 1

data Specifies the 25-bit initial count value for the data field.

Default: 0.

NOTE: Target Edge Field

The target edge field represents the three LSBs of data field register in case of step width = 8, four LSBs for step width = 16, five LSBs for step width = 32 and six LSBs for step width = 64.

Execution

```
Increment Condition: ((Z = 1 AND DCF = 0) OR ACF = 1)
Pin Edge Condition: Specified edge detected on HET[2]
Target Edge Condition: (Target Edge field in data field = 0) AND (Angle
Increment condition is true) AND (GPF = 0)

If (Angle Increment Condition) is false
{
      NAF = 0;
      Register B = Data field register;
}
else
{
      NAF = 1;
      If (Counter value != GapEnd)
      {
           Register B = Data field register + 1;
           Data Field Register = Counter value + 1;
      }
      else
      {
           Register B = 0;
           Data Field Register = 0;
           If (ACF == 0) DCF = 1;
      }
}

Z = 0;

If (Data field register == GapStart)
{
       GPF = 1;
       If (Target Edge condition is true)
       {
           ACF = 0;
           If ((specified edge is not detected on pin HET[2]) AND (data
           field register != 0) AND (ACF == 0) AND (angle increment condition
           is true))
             DCF = 1;
       }
       If (specified edge is detected on pin HET[2])
       {
           DCF = 0;
           If ((target_edge_field != 0) AND (DCF == 0)) ACF = 1;
           If (GPF == 1)
           {
             GPF = 0;
             Z = 1;
             If (Interrupt Enable == 1)
                HETFLG[n] = 1;      /* n depends on address */
             If ([C28:C27] == 01)
                Generate request on request line [P25:P23];;
             If ([C28:C27] == 11)
                Generate quiet request on request line
                [P25:P23];
           }
        }
}

If ((target_edge_field != 0) and (pin_edge_cond == 1))
{
   pin_update = 0;
}
else if (target_edge_field == 0)
{
   pin_update = 1;
}

If (pin_update is true in next loop clock cycle)
{
   Prv bit = Current Lx value of HET[2] pin;
}

Jump to next program address;
```

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 23.2.7.

23.6.3.3 ADCNST (Add Constant)

Syntax

ADCNST {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[control={OFF | ON}]
remote={label | 9-bit unsigned integer}
min_off={25-bit unsigned integer}
data={25-bit unsigned integer}
[hr_data={7-bit unsigned integer}]
}

Figure 23-115. ADCNST Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8 0
0	Reserved	BRK	Next program address	0101	Remote address
6	3	1	9	4	9

Figure 23-116. ADCNST Control Field (C31:C0)

31 27	26	25	24 0
Reserved	Control	Res.	Minimum offset
5	1	1	25

Figure 23-117. ADCNST Data Field (D31:D0)

31 7	6 0
Data	HR Data
25	7

Cycles Two

Register modified Register T (implicity)

Description ADCNST is an extension of ADM32. ADCNST first checks whether the data field value at the remote address is zero; it then performs different adds and moves on the result. ADCNST is typically used to extend the counter value of PWCNT.

min_off A 25-bit constant value that is added to the data field value if the remote data field is null.

data A 25-bit value that is always added to the remote data field.
Default: 0.

hr_data Seven least significant bits of the data addition to the remote data field.
Default: 0.

Figure 23-118 and Figure 23-119 illustrate the behavior of ADCNST if the remote data field is zero or is not zero.

Figure 23-118. ADCNST Operation If Remote Data Field[31:7] Is Not Zero

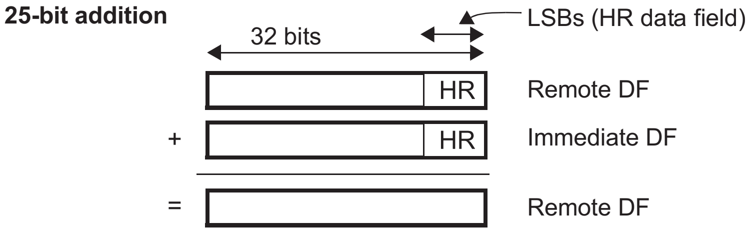
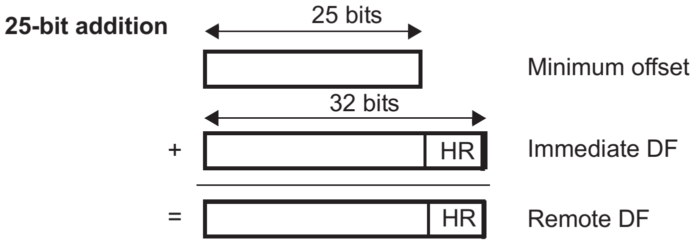


Figure 23-119. ADCNST Operation if Remote Data Field [31:7] Is Zero



Execution

```
If (Remote Data Field Value [31:7] != 0)
      Remote Data Field = Immediate Data Field + Remote Data Field;
else
      Remote Data Field = Immediate Data Field + min. offset(bits C24:C0);

Jump to Next Program Address;
```

23.6.3.4 ADC, ADD, AND, OR, SBB, SUB, XOR

Syntax

ADC | ADD | AND | OR | SBB | SUB | XOR {
src1 = { ZERO | IMM | A | B | R | S | T | ONES | REM | REMP }
src2 = { ZERO | IMM | A | B | R | S | T | ONES }
dest = { NONE | IMM | A | B | R | S | T }
[rdest = { NONE | REM | REMP }]
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[remote={label | 9-bit unsigned integer}]
[control={OFF | ON}]
[init={OFF | ON}]
[smode = {LSL | CSL | LSR | CSR | RR | CRR | ASR }]
[scount = {5 bit unsigned integer}]
[data={25-bit unsigned integer}]
[hr_data={7-bit unsigned integer}]
}

Figure 23-120. ADC, ADD, AND, OR, SBB, SUB, XOR Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8 0
0	Reserved	BRK	Next program address	0100	Remote address
6	3	1	9	4	9

Figure 23-121. ADC, ADD, AND, OR, SBB, SUB, XOR Control Field (C31:C0)

31 27	26	25 23	22 19	18 16
Reserved	Control	Sub Opcode	Src1	Src2
5	1	3	4	3

15 13	12 8	7	6	5	4 3	2 1	0
Smode	Scount	Ext. Reg	Init flag	1	Rdest	Register select	Res.
3	5	1	1	1	2	2	1

Figure 23-122. ADC, ADD, AND, OR, SBB, SUB, XOR Data Field (D31:D0)

31 7	6 0
Data	HR Data
25	7

Cycles One to three cycles, depending on operands selected. (See Table 23-80)

Register modified Selected register (A, B, R, S, T, or NONE)

Description This instruction performs the specified 32-bit arithmetic or logical operation on operands src1 and src2, followed by an optional shift/rotate step. The result of this operation is then stored to either an N2HET register or the immediate data field of the instruction. In addition, the same result may be stored in a remote data field or the least signficant bits of a remote instruction program field (P[8:0]). Bits P[8:0] of the program field are used by most instructions formats to hold the remote address that the instruction operates on, so the ability to update this field programatically makes it easier to write subroutines that operate on different data sets.

The Sub-Opcode field C[25:3] determines which type of operation (ADD, ADC, AND, OR, SBB, SUB, XOR) is executed by the instruction. A list of these operations and the corresponding Sub-Opcode encoding can be found in Table 23-76.

All arithmetic is performed using 32-bit integer math. However, source and destination operands vary in width and can be 9 bits (REMP), 25 bits (A, B) or 32 bits (R,S,T, IMM, REM). Source operands REMP, A,B are extended to 32-bits before being operated on. Also the result of the computation needs to be truncated before being written back to REMP, A, or B when these are selected as destination operands. Table 23-77 provides a list of source operand options, how they are expanded to 32-bit integers (if applicable) and the control field encoding to select the option for src1 and src2 operands.

Table 23-78 provides a similar list of destination operands and their encodings. Up to two destination operands may be selected for each instruction, a register/immediate destination and a remote destination may be selected simultaneously. Truncation is performed independently for each destination operand as appropriate to its size.

An optional shift step following the arithmetic or logical operation may be selected through the smode and scount operands. The shift or rotate type is selected by the smode field; Table 23-79 illustrates the options that are available for smode. The number of bits shifted is determined by the scount operand.

Table 23-76. Arithmetic / Bitwise Logic Sub-Opcodes

Instruction	Description	Operation	Sub-Opcode
ADC	Add with Carry	result = src1 + src2 + C	C[25:23] = 011
ADD	Add	result = src1 + src2	C[25:23] = 001
AND	Bitwise Logic And	result = src1 & src2	C[25:23] = 010
OR	Bitwise Logic Or	result = src1 | src2	C[25:23] = 100
SBB	Subtract with Borrow	result = src1 - src2 - C	C[25:23] = 110
SUB	Subtract	result = src1 - src2	C[25:23] = 101
XOR	Bitwise Logic Exclusive Or	result = src1 ^ src2	C[25:23] = 111

Table 23-77. Source Operand Choices

Source Operand	32-bit value	Address	src1	src2
A	{A[24:0], 0x00}	n/a	C[22:19] = 0010	C[18:16] = 010
B	{B[24:0], 0x00}	n/a	C[22:19] = 0011	C[18:16] = 011
R	R[31:0]	n/a	C[22:19] = 0100	C[18:16] = 100
S	S[31:0]	n/a	C[22:19] = 0101	C[18:16] = 101
T	T[31:0]	n/a	C[22:19] = 0110	C[18:16] = 110
IMM	D[31:0]	current instruction address	C[22:19] = 0001	C[18:16] = 001
ZERO	0x00000000	n/a	C[22:19] = 0000	C[18:16] = 000
ONES	0xFFFFFFFF	n/a	C[22:19] = 0111	C[18:16] = 111
REM	D[31:0]	specified by remote[8:0]	C[22:19] = 1000	n/a
REMP	{0x000000, P[8:0]}	specified by remote[8:0]	C[22:19] = 1001	n/a

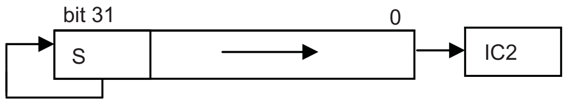
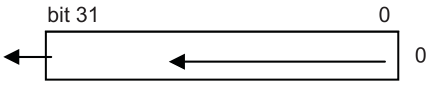
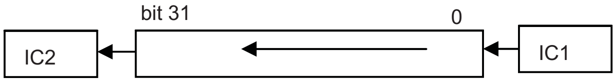
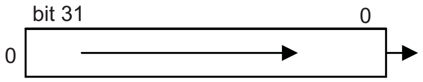
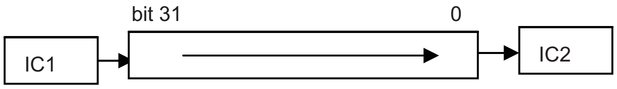
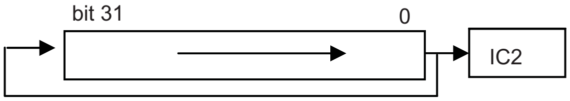
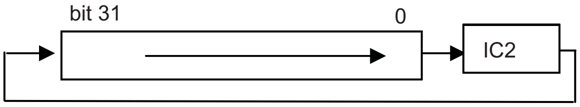
Table 23-78. Destination Operand Choices

Destination Operand	Stored Value	Address	dest	rdest
A	A[24:0] = result [31:8]	n/a	C[7] = 0, C[2:1] = 00	n/a
B	B[24:0] = result [31:8]	n/a	C[7] = 0, C[2:1] = 01	n/a
R	R[24:0] = result [31:0]	n/a	C[7] = 1, C[2:1] = 00	n/a
S	S[24:0] = result [31:0]	n/a	C[7] = 1, C[2:1] = 01	n/a
T	T[24:0] = result [31:0]	n/a	C[7] = 0, C[2:1] = 10	n/a

Table 23-78. Destination Operand Choices (continued)

Destination Operand	Stored Value	Address	dest	rdest
IMM	D[31:0] = result [31:0]	current instruction address	C[7] = 1, C[2:1] = 10	n/a
NONE	n/a	n/a	C[7] = 0, C[2:1] = 11	C[4:3] = 00
REM	D[31:0] = result [31:0]	specified by remote[8:0]	n/a	C[4:3] = 01
REMP	P[8:0] = result [8:0]	specified by remote[8:0]	n/a	C[4:3] = 10

Table 23-79. Shift Encoding

Shift Type	C[15:13] smode	Operation Illustrated [1]
No Shift Applied	0 0 0	n/a - no shift
ASR-Arithmetic Shift Right	0 0 1	
LSL-Logical Shift Left	0 1 0	
CSL-Carry Shift Left	0 1 1	
LSR-Logical Shift Right	1 0 0	
CSR-Carry Shift Right	1 0 1	
RR - Rotate Right	1 1 0	
CRR – Carry Rotate Right	1 1 1	

[1] IC1 is the carry flag after the arithmetic / logical operation is performed. Ic2 is the updated carry flag after the shift operation is performed. s is the sign bit.

Table 23-80. Execution Time for ADC, ADD, AND, OR, SBB, SUB, XOR Instructions

src1	dest	rdest	remote[8:0]	Cycles
ZERO, IMM, A, B, R, S, T, or ONES	A,B,R,S,T, or NONE	NONE	! = next[8:0]	1
REM or REMP	A,B,R,S,T, or NONE	NONE	!= next[8:0]	2
ZERO, IMM, A, B, R, S, T, or ONES	IMM	REM	!= next[8:0]	2
ZERO, IMM, A, B, R, S, T, or ONES	A,B,R,S,T, or NONE	REMP	!= next[8:0]	2
ZERO, IMM, A, B, R, S, T, or ONES	A,B,R,S,T, or NONE	NONE	== next[8:0]	2
REM or REMP	IMM	REM	x	3
x	IMM	REMP	x	3
REM or REMP	x	REM	== next[8:0]	3
x	x	REMP	== next[8:0]	3

Execution

```
         / Notes: IR1, IR2 are 32-bit intermediate results
   // SRC1, SRC2 are 32-bit sources selected
   //            by fields src1, src2
   // IC1, IC2 are intermediate values of the carry flag
   // IZ1, IZ2 are intermediate values of the zero flag
   // IN1, IN2 are intermediate values of the negative flag
   // IV1, IV2 are intermediate values of the overflow flag
   // scount is the shift count (0 to 31) specified by C12:C8

   /********** SOURCE OPERAND DECODING STAGE **********/
   switch (C22:C19)
{
     case 0000:SRC1[31:0] = 0x00000000
     case 0001:SRC1[31:0] = Immediate Data Field D[31:0]
     case 0010:SRC1[31:8] = A[24:0]; SRC1[6:0] = 0
     case 0011:SRC1[31:8] = B[24:0]; SRC1[6:0] = 0
     case 0100:SRC1[31:0] = R[31:0]
     case 0101:SRC1[31:0] = S[31:0]
     case 0110:SRC1[31:0] = T[31:0]
     case 0111:SRC1[31:0] = 0xFFFFFFFF
     case 1000:SRC1[31:0] = Remote Data Field D[31:0]
     case 1001:SRC1[31:9] = 0; SRC1[8:0] = Remote Program Field P[8:0]
   }

switch (C18:C16)
{
     case 000:SRC2[31:0] = 0x00000000
     case 001:SRC2[31:0] = Immediate Data Field[31:0]
     case 010:SRC2[31:8] = A[24:0]; SRC2[6:0] = 0
     case 011:SRC2[31:8] = B[24:0]; SRC2[6:0] = 0
     case 100:SRC2[31:0] = R[31:0]
     case 101:SRC2[31:0] = S[31:0]
     case 110:SRC2[31:0] = T[31:0]
     case 111:SRC2[31:0] = 0xFFFFFFFF
   }
   /******** ARITHMETIC / LOGICAL OPERATION STAGE *******/
   switch (C[25:23])
   {
     case 011:IR1 = src1 + src2 + C // ADC
     case 001:IR1 = src1 + src2 // ADD
     case 010:IR1 = src1 & src2 // AND
     case 100:IR1 = src1 | src2 // OR
     case 110:IR1 = src1 - src2 - C // SBB
     case 101:IR1 = src1 - src2 // SUB
     case 111:IR1 = src1 ^ src2 // XOR
   }

   IC1 = Carry Out if Operation is ADD, ADC, SUB, SBB
           = 0 if Operation is AND, OR, XOR
   IZ1 = Set if IR1 is zero, Clear if IR1 is non-zero
   IN1 = IR[31]
   IV1 = (IC1 XOR IR1[31]) AND NOT(SRC1[31] XOR SRC2[31])

   /******************** SHIFT STAGE ********************/
   switch (C15:C13)
{
     case 000: // smode = No Shift
           IR2 = IR1
           IC2 = IC1; IZ2 = IZ1; IN2 = IN1; IV2 = IV1;

     case 001: // smode = Arithmetic Shift Right
           IR2[31 - scount : 0] = IR1[31:scount]

           if (scount>0) {
              IR2[31 : 31 - scount + 1] = IR1[31]
              IC2 = IR1[scount-1]
           }
           else {
              IC2 = IC1
           }

           IN2 = IR2[31];
           if (IR2 == 0) { IZ2 = 1 } else {IZ2 = 0};
           IV2 = (IR2[31] XOR IR1[31]) OR IV1

     case 010: // smode = Logical Shift Left
           IR2[31 : scount] = IR1[31 - scount: 0]

           if (scount > 0) {
              IR2[scount - 1 : 0] = 0
           }

           IC2 = IC1
           IN2 = IR2[31];
           if (IR2 == 0) { IZ2 = 1 } else {IZ2 = 0};
           IV2 = (IR2[31] XOR IR1[31]) OR IV1

     case 011: // smode = Carry Shift Left
           IR2[31 : scount] = IR1[31 - scount: 0]

        if (scount>0) {
              IR2[scount - 1 : 0] = [IC1,...IC1]
              IC2 = IR1[31 - scount + 1]
           }
           else
        {
              IC2 = IC1
           }

        IN2 = IR2[31];
           if (IR2 == 0) { IZ2 = 1 } else {IZ2 = 0};
           IV2 = (IR2[31] XOR IR1[31]) OR IV1

     case 100: // smode = Logical Shift Right
           IR2[31 - scount : 0] = IR1[31:scount]

           if (scount>0) {
              IR2[31 : 31 - scount + 1] = 0
           }

           IC2 = IC1
           IN2 = IR2[31];
           if (IR2 == 0) { IZ2 = 1 } else {IZ2 = 0};
           IV2 = (IR2[31] XOR IR1[31]) OR IV1

     case 101: // smode = Carry Shift Right
           IR2[31 - scount : 0] = IR1[31:scount]

           if(scount>0) {
              IR2[31:31-scount + 1] = [IC1,...IC1]
              IC2 = IR1[scount-1]
           }
           else {
              IC2 = IC1
           }

           IN2 = IR2[31];
           IZ2 = Set if IR2 == 0;
           IV2 = (IR2[31] XOR IR1[31]) OR IV1

     case 110: // smode = Rotate Right
```

```
            IR2[31 - scount : 0] = IR1[31:scount]

            if(scount>0) {
               IR2[31:31-scount+1] = IR1[scount-1:0]
               IC2 = IR1[scount-1]
            }
            else {
               IC2 = IC1
            }

            IN2 = IR2[31];
            if (IR2 == 0) { IZ2 = 1 } else {IZ2 = 0};
            IV2 = (IR2[31] XOR IR1[31]) OR IV1

      case 111: // smode = Carry Rotate Right
            IR2[31 - scount : 0] = IR1[31:scount]

            if (scount == 0) {
               IC2 = IC1
            }
            else if (scount == 1) {
               IR2[31] = IC1
               IC2 = IR1[0]
            }
            else {
               IR2[31:31-scount+1] = {IR1[scount-2:0],IC1}
               IC2 = IR1[scount - 1]
            }

            IN2 = IR2[31];
            if (IR2 == 0) { IZ2 = 1 } else {IZ2 = 0};
            IV2 = (IR2[31] XOR IR1[31]) OR IV1
      }
      /********** WRITE REGISTER DESTINATION STAGE ***********/
      switch (C7, C2:C1)
      {
            case 000:A[24:0] = IR2[31:8]
            case 001:B[24:0] = IR2[31:8]
            case 010:T[31:0] = IR2[31:0]
            case 011:IR2 is not stored in register, immediate
            case 100:R[31:0] = IR2[31:0]
            case 101:S[31:0] = IR2[31:0]
            case 110:Immediate Data Field[31:0] = IR2
            case 111:IR2 is not stored in register, immediate
      }
      /*********** WRITE REMOTE DESTINATION STAGE ***********/
      switch (C4:3)
   {
            case 00:IR2 is not stored in remote field
            case 01:Remote Data Field D[31:0] = IR2
            case 10:Remote Program Field P[8:0] = IR2[8:0]
            case 11:IR2 is not stored in remote field
      }
      /***************** UPDATE FLAGS STAGE *****************/
      C FLAG = IC2
      N FLAG = IN2
      Z FLAG = IZ2
      V FLAG = IV2
      If (Init Flag == 1)
      {
            ACF = 0;
            DCF = 1;
            GPF = 0;
            NAF = 0;
      }
      else ACF, DCF, GPF, NAF remain unchanged;
```

23.6.3.5 ADM32 (Add Move 32)

Syntax

ADM32 {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
remote={label | 9-bit unsigned integer}
[control={OFF | ON}]
[init={OFF | ON}]
type={IM&REGTOREG | REM&REGTOREG | IM&REMTOREG | IM&REGTOREM}
reg={A | B | R | S | T }
data={25-bit unsigned integer}
[hr_data={7-bit unsigned integer}]
}

Figure 23-123. ADM32 Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8 0
0	Reserved	BRK	Next program address	0100	Remote address
6	3	1	9	4	9

Figure 23-124. ADM32 Control Field (C31:C0)

31 27	26	25 23	22 16
Reserved	Control	000	Reserved
5	1	3	15

15 8	7	6	5	4 3	2 1	0
Reserved	Ext Reg	Init flag	1	Move type	Register select	Res.
15	1	1	1	2	2	1

Figure 23-125. ADM32 Data Field (D31:D0)

31 7	6 0
Data	HR Data
25	7

Cycles One or two cycles (see Table 23-81)

Register modified Selected register (A, B, R, S, or T)

Description This instruction modifies the selected ALU register or data field values at the remote address depending on the move type. The modified value results from adding the immediate or remote data field to the ALU register or the remote data field, depending on the move type. Table description shows the C2 and C1 bit encoding for determining which register is selected.

init (Optional) Determines whether or not system flags are initialized. A value of ON reinitializes the following system flags to these states:

Acceleration flag (ACF) = 0

Deceleration flag (DCF) = 1

Gap flag (GPF) = 0

New angle flag (NAF) = 0

A value of OFF results in no change to the system flags.

Default: OFF

type Specifies the move type to be executed.

Table 23-81. Move Types for ADM32

Type	C4	C3	Add	Destination(s)	Cycles
IM&REGTOREG	0	0	Imm. data field + Reg. A, B, R, S, or T	Register A, B, R, S, or T	1
REM&REGTOREG	0	1	Remote data field + Reg. A, B, R, S, or T	Register A, B, R, S, or T	2
IM&REMTOREG	1	0	Imm. data field + Remote data field	Register A, B, R, S, or T	2
IM&REGTOREM	1	1	Imm. data field + Reg. A, B, R, S, or T	Remote data field	1

If selected register is R, S, or T, the operation is a 32-bit Addition/move. If A or B register is selected, it is limited to 25-bit operation since A and B only support 25-bit.

data Specifies the 25-bit integer value for the immediate data field.

hr_data Specifies the 7 least significant bits of the immediate data field.

Default: 0.

Execution

```
switch (C4:C3)
{
      case 00:
           Selected register = Selected register + Immediate Data Field;
      case 01:
           Selected register = Selected register + Remote Data Field;
      case 10:
           Selected register = Immediate Data Field + Remote Data Field;
      case 11:
           Remote Data Field = Selected register + Immediate Data Field;
}

If (Init Flag == 1)
{
      ACF = 0;
      DCF = 1;
      GPF = 0;
      NAF = 0;
}
else
      All flags remain unchanged;

Jump to Next Program Address;
```

Figure 23-126 and Figure 23-127 illustrate the ADM32 operation for various cases.

Figure 23-126. ADM32 Add and Move Operation for IM&REGTOREG (Case 00)

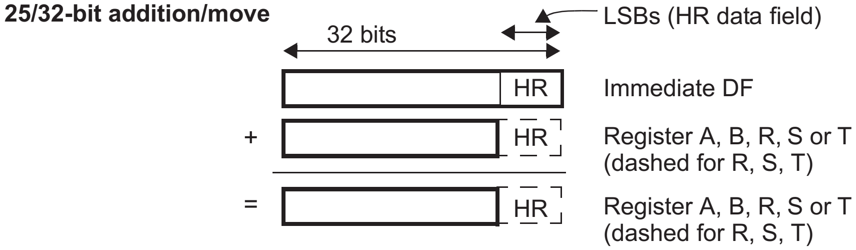
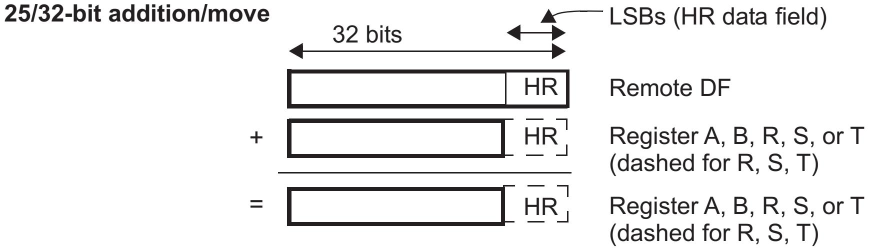


Figure 23-127. ADM32 Add and Move Operation for REM&REGTOREG (Case 01)



23.6.3.6 APCNT (Angle Period Count)

Syntax

```
APCNT {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}]
[request={NOREQ | GENREQ | QUIET}]
[irq={OFF | ON}]
type={FALL2FALL | RISE2RISE}
[control={OFF | ON}]
prv={OFF | ON}}]
period={25-bit unsigned integer}
data={25-bit unsigned integer}
}
```

Figure 23-128. APCNT Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8	7 6	5 0
0	Request Number	BRK	Next program address	1110	Int. ena	Edge select	Reserved
6	3	1	9	4	1	2	6

Figure 23-129. APCNT Control Field (C31:C0)

31 29	28 27	26	25	24 0
Res.	Request type	Control	Prv.	Period Count
3	2	1	1	25

Figure 23-130. APCNT Data Field (D31:D0)

31 7	6 0
Data	Reserved
25	7

Cycles

One or two cycles

- Cycle 1: edge detected (normal operation)
- Cycle 2: edge detected and GPF = 1 and underflow condition is true

One cycle (normal operation) two cycles (edge detected)

Register modified

Register A and T (implicitly)

Description

This instruction is used before SCNT and ACNT to generate an angle-referenced time base synchronized to an external signal (that is, a toothed wheel signal). It is assumed that the pin and edge selections are the same for APCNT and ACNT.

APCNT is restricted to pin HET[2]. The toothed wheel must then be connected to pin HET[2].

APCNT uses the gap flag (GPF) defined by ACNT to start or stop captures in the period count field [C24:C0]. When GPF = 1, the previous period value is held in the control field and in register T. When GPF = 0, the current period value is captured in the control field and in register T.

APCNT uses the step width flags (SWF0 and SWF1) defined by SCNT to detect period durations shorter than one step, and then disables capture.

The edge select encoding is shown in Table 23-82.

irq ON generates an interrupt when the edge state is satisfied. OFF prevents an interrupt from being generated.

Default: OFF.

type Specifies the edge type that triggers the instruction.

Default: Fall2Fall.

Table 23-82. Edge Select Encoding for APCNT

type	P7	P6	Selected Condition
Fall2Fall	1	0	Falling edge
Rise2Rise	1	1	Rising edge

period Contains the 25-bit count value from the previous APCNT period.

data 25-bit value serving as a counter.

Default: 0.

Execution

```
Z = 0;

If (Data field register != 1FFFFFFh)
{
   Register A = Data field register + 1;
   Data field register = Data field register + 1;
}
elseIf (specified edge not detected on HET[2])
{
   Register A = 1FFFFFFh;
   APCNT Ovflw flag = 1;
}

If (specified edge detected on HET[2])
{
   Z = 1;

   If (Data field register == 1FFFFFFh)
   {
      Register A = 1FFFFFFh;
      Register T = 1FFFFFFh;Period count = 1FFFFFFh;
      Period count = 1FFFFFFh;
   }
   elseIf (GPF == 0 AND Data Field register >= Step width)
   {
      Register A = Data field register + 1;
      Register T = Register A;
      Period count = Register T;

      If (Interrupt Enable == 1)
         HETFLG[n] = 1;      /* n depends on address */
      If ([C28:C27] == 01)
         Generate request on request line [P25:P23];
      If ([C28:C27] == 11)
         Generate quiet request on request line [P25:P23];
   }

   If (GPF == 1)
         Register T = Period count;
   If (Data Field register < Step width)
   {
      Register T = Period count;
      APCNT Undflw flag = 1;
      Period Count = 000000h;
   }

   Data field register = 000000h;
}
else
{
      Register T = Period count;
}

Prv bit = Current Lx value of HET[2] pin;

Jump to Next Program Address;
```

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 23.2.7.

23.6.3.7 BR (Branch)

Syntax

BR {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}]
[request={NOREQ | GENREQ | QUIET}]
[control={OFF | ON}]
[prv={OFF | ON}]
cond_addr={label | 9-bit unsigned integer}
[pin= {pin number}]
event={NOCOND | FALL | RISE | BOTH | ZERO | NAF | LOW | HIGH | C | NC | EQ | Z | NE | NZ | N | PZ | V | NV | ZN | P | GE | LT | GT | LE | LO | HS }
[irq={OFF | ON}]
}

Figure 23-131. BR Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8 0
0	Request Number	BRK	Next program address	1101	Reserved
6	3	1	9	4	9

Figure 23-132. BR Control Field (C31:C0)

31 29	28 27	26	25	24 22	21 16
Reserved	Request type	Control	Prv	Reserved	Conditional address
3	2	1	1	3	9

15 13	12 8	7 3	2 1	0
Conditional address	Pin select	Branch cond.	Reserved	Int. ena
9	5	5	2	1

Figure 23-133. BR Data Field (D31:D0)

31 0
Reserved
32

Cycles One

Register modified None

Description This instruction executes a jump to the conditional address [C21:C13] on a pin or a flag condition, and can be used with all pins.

Table 23-83 provides the branch condition encoding.

event Specifies the event that triggers a jump to the indexed program address.

Default: FALL

irq ON generates an interrupt when the event occurs that triggers the jump. If irq is set to OFF, no interrupt is generated.

Default: OFF.

Table 23-83. Branch Condition Encoding for BR

Event	C7	C6	C5	C4	C3	Branch Condition
NOCOND	0	0	0	0	0	Always
FALL	0	0	1	0	0	On falling edge on the selected pin
RISE	0	1	0	0	0	On rising edge on selected pin
BOTH	0	1	1	0	0	On rising or falling edge on selected pin
ZERO	1	0	0	0	0	If Zero flag is set
NAF	1	0	1	0	0	If NAF_global flag is set
LOW	1	1	0	0	0	On LOW level on selected pin
HIGH	1	1	1	0	0	On HIGH level on selected pin
C	0	0	0	0	1	Carry Set: C==1
NC	0	0	0	1	1	Carry Not Set: C==0
EQ, Z	0	0	1	0	1	Equal or Zero: Z==1
NE, NZ	0	0	1	1	1	Not Equal or Not Zero: Z==0
N	0	1	0	0	1	Negative: N==1
PZ	0	0	1	1	1	Positive or Zero: N==0
V	0	1	1	0	1	Overflow: V==1
NV	0	1	1	1	1	No Overflow: V==0
ZN	1	0	0	0	1	Zero or Negative: (Z OR N) == 1
P	1	0	0	1	1	Positive: (Z OR N) == 0
GE	1	0	1	1	1	Signed Greater Than or Equal: (N XOR V) == 0
L	1	0	1	0	1	Signed Less Than (N XOR V) == 1
G	1	1	0	1	1	Signed Greater Than (Z OR (N XOR V)) == 0
LE	1	1	0	0	1	Signed Less Than (Z OR (N XOR V)) == 1
LO	1	1	1	1	1	Unsigned Less Than: (C OR Z) == 0
HS	1	1	1	0	1	Unsigned Higher or Same (C OR Z) == 1

Execution

```
If (Condition is true)
{
   If (Interrupt Enable == 1) HETFLG[n] = 1;      /* n depends on address */
   If ([C28:C27] == 01) Generate request on request line [P25:P23];
   If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];
   Jump to Conditional Address;
}
else
{
   Jump to Next Program Address;
}

Prv bit = Current Lx value of selected pin; (Always Executed)
```

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 23.2.7.

23.6.3.8 CNT (Count)

Syntax

CNT {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}]
[request={NOREQ | GENREQ | QUIET}]
[angle_count={OFF | ON}]
[reg={A | B | T | NONE}]
[comp ={EQ | GE}]
[irq={OFF | ON}]
[control={OFF | ON}]
max={25-bit unsigned integer}
[data={25-bit unsigned integer]
}

Figure 23-134. CNT Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8	7 6	5	4 1	0
0	Request Number	BRK	Next program address	0110	Angle count	Register	Comp. select	Res.	Int. ena
6	3	1	9	4	1	2	1	4	1

Figure 23-135. CNT Control Field (C31:C0)

31 29	28 27	26	25	24 0
Res.	Request type	Control	Res.	Max Count
3	2	1	1	25

Figure 23-136. CNT Data Field (D31:D0)

31 7	6 0
Data	Reserved
25	7

Cycles

One or two

One cycle (time mode), two cycles (angle mode)

Register modified

Selected register (A, B or T)

Description

This instruction defines a virtual timer. The counter value stored in the data field [D31:7] is incremented unconditionally on each execution of the instruction when in time mode (angle count bit [P8] = 0). When the count reaches the maximum count specified in the control field, the counter is reset. It takes one cycle in this mode.

In angle mode (angle count bit [P8] = 1), CNT needs data from the software angle generator (SWAG). When in angle count mode the angle increment value will be 0 or 1. It takes two cycles in this mode.

angle_count Specifies when the counter is incremented. A value of ON causes the counter value to be incremented only if the new angle flag is set (NAF_global = 1). A value of OFF increments the counter each time the CNT instruction is executed.

Default value for this field is OFF.

comp When set to EQ the counter is reset, when it is equal to the maximum count.

When set to GE the counter is reset, when it is greater or equal to the maximum count.

Default: GE.

irq ON generates an interrupt when the counter overflows to zero. The interrupt is not generated until the data field is reset to zero. If irq is set to OFF, no interrupt is generated.

Default: OFF.

max Specifies the 25-bit integer value that defines the maximum count value allowed in the data field. When the count in the data field is equal to max, the data field is reset to 0 and the Z system flag is set to 1.

data Specifies the 25-bit integer value serving as a counter.

Default: 0.

Execution

```
Z = 0;

If (Angle Count (bit P8 == 1))
{
   If (NAF_global == 0)
   {
      Selected register = immediate data field;
      Jump to Next Program Address;
   }
   else
   {
      If ((Immediate Data Field + Angle Increment) >= Max count)
      {
         Z = 1;
         Selected register = ((Immediate Data Field + Angle Inc.) - Max count);
         Immediate Data Field = ((Immediate Data Field + Angle Inc.) - Max count);

         If (Interrupt Enable == 1) HETFLG[n] = 1;      /* n depends on address */
         If ([C28:C27] == 01) Generate request on request line [P25:P23];
         If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];
      }
      else
      {
         Selected register = Immediate Data Field + Angle Increment;
         Immediate Data Field = Immediate Data Field + Angle Increment;
      }
   }
}

else if(Time mode (bit P8 == 0))
{
   If [(P5==0) AND (Immediate Data Field == Max count)]
   OR [(P5==1) AND (Immediate Data Field >= Max count)]
   {
       Z = 1;
       Selected register = 00000;
       Immediate Data Field = 00000;

       If (Interrupt Enable == 1) HETFLG[n] = 1;      /* n depends on address */
       If ([C28:C27] == 01) Generate request on request line [P25:P23];
       If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];
   }
   else
   {
       Selected register = Immediate Data Field + 1;
       Immediate Data Field = Immediate Data Field + 1;
   }
}

Jump to Next Program Address;
```

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 23.2.7.

23.6.3.9 DADM64 (Data Add Move 64)

Syntax

```
DADM64 {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
remote={label | 9-bit unsigned integer}
[request={NOREQ | GENREQ | QUIET}]
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
[cond_addr={label | 9-bit unsigned integer}]
[pin={pin number}]
comp_mode={ECMP | SCMP | MCMP1 | MCMP2}
[action={CLEAR | SET | PULSELO | PULSEHI}]
[reg={A | B | R | S | T | NONE}]
[irq={OFF | ON}]
[data={25-bit unsigned integer]
[hr_data= {7-bit unsigned integer}]
}
```

-or-

Syntax

```
DADM64 {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
remote={label | 9-bit unsigned integer}
cntl_val={29-bit unsigned integer}
data={25-bit unsigned integer}
[hr_data= {7-bit unsigned integer}]
}
```

Figure 23-137. DADM64 Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8 0
0	Reserved	BRK	Next program address	0010	Remote Address
6	3	1	9	4	9

Figure 23-138. DADM64 Control Field (C31:C0)

31 29	28 27	26	25 23	22	21 16
Reserved	Request type	Control	Reserved	En. pin action	Conditional address
3	2	1	3	1	9

15 13	12 8	7	6 5	4 3	2 1	0
Conditional address	Pin select	Ext Reg	Comp. mode	Action	Register select	Int. ena
9	5	1	2	2	2	1

Figure 23-139. DADM64 Data Field (D31:D0)

31 7	6 0
Data	HR Data
25	7

Cycles Two

Register modified Register T (implicitly)

Description This instruction modifies the data field and the control field at the remote address. The remote data field value is not just replaced, but is added with the DADM64 data field.

DADM64 has two distinct syntaxes. In the first syntax, bit values may be set by assigning a value to each of the control fields. This syntax is convenient for modifying control fields that are arranged similarly to the format of the DADM64 control field. A second syntax, in which the entire 29-bit control field is specified by the cntl_val field, is convenient when the remote control field is dissimilar to the DADM64 control field. Either syntax may be used, but you must use one or the other but not a combination of syntaxes.

Figure 23-140 shows the DADM64 add and move operation.

Figure 23-140. DADM64 Add and Move Operation

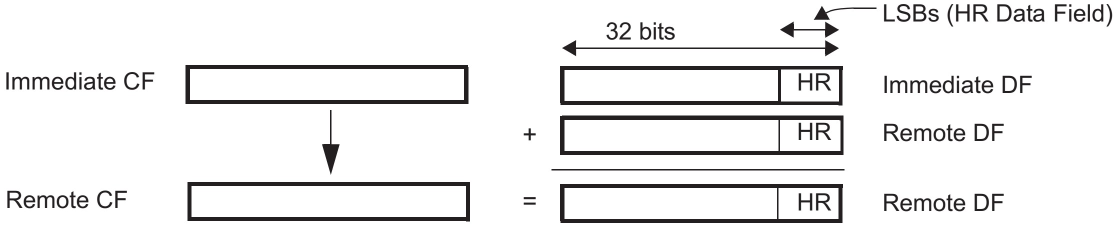


Table 23-84. DADM64 Control Field Description

request	maintains the control field for the remote instruction
control	maintains the control field for the remote instruction
en_pin_action	maintains the control field for the remote instruction
cond_addr	maintains the control field for the remote instruction

Table 23-84. DADM64 Control Field Description (continued)

pin	maintains the control field for the remote instruction
register	maintains the control field for the remote instruction
action	maintains the control field for the remote instruction
irq	maintains the control field for the remote instruction
data	Specifies the 25-bit initial value for the data field.
hr_data	Seven least significant bits of the 32 bit data field. Default: 0
cntl_val	Specifies the 29 least significant bits of the Control field.

Execution

```
Remote Data Field = Remote Data Field + Immediate Data Field;
Register T = Immediate Data Field;
Remote Control Field = Immediate Control Field;
Jump to Next Program Address;
```

23.6.3.10 DJZ (Decrement and Jump if Zero)

DJNZ is also a supported syntax. The functionality of the two instruction names is identical.

Syntax

```
DJZ {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}
[request={NOREQ | GENREQ | QUIET}]
[control={OFF | ON}]
[cond_addr={label | 9-bit unsigned integer}]
[reg={A | B | T | NONE}]
[irq={OFF | ON}]
[data={25-bit unsigned integer]
}
```

Figure 23-141. DJZ Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8	7 6	5 0
0	Request Number	BRK	Next program address	1010	Res.	10	Reserved
6	3	1	9	4	1	2	6

Figure 23-142. DJZ Control Field (C31:C0)

31 29	28 27	26	25 22	21 16
Reserved	Request type	Control	Reserved	Conditional address
3	2	1	4	9

15 13	12 8 7 3	2 1	0
Conditional address	Reserved	Register select	Int. ena
9	10	2	1

Figure 23-143. DJZ Data Field (D31:D0)

31 7	6 0
Data	Reserved
25	7

Cycles One

Register modified Selected register (A, B, or T)

Description This instruction defines a virtual down counter used for delayed execution of certain instructions (to generate minimum on/off times). When DJZ is executed with counter value not zero, the counter value is decremented. If the counter value is zero, the counter remains zero until it is reloaded with a non-zero value. The program flow can be modified when down counter value is zero by using the conditional address.

cond_addr This field is not optional for the DJZ instruction.

irq ON generates an interrupt when the data field reaches zero. No interrupt is generated when the bit is OFF.

Default: OFF.

data Specifies the 25-bit integer value used as a counter. This counter is decremented each time the DJZ instruction is executed until the counter reaches 0.

Default: 0.

Execution

```
If (Data != 0)
{
   Data = Selected register = Data - 1;
   Jump to Next Program Address;
}
else
{
   Selected register = 000000h;

   If (Interrupt Enable == 1) HETFLG[n] = 1;      /* n depends on address */
   If ([C28:C27] == 01) Generate request on request line [P25:P23];
   If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];

   Jump to conditional Address;
}
```

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 23.2.7.

23.6.3.11 ECMP (Equality Compare)

Syntax

```
ECMP {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}
[request={NOREQ | GENREQ | QUIET}]
[hr_lr={HIGH | LOW}]
[angle_comp={OFF | ON}]
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
[cond_addr={label | 9-bit unsigned integer}]
pin={pin number}
[action={CLEAR | SET | PULSELO | PULSEHI}]
[reg={A | B | R | S| T | NONE}]
[irq={OFF | ON}]
[data={25-bit unsigned integer]
[hr_data={7-bit unsigned integer}]
}
```

Figure 23-144. ECMP Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8	7	6 0
0	Request Number	BRK	Next program address	0000	hr_lr	Angle comp.	Reserved
6	3	1	9	4	1	1	7

Figure 23-145. ECMP Control Field (C31:C0)

31 29	28 27	26	25 23	22	21 16
Reserved	Request type	Control	Reserved	En. pin action	Conditional address
3	2	1	3	1	9

15 13	12 8	7	6 5	4 3	2 1	0
Conditional address	Pin select	Ext Reg	00	Action	Register select	Int. ena
9	5	1	2	2	2	1

Figure 23-146. ECMP Data Field (D31:D0)

31 7	6 0
Data	HR Data
25	7

Cycles One

Register modified Register A, B, R, S or T if selected

Description ECMP can use all pins. This instruction compares a 25-bit data value stored in the data field (D31–D7) to the value stored in the selected ALU register (A, B, R, S, or T). Register select encoding can be found in Section 23.6.2.

If R, S, or T registers are selected, and if the 25-bit data field matches, ECMP updates the register with the 32-bit value (D31-D0).

If the hr_lr bit is cleared, the pin action will occur after a high resolution delay from the next loop resolution clock. If the hr_lr bit is set, the delay is ignored. This delay is programmed in the data field (D6–D0).

The behavior of the pins is governed by the four action options in bits C4:C3. ECMP uses the zero flag to generate opposite pin action (synchronized to the loop resolution clock).

angle_comp Determines if an angle compare is performed. A value of ON causes the comparison to be performed only if the new angle flag is set (NAF = 1). If OFF is specified, the compare is then performed regardless of the state of the new angle flag.

Default: OFF.

irq Specifies whether or not an interrupt is generated. A value of ON sends an interrupt if register and data field values are equivalent. If OFF is selected, no interrupt is generated.

Default: OFF.

data Specifies the value for the data field. This value is compared with the selected register.

hr_data Specifies the HR delay.

Default: 0.

Execution

```
If (Angle Comp. bit == 0 OR (Angle Comp. bit == 1 AND NAF_global == 1))
{

   If (Selected register value == Immediate data field value)
   {

       If (hr_lr bit == 0)
       {
          If (Enable Pin action == 1)
          {
             Selected Pin = Pin Action AT next loop resolution clock + HR delay;
          }
       }
       else
       {
          If (Enable Pin action == 1)
          {
             Selected Pin = Pin Action AT next loop resolution clock;
          }
       }

       If (Z == 1 AND Opposite action == 1)
       {
          If (Enable Pin action == 1)
          {
             Selected Pin = opposite Pin Action AT next loop resolution clock;
          }

          If (Interrupt Enable == 1) HETFLG[n] = 1;      /* n depends on address */
          If ([C28:C27] == 01) Generate request on request line [P25:P23];
          If ([C28:C27] == 11)Generate quiet request on request line [P25:P23];

          If (register R is selected) R register = Compare value (32 bit);
          If (register S is selected) S register = Compare value (32 bit);
          If (register T is selected) T register = Compare value (32 bit);

          Jump to Conditional Address;
       }
   }
   elseIf (Z == 1 AND Opposite action == 1)
   {
      If (Enable Pin action == 1)
      {
         Selected Pin = opposite Pin Action AT next loop resolution clock;
      }
      Jump to Next Program Address;
}
else  // Angle Comp. bit == 1 AND NAF_global == 0
{
   Jump to Next Program Address;
}
```

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 23.2.7.

23.6.3.12 ECNT (Event Count)

Syntax

```
ECNT {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}
[request={NOREQ | GENREQ | QUIET}]
[control={OFF | ON}]
[prv={OFF | ON}]
[cond_addr={label | 9-bit unsigned integer}]
pin={pin number}
event={NAF | FALL | RISE | BOTH | ACCUHIGH | ACCULOW}
[reg={A | B | R| S | T | NONE}]
[irq={OFF | ON}]
[data={25-bit unsigned integer]
}
```

Figure 23-147. ECNT Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8	7 6	5 0
0	Request Number	BRK	Next program address	1010	Res.	01	Reserved
6	3	1	9	4	1	2	6

Figure 23-148. ECNT Control Field (C31:C0)

31 29	28 27	26	25	24 22	21 16
Reserved	Request type	Control	Prv.	Reserved	Conditional address
3	2	1	1	3	9

15 13	12 8	7	6 4	3	2 1	0
Conditional address	Pin select	Ext Reg	Event	Res.	Register select	Int. ena
9	5	1	3	1	2	1

Figure 23-149. ECNT Data Field (D31:D0)

31 7	6 0
Data	Reserved
25	7

Cycles One cycle

Register modified Selected Register (A, B, R, S, T or none)

Description This instruction defines a specialized 25-bit virtual counter used as an event counter or pulse accumulator (see Table 23-85). The counter value is stored in the data field [D31:D7] and the selected register. If one of the 32-bit registers (R,S,T) is selected, the 25 bit count value is stored left justified in the register with zeros in the seven least significant bits.

When an event count condition is specified, the counter value is incremented on a pin edge condition or on the NAF condition (NAF is defined in ACNT). This instruction can be used with all pins.

event The event that triggers the counter.

Table 23-85. Event Encoding Format for ECNT

Event	C6	C5	C4	Count Conditions	Mode	Int. Available
NAF	0	0	0	NAF flag is Set	Angle counter	Y
FALL	0	0	1	Falling edge on selected pin	Event counter	Y
RISE	0	1	0	Rising edge on selected pin	Event counter	Y
BOTH	0	1	1	Rising and Falling edge on selected pin	Event counter	Y
ACCUHIGH	1	0	-	while pin is high level	Pulse accumulation	N
ACCULOW	1	1	-	while pin is low level	Pulse accumulation	N

irq ON generates an interrupt when event in counter mode occurs. No interrupt is generated with OFF.

Default: OFF.

data 25-bit integer value serving as a counter.

Default: 0.

Execution

```
If (event occurs)
{
   If (Register A or B Selected) {
      Selected register = Immediate Data Field + 1;
   }

   If (Register R, S or T Selected)
   {
      Selected register[31:7] = Immediate Data Field + 1;
      Selected register[6:0] = 0;
   }

   Immediate Data Field = Immediate Data Field + 1;

   If (Interrupt Enable == 1) HETFLG[n] = 1;      /* n depends on address */
   If ([C28:C27] == 01) Generate request on line [P25:P23];
   If ([C28:C27] == 11) Generate quiet request on line [P25:P23];

   Jump to Conditional Address;
}
else
{
    Jump to Next Program Address;
}

Prv bit = Current Logic (Lx) value of selected pin; (Always executed)
```

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 23.2.7.

23.6.3.13 MCMP (Magnitude Compare)

Syntax

```
MCMP {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}
[request={NOREQ | GENREQ | QUIET}]
[hr_lr={LOW |HIGH}]
[angle_comp={OFF | ON}]
[savesub={OFF | ON}]
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
[cond_addr={label | 9-bit unsigned integer}]
pin={pin number}
order={REG_GE_DATA | DATA_GE_REG}
[action={CLEAR | SET | PULSELO | PULSEHI}]
reg={A | B | R | S | T | NONE}
[irq={OFF | ON}]
[data={25-bit unsigned integer]
[hr_data={7-bit unsigned integer}]
}
```

Figure 23-150. MCMP Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8	7	6	5	4 0
0	Request Number	BRK	Next program address	0000	hr_lr	Angle comp.	Res.	Save sub.	Res.
6	3	1	9	4	1	1	1	1	5

Figure 23-151. MCMP Control Field (C31:C0)

31 29	28 27	26	25 23	22	21 16
Reserved	Request type	Control	Reserved	En. pin action	Conditional address
3	2	1	3	1	9

15 13	12 8	7	6	5	4 3	2 1	0
Conditional address	Pin select	Ext Reg	1	Order	Action	Register select	Int. ena
9	5	1	1	1	2	2	1

Figure 23-152. MCMP Data Field (D31:D0)

31 7	6 0
Data	HR Data
25	7

Cycles One

Register modified T (if save sub bit P[5] is set)

Description

This instruction compares the magnitude of the 25-bit data value stored in the data field (D31-D7) and the 25-bit value stored in the selected ALU register (A, B, R, S, or T).

If the hr_lr bit is reset, pin action will occur after a delay from the next loop resolution clock. If the hr_lr bit is set, the delay is ignored. This delay is programmed in the data field (D6-D0).

When the data value matches, an output pin can be set or reset according to the pin action bit (C[4]). The pin will not change states if the enable pin action bit (C[22]) is reset.

MCMP uses the zero flag set to generate opposite pin action (synchronized to the loop resolution clock). The save sub bit (P[5]) provides the option to save the result of a subtraction into register T.

NOTE: The Difference Between Compare Values

The difference between the two data values must not exceed (2^{24}) - 1.

angle_comp

Determines whether or not an angle compare is performed. A value of ON causes the comparison to be performed only if the new angle flag is set (NAF = 1). If OFF is specified, the compare is then performed regardless of the state of the new angle flag.

Default: OFF.

savesub

When set, the comparison result is saved into the T register (upper 25 bits).

Default: OFF.

order

Specifies the order of the operands for the comparison.

Table 23-86. Magnitude Compare Order for MCMP

Order	C5	Description
REG_GE_DATA	0	Evaluates to true if the register value is greater than or equal to the data field value.
DATA_GE_REG	1	Evaluates to true if the data field value is greater than or equal to the register value.

irq

Specifies whether or not an interrupt is generated. A value of ON sends an interrupt if the compare match occurs according to the order selected. If OFF is selected, no interrupt is generated.

data

Specifies the value for the data field. This value is compared with the selected register.

hr_data

HR delay. The default value for an unspecified bit is 0.

Execution

```
If (Angle Compare P[7] == 0 OR (P[7] == 1 AND NAF_global == 1))
{
    If(  (Order C[5] == 1) AND (Data[31:7]- Selected register[31:7]) >= 0))
      OR ( (Order C[5] == 0) AND Selected register[31:7] - Data[31:7]) >= 0))
    {
       If (Order C[5] == 1 AND Save subtract P[5] == 1)
       {
          Register T[31:7] = Data[31:7] - Selected register[31:7];
          Register T[6:0] = 0;
       }

       If (Order C[5] == 0 AND Save subtract P[5] == 1)
       {
          Register T[31:7] = Selected register[31:7] - Data[31:7];
          Register T[6:0] = 0;
       }

       If (Enable Pin Action C[22] == 1)
       {
          If (hr_lr P[8] = 0) {
             Schedule Action on Selected Pin C[12:8] at start of next loop
             + HR Delay D[6:0];
          }
          else
          {
             Schedule Pin Action on Selected Pin C[12:8] at start of next loop;
          }
       }

       If (Interrupt Enable == 1) HETFLG[n] = 1;      /* n depends on address */
       If ([C28:C27] == 01) Generate request on request line [P25:P23];
       If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];

       Jump to Conditional Address;
    }
    else if (Z == 1 AND Opposite Action C[3] == 1 )
    {
       If (Enable Pin Action C[22] == 1)
       {
          Schedule Opposite Pin Action on Selected Pin C[12:8] at start of next loop;
       }

       Jump to Next Program Address;
    }
    else
       Jump to Next Program Address;
}
else // Angle Comp. bit == 1 AND NAF_global == 0
   Jump to Next Program Address;
```

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 23.2.7.

23.6.3.14 MOV32 (Data Move 32)

Syntax

```
MOV32 {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
remote={label | 9-bit unsigned integer}
[control={OFF | ON}]
[z_cond={OFF | ON}]
[init={OFF | ON}]| ON}]
type={IMTOREG | IMTOREG&REM | REGTOREM | REMTOREG}
[reg={A | B | R | S | T | NONE}]
[data={25-bit unsigned integer]
[hr_data={7-bit unsigned integer}]
}
```

Figure 23-153. MOV32 Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8 0
0	Reserved	BRK	Next program address	0100	Remote Address
6	3	1	9	4	9

Figure 23-154. MOV32 Control Field (C31:C0)

31 27	26	25 23	22	21 16
Reserved	Control	Reserved	Z Fl. Cond.	Reserved
5	1	3	1	14

15 8	7	6	5	4 3	2 1	0
Reserved	Ext Reg	Init flag	0	Move type	Register select	Res.
14	1	1	1	2	2	1

Figure 23-155. MOV32 Data Field (D31:D0)

31 7	6 0
Data	HR Data
25	7

Cycles One or two cycles

Register modified Selected register (A, B, R, S, or T)

Description MOV32 replaces the selected ALU register and/or the data field values at the remote address location depending on the move type.

Figure 23-156 through Figure 23-159 illustrate these operations. If *no register* is selected, the move is not executed, except for configuration C4:C3 = 01, where the remote data field is written with the immediate data field value.

remote Determines the location of the remote address.

Default: Current instruction + 1.

z_cond When set to OFF the MOV32 performs the move operation specified by the move type whenever it is executed (independent on the state of the Z-Flag).

When set to ON the MOV32 performs the move operation specified by the move type only when the Z-Flag is set.

init (Optional) Determines whether or not system flags are initialized. A value of ON reinitializes the following system flags to these states:

Acceleration flag (ACF) = 0

Deceleration flag (DCF) = 1

Gap flag (GPF) = 0

New angle flag (NAF) = 0

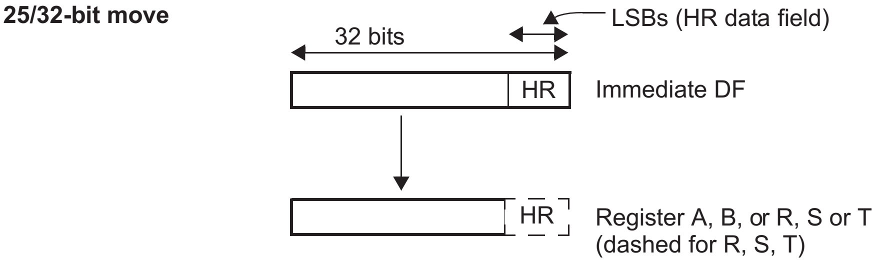
A value of OFF results in no change to the system flags.

type Specifies the move type to be executed.

Table 23-87. Move Type Encoding Selection

Move Type	C4	C3	Source	Destination(s)	Cycles
IMTOREG	0	0	Immediate data field	Register A, B, R, S, or T	1
IMTOREG&REM	0	1	Immediate data field	Remote data field and register A, B, R, S, or T	1
REGTOREM	1	0	Register A, B, R, S, or T	Remote data field	1
REMTOREG	1	1	Remote data field	Register A, B, R, S, or T	2

Figure 23-156. MOV32 Move Operation for IMTOREG (Case 00)



reg Specifies which register (A, B, T, or NONE) is involved in the move. A register (A, B, or T) must be specified for every move type except IMTOREG&REM. If *NONE* is used with move type IMTOREG&REM, the MOV32 executes a move from the immediate data field to the remote data field. If *NONE* is used with any other move type, no move is executed.

data Specifies a 25-bit integer value to be written to the remote data field or selected register.

hr_data (Optional) HR delay. The default value for an unspecified bit is 0.

Figure 23-157. MOV32 Move Operation for IMTOREG&REM (Case 01)

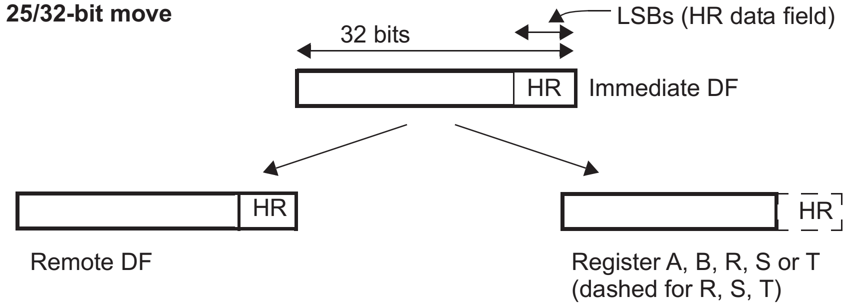


Figure 23-158. MOV32 Move Operation for REGTOREM (Case 10)

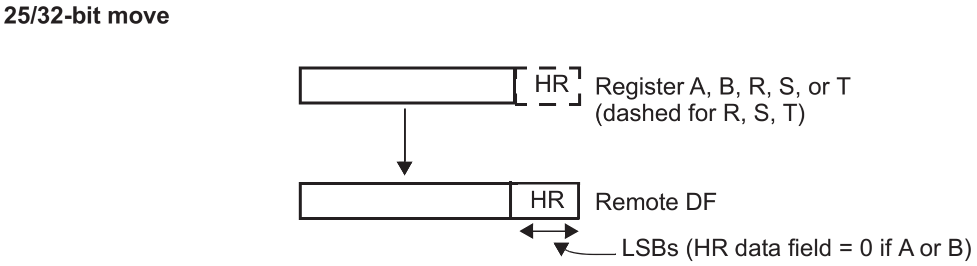
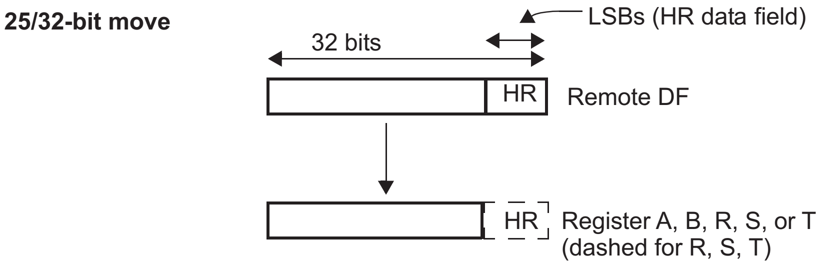


Figure 23-159. MOV32 Move Operation for REMTOREG (Case 11)



Execution

```
If [(z_cond C[22] ==0) OR ((z_cond C[22] == 1) AND (Z Flag == 1))]
{
   switch (type C[4:3])
   {
      case 00: // IMTOREG
          Selected register = Immediate Data Field;
      case 01: // IMTOREG&REM
          Selected register = Immediate Data Field;
          Remote Data Field = Immediate Data Field;
      case 10: // REGTOREM
          Remote Data Field = Selected register;
      case 11: // REMTOREG
          Selected register = Remote Data Field;
   }
}

If (Init Flag == 1)
{
   ACF = 0;
   DCF = 1;
   GPF = 0;
   NAF = 0;
}
else
  All flags remain unchanged;

Jump to Next Program Address;
```

23.6.3.15 MOV64 (Data Move 64)

Syntax

```
MOV64 {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
remote={label | 9-bit unsigned integer}
[request={NOREQ | GENREQ | QUIET}]
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
[cond_addr={label | 9-bit unsigned integer}]
[pin={pin number}]
comp_mode={ECMP | SCMP | MCMP1 | MCMP2}
[action={CLEAR | SET | PULSELO | PULSEHI}]
[reg={A | B | R | S | T | NONE}]
[irq={OFF | ON}]
[data={25-bit unsigned integer]
[hr_data= {7-bit unsigned integer}
}
```

-or-

Syntax

```
MOV64 {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
remote={label | 9-bit unsigned integer}
cntl_val={29-bit unsigned integer}
[data={25-bit unsigned integer]
[hr_data= {7-bit unsigned integer}
}
```

Figure 23-160. MOV64 Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8 0
0	Reserved	BRK	Next program address	0001	Remote Address
6	3	1	9	4	9

Figure 23-161. MOV64 Control Field (C31:C0)

31 29	28 27	26	25 23	22	21 16
Reserved	Request type	Control	Reserved	En. pin action	Conditional address
3	2	1	3	1	9

15 13	12 8	7	6 5	4 3	2 1	0
Conditional address	Pin select	Ext Reg	Comp. mode	Action	Register select	Int. ena
9	5	1	2	2	2	1

Figure 23-162. MOV64 Data Field (D31:D0)

31 7	6 0
Data	HR Data
25	7

Cycles One

Register modified None

Description This instruction modifies the data field and the control field at the remote address.

MOV64 has two distinct syntaxes. In the first syntax, bit values may be set by assigning a value to each of the control fields. This syntax is convenient for modifying control fields that are arranged similarly to the format of the MOV64 control field. A second syntax, in which the entire 29-bit control field is specified by the cntl_val field, is convenient when the remote control field is dissimilar to the MOV64 control field. Either syntax may be used, but you must use one or the either but not a combination of syntaxes. See Figure 23-163.

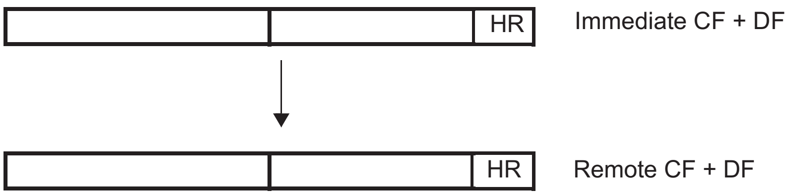


Figure 23-163. MOV64 Move Operation

Table 23-88. MOV64 Control Field Descriptions

request	Maintains the control field for the remote instruction.
control	Maintains the control field for the remote instruction.
en_pin_action	Maintains the control field for the remote instruction.
cond_addr	Maintains the control field for the remote instruction.
pin	Maintains the control field for the remote instruction.
register, ext reg	Maintains the control field for the remote instruction.
comp_mode	Selects the comparison mode type to be used by the remote instruction.

Table 23-88. MOV64 Control Field Descriptions (continued)

action	Maintains the control field for the remote instruction.
irq	Maintains the control field for the remote instruction.
data	Specifies the 25-bit initial count value for the data field. If omitted, the field defaults to 0.
hr_data	(Optional) HR delay. The default value for an unspecified bit is 0.

Table 23-89. Comparison Type Encoding Format

comp_mode	C[6]	C[5]	MCMP Order
ECMP	0	0	
SCMP	0	1	
MCMP1	1	0	REG_GE_DATA
MCMP2	1	1	DATA_GE_REG

Execution

```
Remote Data Field = Immediate Data Field;
Remote Control Field = Immediate control Field;
Jump to Next Program Address;
```

23.6.3.16 PCNT (Period/Pulse Count)

Syntax

```
PCNT {
[hr_lr={HIGH | LOW}]
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}
[request={NOREQ | GENREQ | QUIET}]
[irq={OFF | ON}]
type={FALL2RISE | RISE2FALL | FALL2FALL | RISE2RISE}
pin={pin number}
[control={OFF | ON}]
[prv={OFF | ON}]
[period={25-bit unsigned integer}]
[data={25-bit unsigned integer]
[hr_data= {7-bit unsigned integer}
}
```

Figure 23-164. PCNT Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8	7 6	5	4 0
0	Request Number	BRK	Next program address	0111	Int. ena	Type select	hr_lr	Pin select
6	3	1	9	4	1	2	1	5

Figure 23-165. PCNT Control Field (C31:C0)

31 29	28 27	26	25	24 0
Res.	Request type	Control	Prv.	Period Count
3	2	1	1	25

Figure 23-166. PCNT Data Field (D31:D0)

31 7	6 0
Data	HR Data
25	7

Cycles One

Register modified Register A

Description This instruction detects the edges of the external signal at loop start and measures its period or pulse duration. The counter value stored in the control field C[24:0] and in the register A is incremented each N2HET loop. PCNT uses the HR structure on the pin to measure an HR period/pulse count value.

hr_lr (Optional) Specifies whether the PCNT instruction captures the HR delay into the HR data field on the selected edge condition. If hr_lr is 0 (HIGH) then PCNT captures the HR delay. if hr_lr is 1 (LOW) then PCNT only captures at loop resolution.

irq (Optional) Specifies whether or not an interrupt is generated. A value of ON sends an interrupt when a new value is captured. If OFF is selected, no interrupt is generated.

type (Optional) Determines the type of counter that is implemented.

Table 23-90. Counter Type Encoding Format

	P7	P6	Period/Pulse Select	Reset On	Capture On
FALL2RISE	0	0	Count low-pulse duration on selected pin	Falling edge	Rising edge
RISE2FALL	0	1	Count high-pulse duration on selected pin	Rising edge	Falling edge
FALL2FALL	1	0	Count period between falling edges on selected pin	Falling edge	Falling edge
RISE2RISE	1	1	Count period between rising edges on selected pin	Rising edge	Rising edge

period Specifies the 25-bit integer value that holds the counter value. The counter value is also stored in register A.

Default: 0.

data 25-bit integer representing the last captured counter value.

Default: 0.

hr_data HR delay.

Default: 0.

If *period-measure* is selected, PCNT captures the counter value into the period/pulse data field [D31:D7] on the selected edge. The HR structure provides HR capture field [D6:D0]. The counter value [C24:C0] is reset on the same edge. The captured period value is a 32-bit value.

If *pulse-measure* is selected, PCNT captures the counter value into the period/pulse count field [D31:D7] on the selected edge. The HR structure provides HR capture field [D6:D0]. The counter value [C24:C0] is reset on the next opposite edge. The captured pulse value is a 32-bit value.

When the overflow count (all 1’s in the counter value) is reached, PCNT stops counting until the next reset edge is detected.

Note: For FALL2FALL/RISE2RISE, the user should always discard the first interrupt/HTU request if interrupt/request are enabled before HET_ON. For both the types, reset edge and capture edge are the same and the interrupt or HTU request is triggered on capture edge (which is nothing but the reset edge). Once the execution unit is enabled, the first edge generates an interrupt but the value of the counter is of no use as this is not the period between 2 edges. So first edge after turning on N2HET is used mainly for resetting the counter and start the period count.

Execution

```
Z = 0;

If (Period C[24:0] != 1FF_FFFFh) {
    Period C[24:0] = Period C[24:0] + 1;
}

Register A = Period C[24:0];

If (specified capture edge detected on selected pin)
{
    Z = 1;

    If (Period value != 1FF_FFFFh)
    {
       HR Capture Value = selected HR counter;
    }
    else
    {
       HR Capture Value = 7Fh;
    }

    If (Interrupt Enable == 1) HETFLG[n] = 1;      /* n depends on address */
    If ([C28:C27] == 01) Generate request on request line [P25:P23];
    If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];
}

If (specified reset edge detected on selected pin)
{
    Period value = 0000000h;
}

Prv bit = Current Logic (Lx) value of selected pin;

Jump to Next Program Address;
```

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 23.2.7.

23.6.3.17 PWCNT (Pulse Width Count)

Syntax

```
PWCNT {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}
[request={NOREQ | GENREQ | QUIET}]
[hr_lr={HIGH | LOW}]
[control={OFF | ON}]
[cond_addr={label | 9-bit unsigned integer}
[en_pin_action={OFF | ON}]
pin ={pin number}
[action={CLEAR | SET | PULSELO | PULSEHI}]
[reg={A | B | T | NONE}]
[irq={OFF | ON}]
[data={25-bit unsigned integer]
[hr_data={7-bit unsigned integer}]
}
```

Figure 23-167. PWCNT Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8	7 6	5 0
0	Request Number	BRK	Next program address	1010	hr_lr	11	Reserved
6	3	1	9	4	1	2	6

Figure 23-168. PWCNT Control Field (C31:C0)

31 29	28 27	26	25 23	22	21 16
Reserved	Request type	Control	Reserved	En. pin action	Conditional address
3	2	1	3	1	9

15 13	12 8	7 5	4 3	2 1	0
Conditional address	Pin select	Reserved	Action	Register select	Int. ena
9	5	3	2	2	1

Figure 23-169. PWCNT Data Field (D31:D0)

31 7	6 0
Data	HR Data
25	7

Cycles One

Register modified Selected register (A, B or T)

Description This instruction defines a virtual timer used to generate variable length pulses. The counter value stored in the data field is decremented unconditionally on each timer resolution until it reaches zero, and it then stays at zero until it is reloaded with a non-zero value.

The specified pin action is performed as long as the count after count value is decremented is greater than 0. The opposite pin action is performed when the count after decrement just reaches 0.

If the hr_lr bit is reset, the opposite pin action will be taken after a HR delay from the next loop resolution clock. If the hr_lr bit is set, the delay is ignored. This delay is programmed in bits [D6:D0].

irq ON generates an interrupt when the data field value reaches 0. No interrupt is generated for OFF.

Default: OFF.

data 25-bit integer value serving as a counter.

hr_data HR delay.

Default: 0.

Execution

```
If (Data field value == 0)
{
   Selected register = 0;
   Jump to Next Program Address;
}

If (Data field value > 1)
{
   Selected register = Data field value - 1;
   Data field value = Counter value - 1;

   If (Enable Pin action == 1)
   {
      Selected Pin = Pin Action AT next loop resolution clock;
   }

   Jump to Next Program Address;
}

If (Data field value == 1)
{
   Selected register = 0000000h;
   Data field value = 0000000h;

   If (Opposite action == 1)
   {
      If (hr_lr bit == 0)
      {
         If (Enable Pin action == 1)
         {
              Selected Pin = Opposite level of Pin Action AT next loop resolution clock
              + HR delay;
         }
      }
      else
      {
         If (Enable Pin action == 1)
         {
            Selected Pin = Opposite level of Pin Action AT next loop
            resolution clock;
         }
      }

      If (Interrupt Enable == 1) HETFLG[n] = 1;      /* n depends on address */
      If ([C28:C27] == 01) Generate request on request line [P25:P23];
      If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];

   }

   Jump to Conditional Address
}
```

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 23.2.7.

23.6.3.18 RADM64 (Register Add Move 64)

Syntax

```
RADM64 {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
remote={label | 9-bit unsigned integer}
[request={NOREQ | GENREQ | QUIET}]
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
[cond_addr={label | 9-bit unsigned integer}]
[pin={pin number}]
comp_mode={ECMP | SCMP | MCMP1 | MCMP2}
[action={CLEAR | SET | PULSELO | PULSEHI}]
[reg={A | B | R | S | T | NONE}]
[irq={OFF | ON}]
[data={25-bit unsigned integer]
[hr_data= {7-bit unsigned integer}
}
```

-or-

Syntax

```
RADM64 {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
remote={label | 9-bit unsigned integer}
cntl_val={29-bit unsigned integer}
[data={25-bit unsigned integer]
[hr_data= {7-bit unsigned integer}
}
```

Figure 23-170. RADM64 Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8 0
0	Reserved	BRK	Next program address	0011	Remote Address
6	3	1	9	4	9

Figure 23-171. RADM64 Control Field (C31:C0)

31 29	28 27	26	25 23	22	21 16
Reserved	Request type	Control	Reserved	En. pin action	Conditional address
3	2	1	3	1	9

15 13	12 8	7	6 5	4 3	2 1	0
Conditional address	Pin select	Ext Reg	Comp. mode	Action	Register select	Int. ena
9	5	1	2	2	2	1

Figure 23-172. RADM64 Data Field (D31:D0)

31 7	6 0
Data	HR Data
25	7

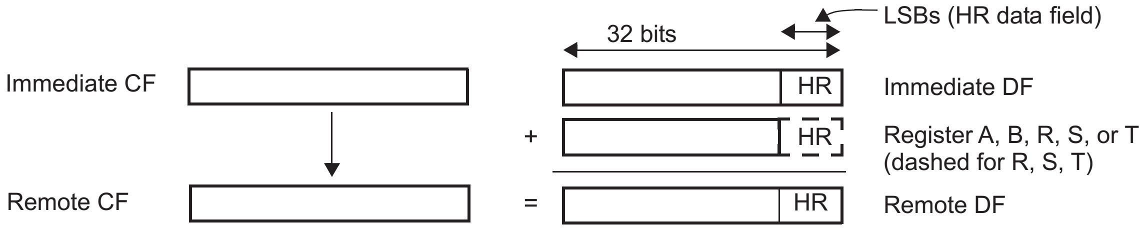
Cycles Normally One Cycle. Two cycles if writing to remote address that is also the next address.

Register modified None

Description This instruction modifies the data field, the HR data field and the control field at the remote address. The advantage over DADM64 is that It executes one cycle faster. In case the R, S, or T register is selected, the addition is a 32-bit addition. The table description shows the bit encoding for determining which ALU register is selected.

RADM64 has two distinct syntaxes. In the first syntax, bit values may be set by assigning a value to each of the control fields. This syntax is convenient for modifying control fields that are arranged similar to the format of the RADM64 control field. A second syntax, in which the entire 29-bit control field is specified by the cntl_val field, is convenient when the remote control field is dissimilar from the RADM64 control field. Either syntax may be used, but you must use one or the either but not a combination of syntaxes. See Figure 23-173.

Figure 23-173. RADM64 Add and Move Operation



comp_mode Selects the comparison mode type to be used.

Table 23-91. Comparison Type Encoding Format

comp_mode	C[6]	C[5]	MCMP Order
ECMP	0	0	
SCMP	0	1	
MCMP1	1	0	REG_GE_DATA
MCMP2	1	1	DATA_GE_REG

Table 23-92. RADM64 Control Field Descriptions

request	Maintains the control field for the remote instruction.
Control	Maintains the control field for the remote instruction.
en_pin_action	Maintains the control field for the remote instruction.
cond_addr	Maintains the control field for the remote instruction.
pin	Maintains the control field for the remote instruction.
register	Maintains the control field for the remote instruction.
action	Maintains the control field for the remote instruction.
irq	Maintains the control field for the remote instruction.
data	Specifies the 25-bit initial value for the data field. If omitted, the field defaults to 0.
hr_data	Seven least significant bits of the 32-bit data field. Default: 0.
cntl_val	Specifies the 29 least significant bits of the Control field.

Execution

```
Remote Data Field = Selected register + Immediate Data Field (including HR field);
Remote Control Field = Immediate Control Field;
Jump to Next Program Address;
```

23.6.3.19 RCNT (Ratio Count)

Syntax

RCNT {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[control={OFF | ON}]
divisor={25-bit unsigned integer}
[data={25-bit unsigned integer]
}

Figure 23-174. RCNT Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8	7 6	5 4	3 1	0
0	Reserved	BRK	Next program address	1010	Res.	00	Step width	Res.	1
6	3	1	9	4	1	2	2	3	1

Figure 23-175. RCNT Control Field (C31:C0)

31 27	26	25	24 0
Reserved	Control	Res.	Divisor
5	1	1	25

Figure 23-176. RCNT Data Field (D31:D0)

31 7	6 0
Data	Reserved
25	7

Cycles Two Cycles (One Cycle if T=0)

Register modified None

Description RCNT is used with other instructions to convert an input period measurement T_{Input} to the form of (Equation 35) where the input period is expressed as a fraction of a reference period $T_{Reference}$.

$$T_{Input} = T_{Reference} \bullet \left(\frac{N}{M}\right) \tag{35}$$

RCNT computes the numerator N of (Equation 35). The denominator M of (Equation 35) is a constant that is of interest. For example, choosing M = 100 allows the input period to be expressed as a percentage (%) of the reference period. Note that if $T_{Input} > T_{Reference}$, then RCNT will return N > M ; which would be correct if, for example, the input pulse period is 110% of the reference pulse period.

RCNT expects that register T is loaded with the value of $T_{Reference}$. The input period T_{Input} is determined by counting the number of loop resolution periods between edges on the input pin. This information is conveyed through the Z flag from a PCNT instruction that precedes the RCNT instruction.

The divisor field of the RCNT instruction should be chosen as: `Divisor = M ·          lr` , where M is the desired denominator from (Equation 35) and lr is the loop resolution prescale value.

An example N2HET program that makes use of the RCNT instruction is:

```
L0: MOV32 { remote=dummy,type=IMTOREG,reg=T,data=0x8,hr_data=0};

L1: PCNT { hr_lr=HIGH,brk=OFF,type=FALL2FALL,pin=0};

L2: RCNT { divisor=320,data=0x4};

L3: BR {cond_addr=L5, event = Z}

L4: ADC { src1=ZERO,src2=IMM,dest=IMM,next=L0,data=0,hr_data=0};

L5: ADD { src1=REM,src2=ZERO,dest=IMM,remote=L4,data=0,hr_data=0};

L6: ADD { src1=ZERO,src2=ZERO,dest=NONE,rdest=REM,
           next=L0,remote=L4,data=0,hr_data=0};

dummy
```

In this small program an input signal on pin 0 is measured both in terms of absolute cycles by the PCNT instruction at L1 and as in 1/10ths of the reference period by the RCNT instruction at L2. In this example the reference period is a constant 0x400 cycles; this value is loaded into register T by the MOV32 instruction at L0. (0x400 is data=8, hr_data=0)

RCNT follows PCNT and is initialized to a working count of T/2 (0x200) whenever the PCNT instruction detects a falling edge on pin 0. Between falling edges on pin0, RCNT accumulates counts 10x faster than PCNT; so that the working data field of RCNT will reach the reference value of 0x400 in 1/10th the time that a PCNT instruction would. Each time the RCNT instruction passes the reference value, it sets the carry out flag and subtracts the reference value from the working count. By accumulating carry-outs from RCNT, the add with carry instruction at L4 effectively counts in increments of 1/10th of the reference period. Note that the divisor value 320 is 10 times 32; this assumes lr=32.

When the next falling edge is detected on pin 0, PCNT sets the Z flag and the RCNT instruction resets again to the initial data field of T/2. RCNT does not modify the Z flag, so that the branch instruction at L3 can execute instructions at L5, L6 instead of L4. The instructions at L5 and L6 capture the final result from L4 and reset the ADC instruction at L4 to zero for the start of the next period measurement.

Execution

```
If (register T[31:0] != 00000000h)
{
   C = 0;

   If (Z == 0)
   {
       Data Field[31:0] = Data Field[31:0] + Divisor[24:0];

       If (Data Field[31:0] >= Reg T[31:0])
       {
          Data Field[31:0]=Data Field[31:0] - Reg T[31:0];
          C = 1;
       }
   }
   else
   {
      Data Field[31:0] = T[31:0] >> 1; /* T/2 */
   }
}
Jump to Next Program Address;
```

23.6.3.20 SCMP (Sequence Compare)

Syntax

SCMP {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}
[request={NOREQ | GENREQ | QUIET}]
[control={OFF | ON}]
[en_pin_action={OFF | ON}]
cond_addr={label | 9-bit unsigned integer}
pin ={pin number}
[action={CLEAR | SET}]
[restart={OFF | ON}]
[irq={OFF | ON}]
[data={25-bit unsigned integer]
}

Figure 23-177. SCMP Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8 0
0	Request Number	BRK	Next program address	0000	Reserved
6	3	1	9	4	9

Figure 23-178. SCMP Control Field (C31:C0)

31 29	28 27	26	25	24 23	22	21 16
Reserved	Request type	Control	Cout prv	Reserved	En. pin action	Conditional address
3	2	1	1	2	1	9

15 13	12 8	7	6 5	4	3 2	1	0
Conditional address	Pin select	Res.	01	Action	Reserved	Restart enable	Int. ena
9	5	1	2	1	2	1	1

Figure 23-179. SCMP Data Field (D31:D0)

31 7	6 0
Data	Reserved
25	7

Cycles One

Register modified Register T (implicitly)

Description This instruction alternately performs angle- and time-based operations to generate pulse sequences, using the angle referenced time base. These pulse sequences last for a relative duration using a free running time base. Generally, register B holds the angle values and register A holds the time values. Bit 0 of the conditional address field (C13) specifies whether the instruction is operating in angle or time operation mode.

When the compared values match in angle mode, a pin can be set or reset according to the pin action bit (C4). The pin does not change states if the enable pin action bit (C22) is reset.

The restart enable bit (C1) provides the option to unconditionally restart a sequence using the X-flag bit of ACMP.

restart If restart is set to ON and the X flag = 1, the assembler writes a value of 1 into the immediate index field, writes the value in register A into the immediate data field, and jumps to the next program address. The X flag is set or cleared by the ACMP instruction. If restart is set to OFF, the X flag is ignored; no special action is performed.

Default: OFF.

irq ON generates an interrupt if the compare match occurs in angle mode. No interrupt is generated when the field is OFF.

Default: OFF.

data Specifies the 25-bit compare value.

cond_addr Since the LSB of the conditional address is used to select between time mode and angle mode, and since the conditional address is taken only in time mode, the destination for the conditional address must be odd.

Execution

```
If (Data field value <= Selected register value) Cout = 0; else Cout = 1;

If (Restart Enable == 1 AND X == 1)
{
   C13 = 1;
   Immediate Data Field = Register A;
   Cout = 0;
   Jump to Next Program Address;
}

If (Angle Mode (C13 == 0) AND ((Restart En. == 1 AND X == 0) OR Restart En. == 0))
{
   If (Z == 0 AND (Register B value - Angle Inc. < Data field value) AND Cout == 0) OR
      (Z == 1 AND (Cout_prv == 1 OR Cout == 0)))
   {
      If (Enable Pin Action == 1) Selected Pin = Pin Action;
      If (Interrupt Enable == 1) HETFLG[n] = 1;      /* n depends on address */
      If ([C28:C27] == 01) Generate request on request line [P25:P23];
      If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];

      Immediate Data Field = Register A;
      C13 = 1; /*** switch to Time Mode ***/

   }
   Jump to Next Program Address;
}
Else If (Time Mode (C13 == 1)) AND ((Restart En. == 1 AND X == 0) OR Restart En. == 0)
{
   /* Result of subtract must not exceed 2^24 - 1 */
   Register T = Register A - Immediate Data Field;
   Jump to Conditional Program Address;
}
Cout_prv = Cout; (always executed)
```

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 23.2.7.

23.6.3.21 SCNT (Step Count)

Syntax

SCNT {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
step={8 | 16 | 32 | 64}
[control={OFF | ON}]
gapstart={25-bit unsigned integer}
[data={25-bit unsigned integer]
}

Figure 23-180. SCNT Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8	7 6	5 4	3 1	0
0	Reserved	BRK	Next program address	1010	Res.	00	Step width	Res.	1
6	3	1	9	4	1	2	2	3	1

Figure 23-181. SCNT Control Field (C31:C0)

31 27	26	25	24 0
Reserved	Control	Res.	Gap start
5	1	1	25

Figure 23-182. SCNT Data Field (D31:D0)

31 7	6 0
Data	Reserved
25	7

Cycles One or two cycles (two cycles when DF is involved in the calculations)

Register modified Register A

Description This instruction can be used only once in a program and defines a specialized virtual timer used after APCNT and before ACNT to generate an angle-referenced time base synchronized to an external signal (that is, a toothed wheel signal) as defined in APCNT and ACNT. Step width selection bits are saved in two flags, SWF0, and SWF1, to be re-used in ACNT.

SCNT multiplies the frequency of the external signal by a constant *K* defined in the step width field, [P5:P4]. The bit encoding for this field is defined in Table 23-93.

step Specifies the step increment to be added to the counter value each program resolution. These two bits provide the values for the SWF0 and SWF1 flags. The valid values are listed in Table 23-93.

Table 23-93. Step Width Encoding for SCNT

P5	P4	Step Width (K)
0	0	8
0	1	16
1	0	32
1	1	64

gapstart Defines the gap start angle, which SCNT writes to register A. The gap start value has no effect on the SCNT instruction, but if the ACNT instruction is being used, register A must contain the correct gap start value. For a typical toothed wheel gear:

GAPSTART = (*stepwidth* × (*actual teeth on gear* - 1)) + 1.

data Specifies the 25-bit integer value serving as a counter.

Default: 0.

This instruction is incremented by the step value K on each timer resolution up to the previous period value P(n-1) measured by APCNT (stored in register T). The resulting period of SCNT is: P(n - 1)/K

Due to stepping, the final count of SCNT will not usually exactly match the target p(n-1). SCNT compensates for this error by starting each cycle with the remainder of the previous cycle.

When SCNT reaches the target p(n-1), the zero flag is set as an increment condition for ACNT.SCNT also specifies a gap start angle, defining the start of a range in ACNT where period measurements in APCNT are temporarily stopped to mask singularities in the external signal.

SCNT uses register A to store the gap start value. Gap start has no effect for SCNT.

Execution

```
SWF1 = P5;
SWF0 = P4;
Z = 0;

If (register T != 0000000h)
{
   If (DCF == 1 OR ACF == 1)
   {
      Data Field register = 0000000h;
      Counter value = 0000000h;
   }

   If (DCF == 0 AND ACF == 0)
   {
      Data Field register = Data field register + Step Width;
   }

   If ((Data Field register - register T) >= 0)
   {
      Data field register = Data Field register - register T;
      Z = 1;
   }

   Register A = Gap start value;

}

Jump to Next Program Address;
```

23.6.3.22 SHFT (Shift)

Syntax

SHFT {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}
[request={NOREQ | GENREQ | QUIET}]
smode={OR0 | OL0 | OR1 | OL1 | ORZ | OLZ | IRM | ILL | IRZ | ILZ}
[control={OFF | ON}]
[prv={OFF | ON}]
[cond_addr={label | 9-bit unsigned integer}
cond={UNC | FALL | RISE}
pin ={pin number}
[reg={A | B | R | S | T | NONE}]
[irq={OFF | ON}]
[data={25-bit unsigned integer]
}

Figure 23-183. SHFT Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8 4	3 0
0	Request Number	BRK	Next program address	1111	Reserved	Smode
6	3	1	9	4	5	4

Figure 23-184. SHFT Control Field (C31:C0)

31 29	28 27	26	25	24 22	21 16
Reserved	Request type	Control	Prv.	Reserved	Conditional address
3	2	1	1	3	9

15 13	12 8	7	6 5	4	3	2 1	0
Conditional address	Pin select	Ext Reg	Shift condition	Res.	0	Register select	Int. ena
9	5	1	2	1	1	2	1

Figure 23-185. SHFT Data Field (D31:D0)

31 7	6 0
Data	Reserved
25	7

Cycles One

Register modified Selected register (A, B, R, S or T)

Description This instruction shifts the data field of the Instruction. N2HET pins can be used for data in or data out. SHFT includes parameters to select the shift direction (in, out, left, right), shift condition (shift on a defined clock edge on HET[0] or shift always), register for data storage (A, B, R, S or T), and the data pin.

smode Shift mode

Table 23-94. SHIFT MODE Encoding Format

smode	P3	P2	P1	P0	Operation	
OR0	0	0	0	0	Shift Out / Right	LSB 1st on HETx / 0 into MSB
OL0	0	0	0	1	Shift Out / Left	MSB 1st on HETx / 0 into LSB
OR1	0	0	1	0	Shift Out / Right	LSB 1st on HETx / 1 into MSB
OL1	0	0	1	1	Shift Out / Left	MSB 1st on HETx / 1 into LSB
ORZ	0	1	0	0	Shift Out / Right	LSB 1st on HETx / Z into MSB
OLZ	0	1	0	1	Shift Out / Left	MSB 1st on HETx / Z into LSB
IRM	1	0	0	0	Shift In / Right	HETx into MSB
ILL	1	0	0	1	Shift In / Left	HETx into LSB
IRZ	1	0	1	0	Shift In / Right	HETx in MSB / LSB into Z
ILZ	1	0	1	1	Shift In / Left	HETx in LSB / MSB into Z

cond Specifies the shift condition.

Table 23-95. SHIFT Condition Encoding

C6	C5	Shift Condition
0	X	Always
1	0	Rising edge of HET[0]
1	1	Falling edge of HET[0]

irq ON generates an interrupt if the Z flag is set. A value of OFF does not generate an interrupt.

Default: OFF.

data Specifies the 25-bit value for the data field.

Execution

```
If (SHIFT condition == 0X)
OR (SHIFT condition == 10 AND HET[0] rising edge)
OR (SHIFT condition == 11 AND HET[0] falling edge)
{
   If ([P3:P2] == 00)
   {
       If ((Immediate Data Field == all 0’s AND [P3:P0] == 000X)
           OR (Immediate Data Field == all 1’s AND [P3:P0] == 001X))
       {
          Z = 1;
       }
       else
       {
          Z = 0;
       }
   }
   else If ([P3:P0] == 1010)
   {
      Z = LSB of the Immediate Data Field;
   }
   else if ([P3:P0] == 1011)
   {
      Z = MSB of the Immediate Data Field;
   }
}

If( (Immediate Data Field == all 0’s) OR
    (Immediate Data Field == all 1’s))
{
   if (Interrupt Enable == 1)
                                          {
      HETFLG[n] = 1;      /* n depends on address */
   }
   Jump to Conditional Address;
}
else
{
   Jump to Next Program Address;
}

Prv. bit = HET[0] Pin level; (Always executed)

Shift Immediate Data Field once according to P[3:0];

Immediate Data Field = Result of the shift;

Selected register = Result of the shift;

Jump to Next Program Address;
```

NOTE: The immediate data field evaluates all 0s or all 1s and is performed before the shift operation.

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 23.2.7.

23.6.3.23 WCAP (Software Capture Word)

Syntax

```
WCAP {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}
[request={NOREQ | GENREQ | QUIET}]
[hr_lr={HIGH | LOW}]
[control={OFF | ON}]
[prv={OFF | ON}]
[cond_addr={label | 9-bit unsigned integer}]
pin ={pin number}
event={NOCOND | FALL | RISE | BOTH}
reg={A | B | R | S | T | NONE}
[irq={OFF | ON}]
[data={25-bit unsigned integer]
[hr_data={7-bit unsigned integer}]
}
```

Figure 23-186. WCAP Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8	7 0
0	Request Number	BRK	Next program address	1011	hr_lr	Reserved
6	3	1	9	4	1	8

Figure 23-187. WCAP Control Field (C31:C0)

31 29	28 27	26	25	24 22	21 16
Reserved	Request type	Control	Prv.	Reserved	Conditional address
3	2	1	1	3	9

15 13	12 8	7	6 5	4 3	2 1	0
Conditional address	Pin select	Ext Reg	Capture condition	Reserved	Register select	Int. ena
9	5	1	2	2	2	1

Figure 23-188. WCAP Data Field (D31:D0)

31 7	6 0
Data	HR Data
25	7

Cycles One

Register modified None

Description This instruction captures the selected register into the data field if the specified capture condition is true on the selected pin. This instruction can be used with all pins.

If the hr_lr bit is reset, the WCAP instruction will capture an HR time stamp into the data field on the selected edge condition. If the hr_lr bit is set, the HR capture is ignored.

event Specifies the event that triggers the capture.

Table 23-96. Event Encoding Format for WCAP

C6	C5	Capture Condition
0	0	Always
0	1	Capture on falling edge
1	0	Capture on rising edge
1	1	Capture on rising and falling edge

irq ON generates an interrupt when the capture condition is met. No interrupt is generated for OFF.

Default: OFF.

data Specifies the 25-bit integer value to be written to the data field or selected register.

hr_data HR capture value.

Default: 0.

NOTE: WCAP in HR Mode: The HR Counter starts on a WCAP instruction execution (in the first loop clock) and will synchronize to the next loop clock. When N2HET is turned on and a capture edge occurs in the first loop clock (where the HR counter hasn’t been synchronized to the loop clock), then the captured HR counter value is wrong and is of no use. So the captured HR data in the first loop clock should be ignored.

Execution

```
If (Specified Capture Condition is true on Selected Pin
OR Unconditional capture is selected)
{
   Immediate Data Field = Selected register value;

   If (hr_lr bit == 0) Capture the HR value in Immediate HR Data Field;

   If (Interrupt Enable == 1) HETFLG[n] = 1;      /* n depends on address */
   If ([C28:C27] == 01) Generate request on request line [P25:P23];
   If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];

   Jump to Conditional Address;
}

Jump to Next Program Address;

Prv bit = Current Logic (Lx) value of selected pin; (always executed)
```

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 23.2.7.

23.6.3.24 WCAPE (Software Capture Word and Event Count)

Syntax

WCAPE {
[brk={OFF | ON}]
[next={label | 9-bit unsigned integer}]
[reqnum={3-bit unsigned integer}
[request={NOREQ | GENREQ | QUIET}]
[control={OFF | ON}]
[prv={OFF | ON}]
[cond_addr={label | 9-bit unsigned integer}
pin ={pin number}
event={NOCOND | FALL | RISE | BOTH}
[reg={A | B | R | S | T | NONE}]
[irq={OFF | ON}]
[ts_data={25-bit unsigned integer]
[ec_data={7-bit unsigned integer}]
}

Figure 23-189. WCAPE Program Field (P31:P0)

31 26	25 23	22	21 13	12 9	8 0
0	Request Number	BRK	Next program address	1000	Reserved
6	3	1	9	4	9

Figure 23-190. WCAPE Control Field (C31:C0)

31 29	28 27	26	25	24 23 22	21 16
Reserved	Request type	Control	Prv.	Reserved	Conditional address
3	2	1	1	3	9

15 13	12 8	7	6 5	4 3	2 1	0
Conditional address	Pin select	Ext Reg	Capture condition	Reserved	Register select	Int. ena
9	5	1	2	2	2	1

Figure 23-191. WCAPE Data Field (D31:D0)

31 7	6 0
Time Stamp	Edge Counter
25	7

Cycles One

Register modified None

Description This instruction captures the selected register into the data field [D31:D7] and increments an event counter [D6:D0] if the specified capture condition is true on the selected pin. This instruction can be used with all pins, but the time stamp [D31:D7] has loop resolution only.

event Specifies the event that triggers the capture.

Table 23-97. Event Encoding Format for WCAPE

C6	C5	Capture Condition
0	0	Always
0	1	Capture on falling edge
1	0	Capture on rising edge
1	1	Capture on rising and falling edge

irq ON generates an interrupt when the capture condition is met. No interrupt is generated for OFF.

Default: OFF.

ts_data Specifies the 25-bit integer value for [D31:D7]

Default: 0.

ec_data Specifies the initial 7-bit integer value for [D6:D0].

Default: 0.

Execution

```
If (Specified Capture Condition is true on Selected Pin
OR Unconditional capture is selected)
{
   Immediate Data Field[31:7] = Selected register value;
   Immediate Data Field [6:0] = Immediate Data Field [6:0] + 1;

   If (Interrupt Enable == 1) HETFLG[n] = 1;      /* n depends on address */
   If ([C28:C27] == 01) Generate request on request line [P25:P23];
   If ([C28:C27] == 11) Generate quiet request on request line [P25:P23];

   Jump to Conditional Address;
}

Jump to Next Program Address;

Prv bit = Current Logic (Lx) value of selected pin; (always executed)
```

The specific interrupt flag that is triggered depends on the address from which the instruction is executed, see Section 23.2.7.

High-End Timer Transfer Unit (HTU) Module

This chapter describes the high-end timer transfer unit (HTU) module. The HTU is similar to the DMA (Direct Memory Access) module, but it is specialized to transfer N2HET (High-End Timer) data to or from the microcontroller RAM.

NOTE: This chapter describes a superset implementation of the HTU module that includes features and functionality that require DMA. Since not all devices have DMA capability, consult your device-specific datasheet to determine applicability of these features and functions to your device being used.

Topic **Page**

24.1 Overview 1250
24.2 Module Operation 1251
24.3 Use Cases 1263
24.4 HTU Control Registers 1266
24.5 Double Control Packet Configuration Memory 1292
24.6 Examples 1299

24.1 Overview

The HET transfer unit is a dedicated direct memory access controller that transfers data between the N2HET RAM and RAM buffers located in the main memory address range. This eliminates time consuming CPU accesses to the N2HET RAM to gather measurement data or creating output waveforms and thus freeing up the CPU to perform other tasks.

24.1.1 Features

- Independently transfers data between the N2HET and the main memory
- 8 double control packets supporting dual buffer configuration
- Transfer requests generated by N2HET instructions/events
- One shot, circular and auto switch buffer transfer modes for each double control packet for flexible buffer handling
- Constant and post-increment addressing modes
- 32- or 64-bit transactions
- Programmable memory protection region
- Parity protect control packet RAM
- Extensive diagnostic functionality

24.2 Module Operation

The HTU is tightly coupled to the N2HET and is not intended to transfer data from other peripheral modules. It initiates transfers with the help of requests generated by the N2HET program and configurable control packets. Figure 24-1 shows a system block diagram of the HTU and the main path for the data transfer. The tight coupling and the dedicated bus into the SCR (Switched Central Resource) reduces the amount of data transferred on the peripheral bus, which increases the overall system performance. However if the application decides to use the direct CPU access method to the N2HET RAM, it is free to do so.

Figure 24-2 shows a more detailed block diagram of the HTU module.

Figure 24-1. System Block Diagram

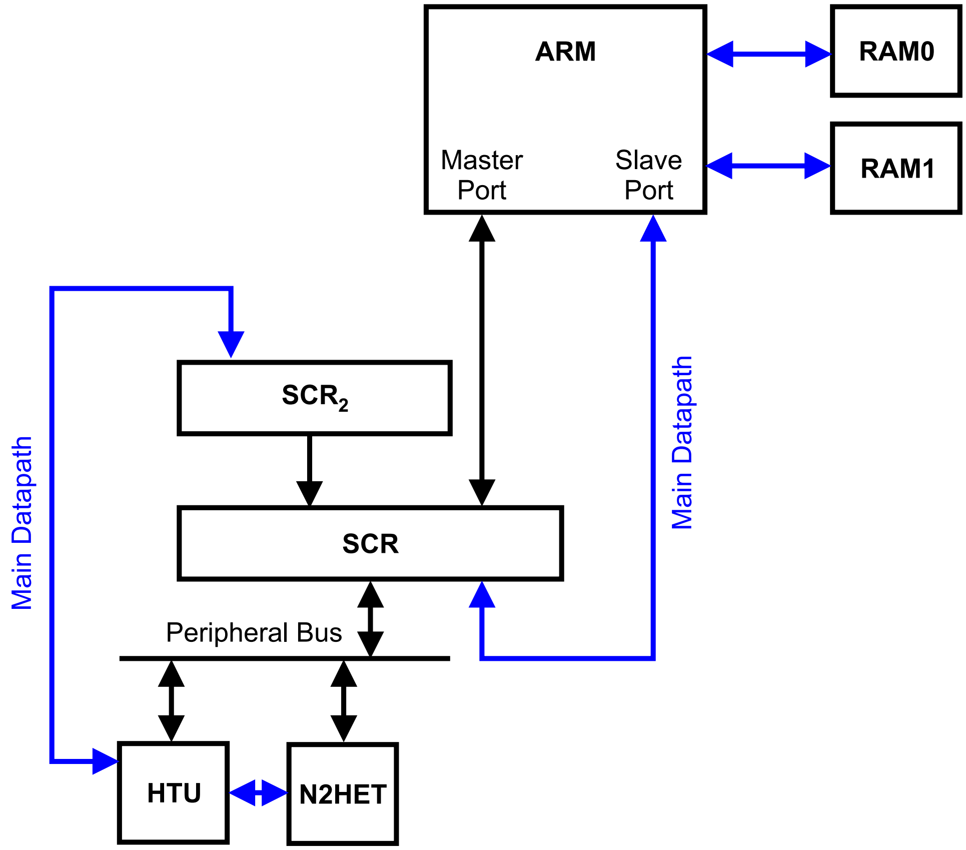
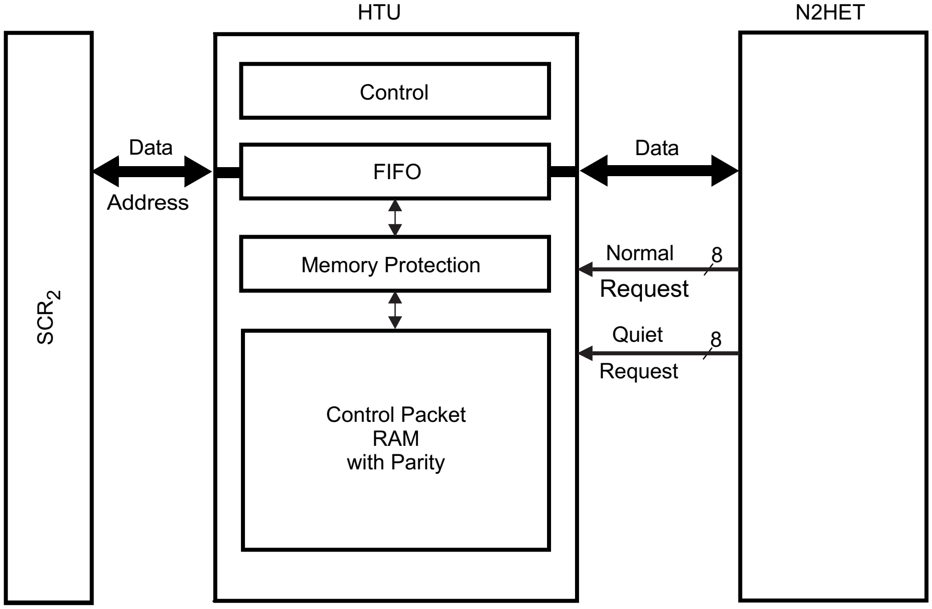


Figure 24-2. HTU Block Diagram



Transfers between N2HET RAM and the main memory are triggered by 8 different normal N2HET requests. Quiet requests are used for specific cases and are discussed in Section 24.2.4.1. Control packets, which store the source and destination addresses, the transfer count and other information (see Section 24.5), are associated with the requests. A FIFO decouples the read- and write-path and allows to do data-packing in the case of different read- and write-data sizes. The application can specify a section of memory into or from which the data is transferred. This serves as memory protection in the case that information in the control packet RAM was unintentionally altered and avoids that the HTU can overwrite important application data.

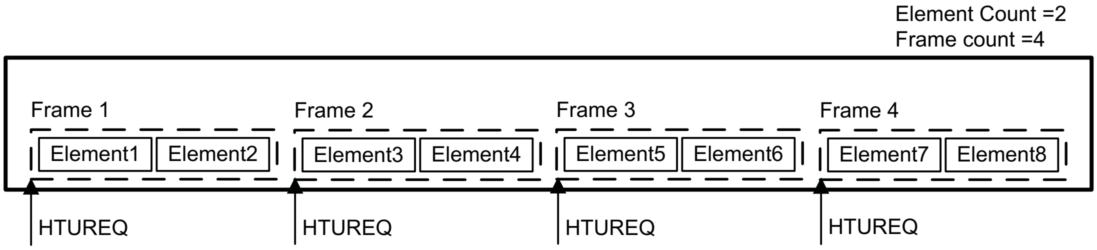
Control packets are implemented as double control packets (DCP) which allow to specify two buffers for the data transfer. This enables the CPU to work with one buffer, while new data is transferred to/from the other buffer.

The control packet defines:

- the start address of the source/destination buffers
- the N2HET instruction address location
- how many elements need to be transferred per request
- the buffer size as the number of elements times the number of frames
- the buffer handling

A transfer is triggered when a certain condition (for example, capture, compare condition) is detected by a N2HET instruction. The N2HET instruction specifies which request line to the HTU will be triggered at the event. The DCPs have a fixed assignment to the request lines and the corresponding assignment can be found in the device datasheet. Once a request is triggered, it starts a frame transfer. A frame can contain one or more elements. Elements are defined as 32-bit or 64-bit words of data.

Figure 24-3. Example of a HTU Transfer



24.2.1 Data Transfers between Main RAM and N2HET RAM

24.2.1.1 Addressing Modes

The addressing modes of a control packet need to be distinguished between the main RAM of the CPU and the N2HET RAM.

Main RAM

For each double control packet (see Section 24.2.1.3), the addressing mode for the main RAM (RAM0/1) can be configured to constant or post-increment mode in register IHADDRCT.

- **Constant Addressing:** In constant mode, the HTU writes/reads the data to/from the same address in the main RAM.
- **Post-increment Addressing:** In post-increment mode, the HTU writes/reads the data to/from the main RAM by incrementing through the addresses after each transfer. If 32-bit transfers are selected it will automatically increment by 4 Byte, if 64-bit transfers are selected, it will increment by 8 Byte. The examples of Use Cases illustrate the post-increment mode, where the elements of consecutive frames are transferred to/from consecutive locations in the main RAM buffer.

N2HET RAM

How a DCP addresses the N2HET RAM is determined by the initial N2HET address, the initial element counter (IETCOUNT) and the N2HET addressing mode (ADDMH). The main difference to the main RAM addressing mode is that the HET address is reset to the initial HET address for every first element of a frame. To implement constant addressing, the initial element counter needs to be set to 1. Post-increment addressing is selected by programming the initial element counter to a value other than 1.

24.2.1.2 Single Buffer Implementation

In a single buffer implementation, the DCP is set up to transfer data to/from a single buffer in the main RAM. With each transfer request, the programmed number of elements is transferred and the buffer pointer is reset to its starting address after the programmed number of frame transfers have completed.

Figure 24-4 shows the request on one request line of the HTU and the frame running on the assigned control packet visualized by the element counter. In the diagram, the frame has 5 element transfers (element count = 5).

Before the application reads the buffer, it has to disable the control packet to avoid that new data overwrites the buffer while it's being accessed by the application. Regardless of the control packet being disabled at t1 or t2 the last frame will always be completed, since the trigger request has been received already. The application can determine any ongoing transfers by the TIPF flag and the NACP bits.

- **One Shot Buffer Mode:** If TMBA or TMBB is set to one shot buffer mode then the data stream will stop after all elements of buffer A or buffer B have been transferred. This means that the corresponding DCP will be disabled after the last frame was transferred to/from buffer A or B and CFTCTA or CFTCTB decrements to 0.
- **Circular Buffer Mode:** If TMBA or TMBB is set to circular buffer mode, then the data stream will continue back at the start of buffer A or B after all elements of buffer A or B have been transferred. The example of Timing Example for Circular Buffer Mode assumes IETCOUNT = 3 (Initial Element Transfer Count), IFTCOUNT = 3 (Initial Frame Transfer Count, SIZE = 0 (Size of Transfer = 32-bit) and ADDFM = 0 (Addressing Mode Main Memory = Post Increment). So there are in total 9 32-bit values in the buffer. It also assumes IFADDRx = 10h. "U" means uninitialized.

Figure 24-4. Single Buffer Timing and Memory Representation

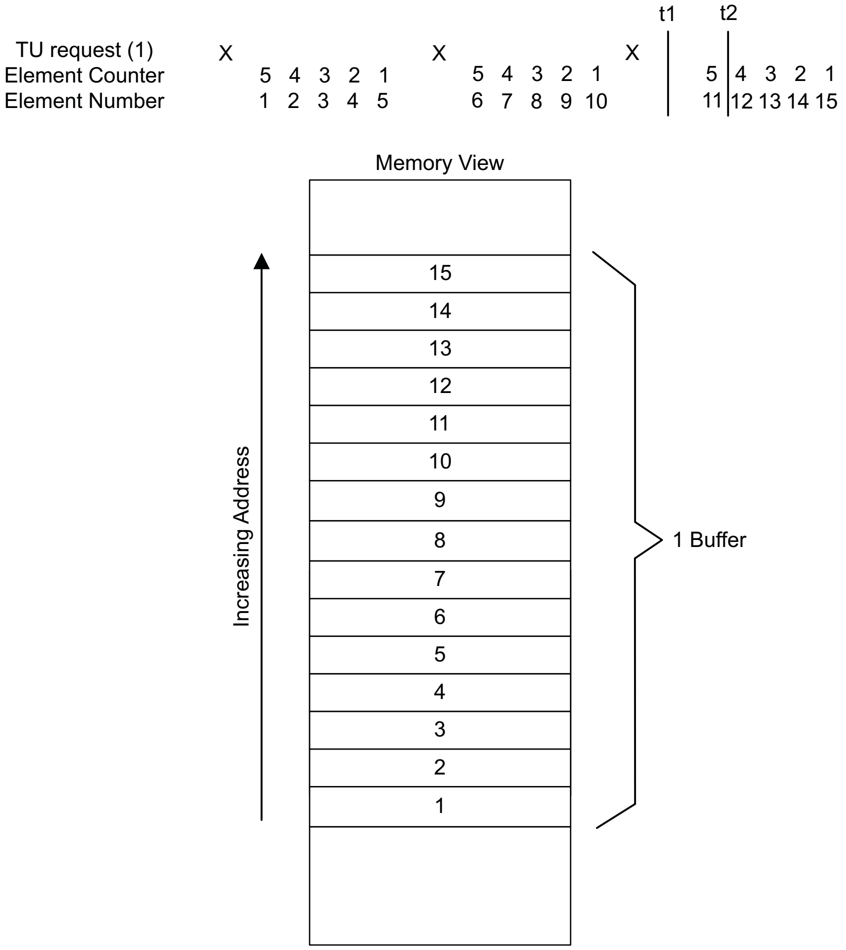
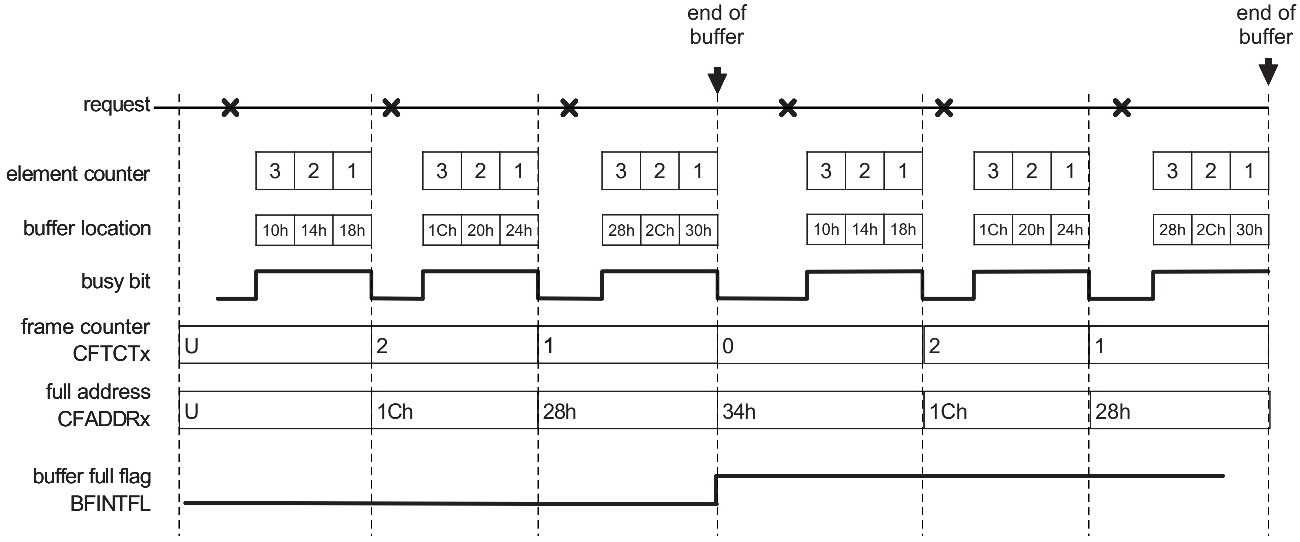


Figure 24-5. Timing Example for Circular Buffer Mode



24.2.1.3 Dual Buffer Implementation

The transfer unit provides **double control packets (DCPs)** supporting the use of two buffers per data stream (per HTU request source). If one buffer should be read by the CPU or DMA, the data stream is directed to the other buffer and the first buffer is frozen. Switching to the other buffer can be triggered with a write access to the CPENA register or with the DCP configured to automatically switch to the other buffer when the programmed number of frames has been transmitted. Freezing the buffer avoids this buffer to be overwritten with new HET data while the CPU or DMA reads this buffer.

Figure 24-6 shows a timing example of two HET instructions 1 and 2, which are the request sources for the HTU (and are controlled by DCP 1 and DCP 2). Each generated frame has 5 element transfers. Request source 1 has two RAM buffers, controlled by two control packets 1A and 1B. Request source 2 has two RAM buffers, controlled by two control packets 2A and 2B.

Figure 24-6. Dual Buffer Timing

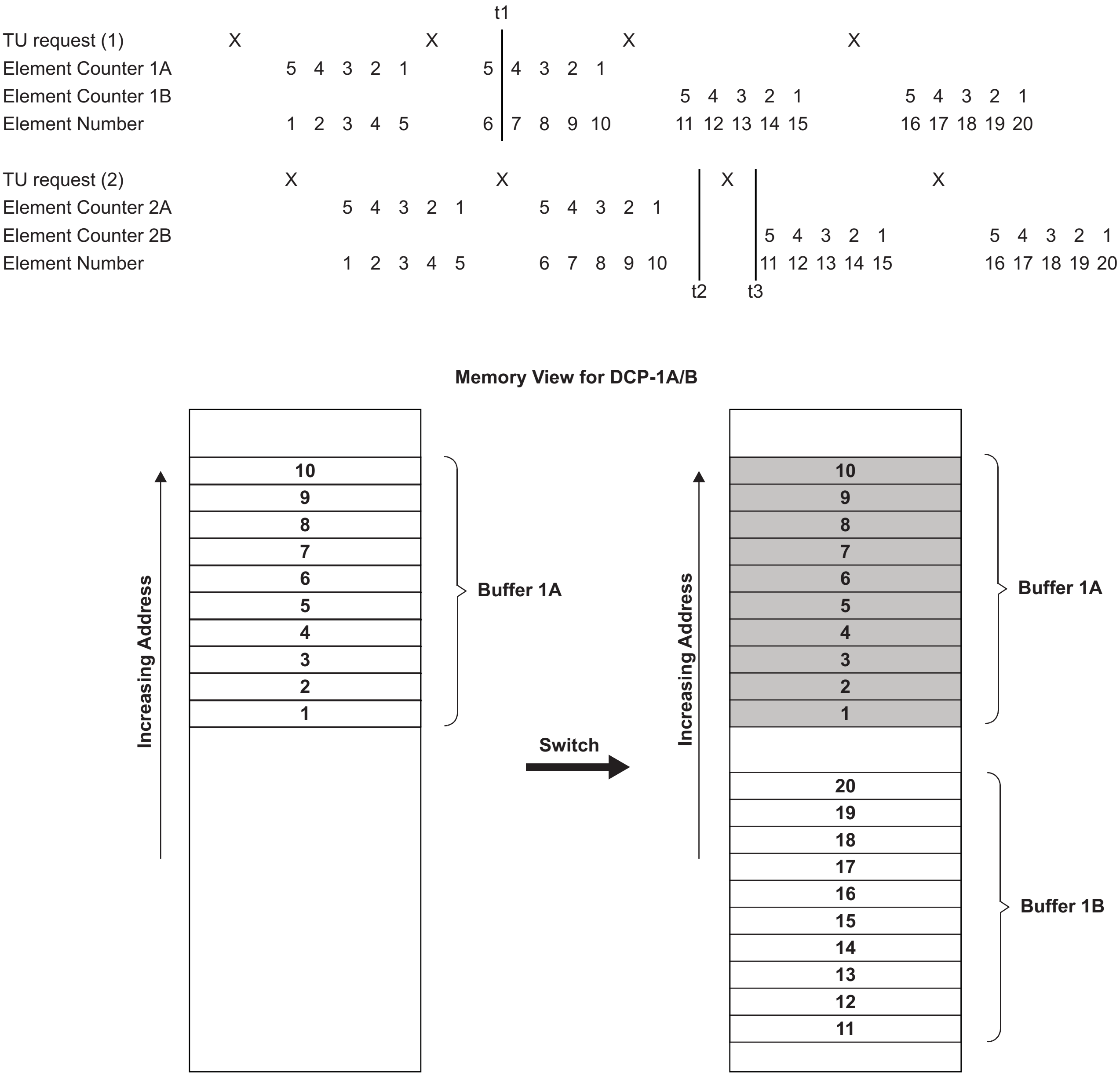

t1
TU request (1) X X X X
Element Counter 1A 5 4 3 2 1 5 4 3 2 1
Element Counter 1B 5 4 3 2 1 5 4 3 2 1
Element Number 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
TU request (2) X X X X
Element Counter 2A 5 4 3 2 1 5 4 3 2 1
Element Counter 2B 5 4 3 2 1 5 4 3 2 1
Element Number 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
t2
t3
Memory View for DCP-1A/B
Increasing Address
10
9
8
7
6
5
4
3
2
1
Buffer 1A
Switch
Increasing Address
10
9
8
7
6
5
4
3
2
1
Buffer 1A
20
19
18
17
16
15
14
13
12
11
Buffer 1B

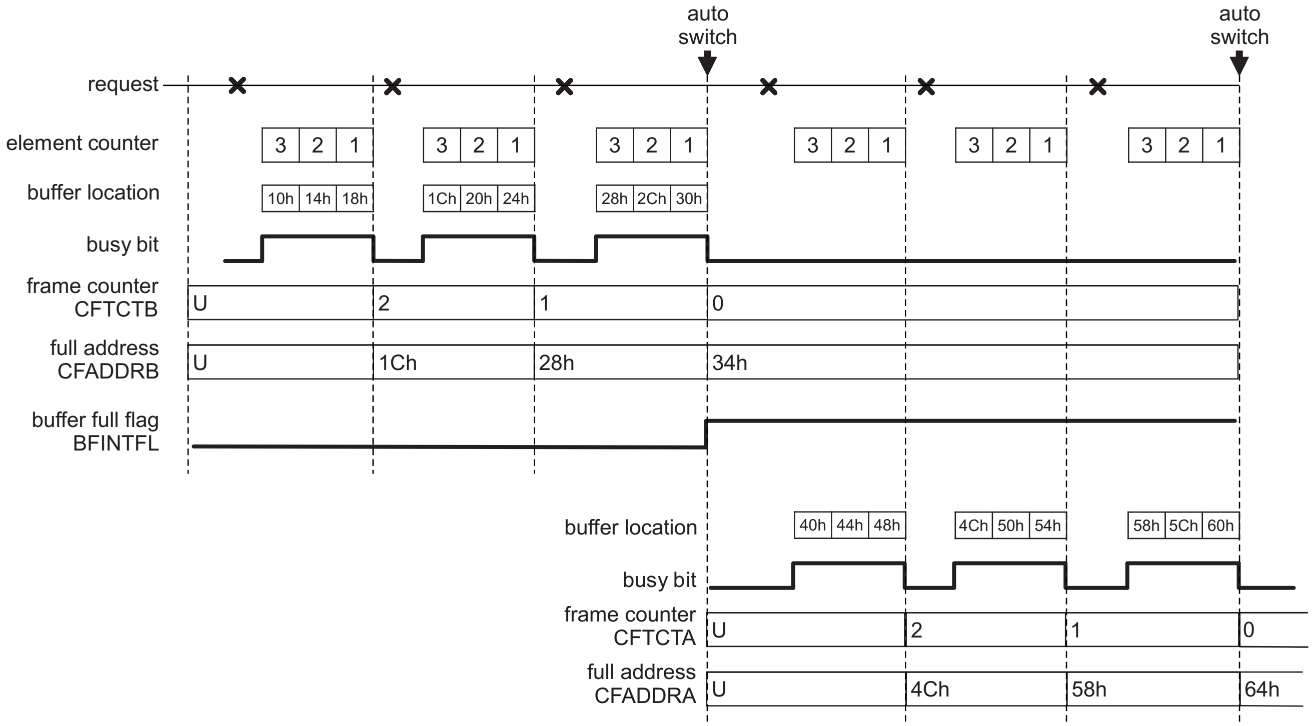
Figure 24-6 shows a switch at time t1, where buffer 1A is frozen and data stream 1 is directed to buffer 1B, but only after the frame has been completed. It also shows the time (t2 or t3) where 2A is frozen and data stream 2 is directed to buffer 2B. If the switch happens between the request and the start of the frame (for example, time t3), then the frame is processed by the new control packet (although the old control packet was active at the time of the request). The delays between the HTU requests and the start of the element transfers result from the fact that the HTU can process only one transfer at a time.

Auto Switch Buffer Mode

If TMBA is set to auto switch mode, then the data stream will continue at the start of buffer B after all elements of buffer A have been transferred. This means that in the CPENA register, CP A is disabled and CP B is enabled automatically and buffer B uses its initial main memory address and initial frame counter to start. The same principle is valid for TMBB and buffer B.

The examples of Figure 24-7 assumes IETCOUNT=3 (Initial Element Transfer Count), IFTCOUNT=3 (Initial Frame Transfer Count, SIZE=0 (Size of Transfer = 32-bit) and ADDFM=0 (Addressing Mode Main Memory = Post Increment). So there are in total 9 32-bit values in buffer A and B. It also assumes IFADDRB=10h and IFADDRA=40h. "U" means uninitialized.

Figure 24-7. Timing Example for Auto Switch Buffer Mode



24.2.1.4 General Control Packet Behavior

The action defined by the selected mode will be performed at the end of the last frame, which has the frame counter value of 1. The one shot and auto switch mode will automatically update the CPENA register at this time. Note, that for all three modes listed above, it is possible to switch to the other buffer by writing to CPENA before the end of the current buffer is reached.

If a write access to CPENA happens while the last frame of DCP x (with frame counter = 1) is transferred then the priority is defined by Table 24-1.

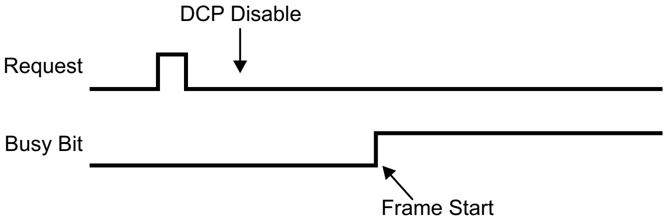
Table 24-1. CPENA / TMBx Priority Rules

Write access to CPENA bits (2 × x+1) and (2 × x) during the frame with frame counter = 1 [1]	Priority Rule
Disable: 01 --> 00 or 10 --> 00	Disabling the DCP by the write to CPENA has priority, TMBx is ignored.
Stay: 01 --> 01 or 10 --> 10	The write access to CPENA is ignored, TMBx has priority and defines the action.
Switch: 01 --> 10 or 10 --> 01	Switching the DCP by the write to CPENA has priority, TMBx is ignored.

[1] See read table of CPENA register (Table 24-14)

There could be a case where the CPU wants to do main memory operations, but does not want the HTU modifying the main memory. It could happen that a request was already active, but the frame transfer hasn't started yet when the application disabled the control packets. The timing diagram in Figure 24-8 shows this scenario.

Figure 24-8. Timing for Disabling Control Packets



Since the request for the transfer was already received before the DCPx is disabled, the HTU will still start the frame transfer. The application would poll the BUSYx bit during the time the DCPx was disabled and before the frame was started and would read a non-busy information. It then would start the main memory operations thinking all transfers have completed, however after some time the HTU will start the outstanding frame transfer and corrupt the main memory.

To avoid this, the application can set the VBUSHOLD bit to disable all transactions between the HTU and the main memory. It has to poll the BUSBUSY bit to ensure that no outstanding transactions on the bus are pending. The HTU will still receive all transfer requests from the N2HET, but it will not be able to transfer any data to or from the main memory, while the VBUSHOLD bit is set.

24.2.2 Arbitration of HTU Elements and Frames

- Frames do not interrupt each other. If a request occurs on DCP x while another frame runs on DCP y (and x ≠ y), then the current frame completes before the new frame starts.
- If two or more request lines are active, the request line with the lower number (specified in the request number field of the corresponding N2HET instruction) is serviced first.

24.2.3 Conditions for Frame Transfer Interruption

If a frame is currently transferred on DCP x and one of the events listed below happens, then the event will (1.) clear the element counter of DCP x, (2.) stop new element transfers on DCP x (3.) clear the active busy bit of DCP x and (4.) disable DCP x in the CPENA register. The DCPs other than DCP x will not be affected.

- Request Lost Error of DCP x (with CORL bit set to 0).
- Parity Error of DCP x (with parity check enabled and COPE bit set to 0). See also Section 24.2.6.
- Bus Error of DCP x.
- Memory Protection Error of DCP x (with memory protection enabled). See also Section 24.2.5.
- Writing a 1 to a BUSY bit (belonging to DCP x) if that bit is 1. There is no effect if the BUSY bit is 0.
- Writing a 1 to the HTURES bit.

When a memory protection error occurs, the access to the protected address is blocked. The frame is stopped before the element, which caused the violation transfer, starts. All other errors will let the current element transfer finish.

In case of the Request Lost and Bus Error, one more element transfer goes on the bus, before the frame is actually stopped. Accordingly, the busy bit is cleared after the element, which follows the element that caused the error.

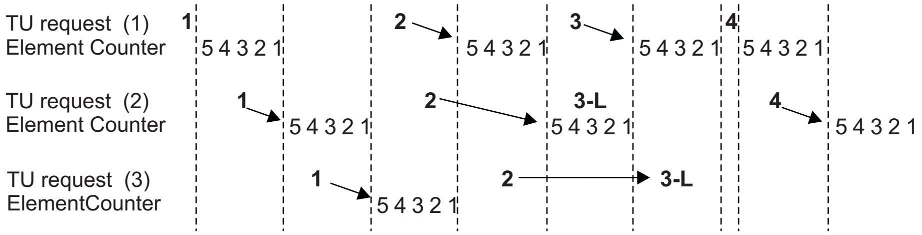
In case of the Bus Error, the counter for the element, which follows the element that caused the error, is captured to the ERRETC register field.

NOTE: If the HTUEN bit is cleared during a frame is transferred, then the frame will be completed before the HTU is disabled.

24.2.4 HTU Overload and Request Lost Detection

If the number of different HTU request sources is "high", the period between the requests is "short" and/or the initial element counter values are "big", then the HTU could get into a overload situation. In Figure 24-9, all requests marked with "L" are lost, since their frame is not completed at the time the next request occurs. Each number in the rows "TU request (x)" represents a time, where the associated N2HET instruction generates a request on DCP x. The arrows in Figure 24-9 point to the associated frame, which could be delayed compared to the request. The delays are caused by different frames, which are currently processed. The figure assumes that the CORL bit in the RLBECTRL register is set, which causes the DCP to stay enabled and let the data stream continue after a request lost error occurred on the DCP (see 3-L for TU request (2)).

Figure 24-9. Timing Example Including Lost Requests



Lost requests are signaled with the RLOSTFL register, and if enabled, can generate request lost interrupts.

If the CORL bit is set, a frame will be completed and the corresponding DCP stays enabled even if a request lost was generated during this frame.

In dual buffer mode, the request lost detection works continuously, independent of the CP switches.

24.2.4.1 Requests and Quiet Requests

In addition to generating too many transfer requests and thus overloading the HTU and not being able to transfer data at all, it can happen that inconsistent data is transferred. The following examples illustrate such scenarios.

In the examples below, the HTU reads a frame of three elements from the datafield of three different instructions. In Figure 24-10, the L3-Instruction generates the HTU request at time t2, t7, and so on. and the according frame (at t3). The frame is delayed because of the HTU load. However, as shown in Figure 24-10, the delay still allows the frame to complete before the datafield of instruction L1 is updated again. However, when the delay is longer (as shown in Figure 24-11), then the frame could fall into the N2HET loop (LRP), in which the N2HET updates the data fields of the L1, L2 and L3 instructions. In this case, the HTU could read inconsistent data as shown in the diagram. A wrong (new) value is read from L1 (at time t3), but correct ("old") values are read from L2 and L3 (at times t4 and t5).

Figure 24-10. Timing that Generates No Request Lost Error

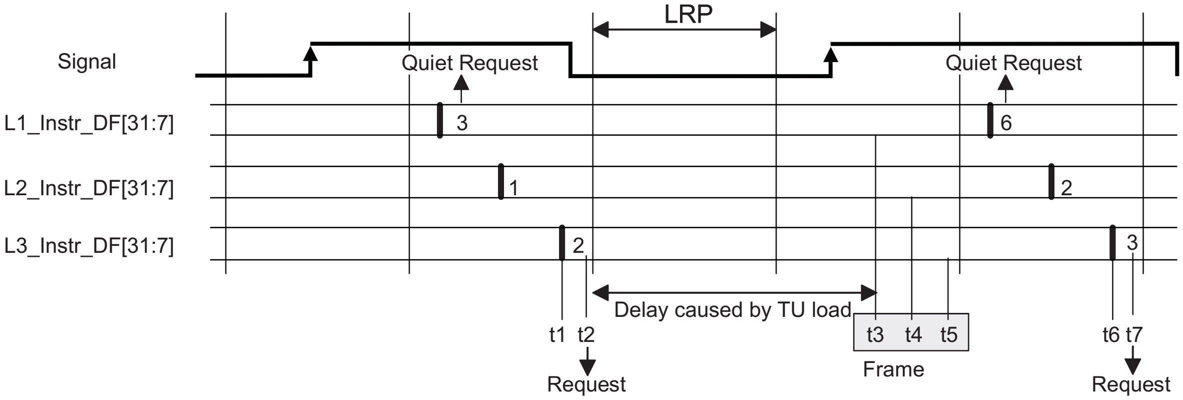
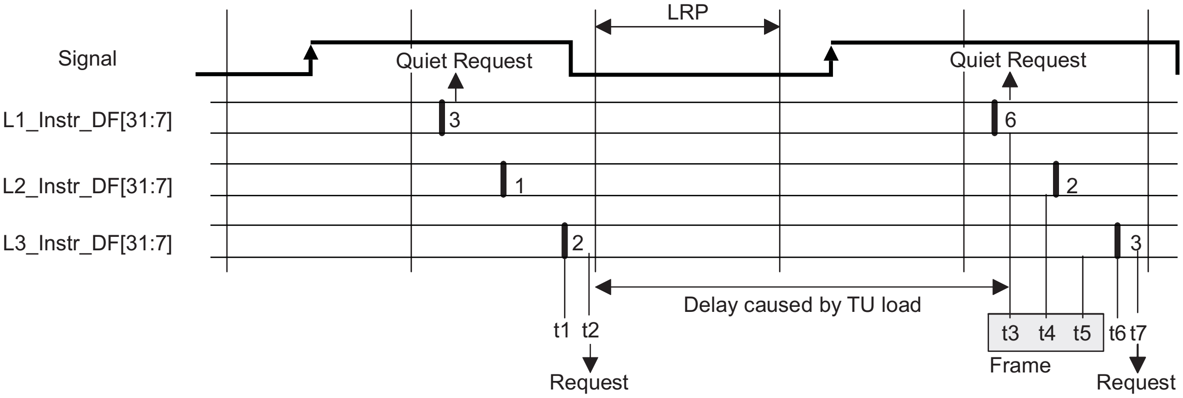


Figure 24-11. Timing that Generates a Request Lost Error



To prevent sending inconsistent data, the N2HET instructions are able to generate a **quiet request**, which does not originate a transfer but is only used by the HTU for consistency check. If a frame has not completed since the last request (or has not even started) at the time the quiet request occurs, then the HTU signals a request lost error. All instructions, which allow to generate a request can be configured to generate a quiet request instead. So in the examples of Figure 24-10 and Figure 24-11, instruction L1 should be configured to generate a quiet request and instruction L3 to generate a normal request. In the case of Figure 24-11, the corresponding bit in the RLOSTFL register will be set.

It is the responsibility of the N2HET software to enable a quiet request for the first instruction of an instruction block, which is addressed by DCP x, and to enable a normal request only for the last instruction of this block. Since enabling the quiet request should enable a proper request lost detection for DCP x, both N2HET instructions need to specify the same DCP x (reqnum=x).

The control fields of the HET instructions provide a 2-bit field to configure one of the following possibilities (as shown in Table 24-2). A 3-bit field in the program field will select which of the 8 Double Control Packets will be triggered by the request.

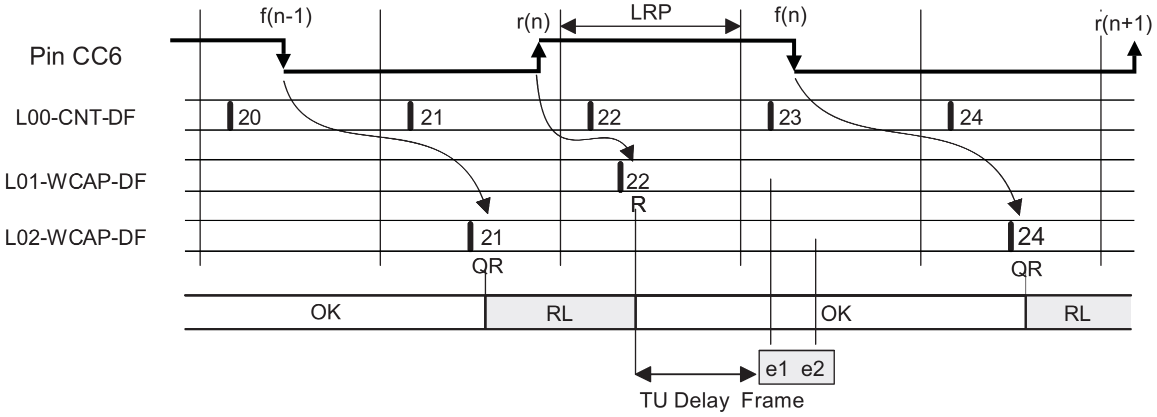
Table 24-2. Triggered Control Packets

Request Type Bit 1	Request Type Bit 0		Request Number
Don't care	0	No request	Specify number 0, 1,... or 7, which selects the HTU or DMA request line.
0	1	Generate normal request	
1	1	Generate quiet request	

In the case of very light HTU load, but higher signal requirements (for example, high frequency), the quiet request could also be used to define periods in which the data read by a control packet is safe. The following HET code will capture counter time stamps to the L1-WCAP data field after rising edges (at pin CC6) and to the L2-WCAP data field after falling edges (at pin CC6):

```
L0 CNT  {reg=A, max=0x1FFFFFF}
L1 WCAP {reqnum=3, request=GENREQ, event=RISE, reg=A, pin=CC6}
L2 WCAP {reqnum=3, request=QUIET, event=FALL, reg=A, pin=CC7}
; HET HRSHARE feature configured to assign both WCAPs to pin CC6
```

Figure 24-12. Timing Example for Two WCAP Instructions



The HTU frame will have two elements: The first gives the time stamp of the rising edge r(n) and the second gives the time stamp of the previous falling edge f(n-1). Using the code above, requests (R) and the quiet requests (QR) will occur at the times shown in Figure 24-12, and a request lost will only be signaled when the frame makes an access during the times marked with RL. So reading [22, 21] as frame elements is correct. If the signal frequency would increase, then a wrong pair [22, 23] could be read, but this will be signaled by a request lost error since at least e2 falls into the RL period.

24.2.5 Memory Protection

This feature allows restricting accesses to certain areas in memory in order to protect critical application data from unintentionally being manipulated by the HTU.

If the HTU memory protection feature is disabled, the full 4 GB address range can be accessed by the HTU without exception. There are two memory regions that start and end addresses can be configured.

With the HTU memory protection feature enabled, read and write accesses by the HTU IFADDRA and IFADDRB registers inside the defined regions are allowed. HTU access to its tightly-coupled memory is independent of the MPU, it routes through the dedicated HTU/N2HET bus using the IHADDR bits in the IHADDRCT register. See Section 24.2 for details on the tightly-coupled bus. For accesses outside the regions, one of two modes is configurable:

- Any access performed by the HTU is forbidden and will be signaled to the ESM module. Write accesses will be blocked.
- Read access is allowed but write access will be blocked and signaled to the ESM module.

To use one region only, REG01ENA must be 0. Bits ACCR01, INTENA01, and register settings of MP1S and MP1E will be ignored.

To use both regions, the following rules must be followed:

1. Memory mapped region 0 covers a lower memory area as Memory mapped region 1.
2. REG01ENA is a 1 and REG0ENA is a 0.
3. ACCR01 is set for the desired access type, ACCR0 is ignored.
4. INTENA01 is set for the desired action, INTENA0 is ignored.

If an element transfer of DCP x generates a memory protection error, then:

1. The element counter of DCP x is cleared.
2. All new element transfers on DCP x are stopped.
3. The active busy bit of DCP x is cleared.
4. DCP x is disabled in the CPENA register. The DCPs other than DCP x will not be affected.
5. The FT flag will be set.
6. An error is signaled to the ESM module.

24.2.6 Control Packet RAM Parity Checking

The HTU module can detect parity errors in the DCP (Double Control Packet) RAM. DCP RAM parity checking is implemented using one parity bit per byte. Even or odd parity checking can be selected in the DEVCR1 register of the system module and can be enabled/disabled by a 4-bit key in the PCR register.

During a read access to the DCP RAM, the parity is calculated based on the data read from the RAM and compared with the good parity value stored in the parity bits. The parity check is performed when the HTU or any other master (for example, CPU) makes a read access to the DCP RAM. A read access within the RAM section of an initial or current DCP checks all 16 bytes of the DCP at a time (see also DCP memory map). For example, if a byte read access happens for DCP RAM address 0, but there is a parity error at byte address Ch then the parity error will occur and the captured parity address will be Ch and not 0. The address of the byte in which the error occurred can be read from the PAR register. If successive DCP RAM read accesses generate multiple parity errors, only the address of the first detected error will be captured and the PAR register will not be updated by subsequent errors until it is read by the application. When multiple errors in a 16 byte word are detected, only the address of the lowest byte will be captured.

The application can decide whether to stop any transfers when a parity error is detected or to continue transferring data. If the COPE (Continue On Parity Error) bit is 0 and parity checking is enabled, then the HTU will not start the frame and the corresponding DCP will be automatically disabled in the CPENA register. If a master other than the HTU (for example, CPU) reads the RAM section of DCP x and a parity error is detected during this read access, while the parity check is enabled and the COPE bit is 0, then the DCP x will be automatically disabled in the CPENA register. If a frame for this DCP x is ongoing during this read access, then in addition the element counter of DCP x is cleared, all new element transfers on DCP x are stopped and the active busy bit of DCP x is cleared. With COPE set to 1 and the parity check enabled, the parity checking will still be performed, but the data transfer of an active DCP continues after a parity error was detected for this DCP. So neither the DCP with the parity error will be disabled nor the frame will be stopped.

After a DCP is enabled (with CPENA using BIM=0), then at the start of the first frame, the HTU performs the parity check only on the initial DCP, since it does not need the current DCP information. For further frames, the HTU performs the parity check for both initial and current DCP, since it needs both information.

On a parity error detection, an error will also be signaled to the ESM module.

24.2.6.1 Parity Bit Mapping and Testing

To test the parity checking mechanism, the parity RAM can be made accessible in order to allow manual fault insertion. Once the TEST bit is set, the parity bits are mapped to address FF4E 0200h.

When in test mode (the parity RAM is accessible), no parity checking will be done when reading from parity RAM, but parity checking will still be performed for read accesses to the DCP RAM.

Table 24-3 and Table 24-4 show how the corresponding parity bits of the DCP RAM bytes are mapped into the memory.

Table 24-3. DCP RAM

Bit	31	24	23	16	15	8	7	0
FF4E 0000h	Byte 0		Byte 1		Byte 2		Byte 3	
FF4E 0004h	Byte 4		Byte 5		Byte 6		Byte 7	
FF4E 0008h	Byte 8		Byte 9		Byte 10		Byte 11	
FF4E 000Ch	Byte 12		Byte 13		Byte 14		Byte 15	

Table 24-4. DCP Parity RAM

Bit		24		16		8		0
FF4E 0200h		P0		P1		P2		P3
FF4E 0204h		P4		P5		P6		P7
FF4E 0208h		P8		P9		P10		P11
FF4E 020Ch		P12		P13		P14		P15

Each byte in DCP RAM has its own parity bit in the DCP Parity RAM. P0 is the parity bit for byte 0, P1 is the parity bit for byte 1, and so on.

24.2.6.2 Initializing Parity Bits

After device power up, the DCP RAM content including the parity bit cannot be guaranteed. In order to avoid parity failures, when reading DCP RAM, the RAM has to be initialized first. This can simply be done by writing known values into the RAM by software and the corresponding parity bit will be automatically calculated.

Another possibility to initialize the DCP memory and its parity bits is to use the system module, which is an on-chip module external to the HTU. This module can start the automatic initialization of all RAMs on the microcontroller including the HTU DCP RAM. This function initializes the complete DCP RAM to 0 when activated by the system module. Depending on the even/odd parity selection, all parity bits will be calculated accordingly. The HTUEN bit must be cleared and the parity functionality must be enabled (by PARITY_ENA) during the automatic DCP RAM initialization. If HTUEN is 1 when the initialization is triggered by the system module, then the initialization will not be performed and the HTU operation is not affected. If a 1 is written to HTUEN during the initialization, then the HTUEN bit will be set but the HTU will not be enabled before the initialization completes.

24.3 Use Cases

24.3.1 Example: Single Element Transfer with One Trigger Request

This example considers the case that the HTU fills a RAM buffer in the main (CPU) data RAM. The HTU reads from the instruction which generates the HTU requests.

This example uses a PCNT instruction. Every time the PCNT has captured a new pulse or period value, it will automatically generate a transfer request to the HTU, which then transfers the value from the N2HET RAM to the buffer RAM. So over time consecutive locations in the RAM buffer can be filled with consecutive measurement values captured into the N2HET RAM data field of the same PCNT instruction without loading or interrupting the CPU.

24.3.2 Example: Multiple Element Transfer with One Trigger Request

The following example shows how the HTU could be used to fill a RAM buffer with a data stream including different types of measurement values belonging to the same N2HET input signal (on one pin): Time stamp values (WCAP), edge counter values (ECNT) and last period values (PCNT).

Figure 24-13 shows the timing and Table 24-5 shows the byte addresses of the program- (PF), control- (CF), data- (DF) and reserved field (res) of the WCAP-ECNT-PCNT instruction block. The timing and code example assumes that all three instructions are assigned to the same N2HET pin.

Figure 24-13. Timing of the WCAP, ECNT, PCNT Example

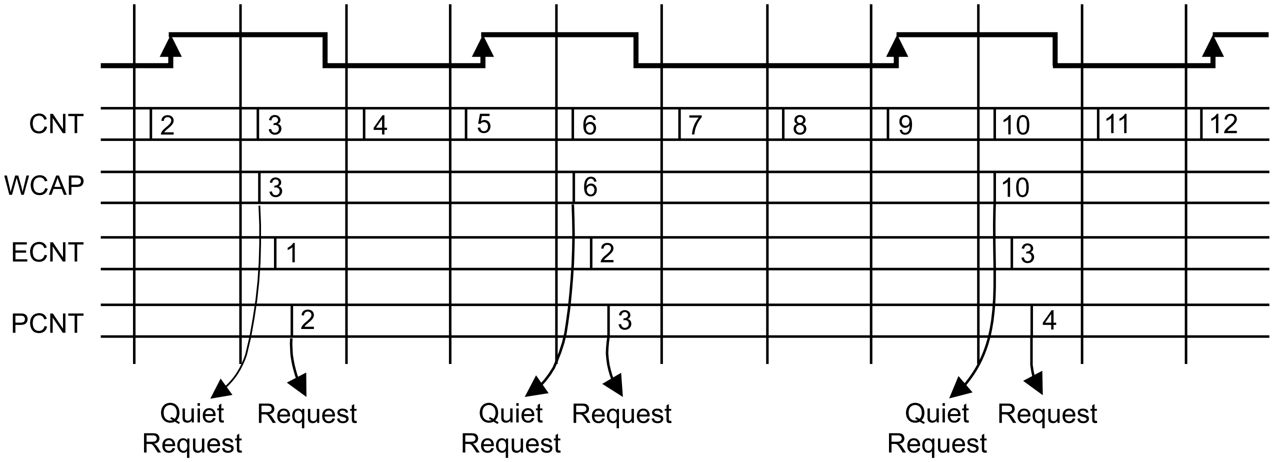


Table 24-5. Field Addresses of the WCAP, ECNT, PCNT Example

	PF	CF	DF	Res
WCAP	30h	34h	38h	3Ch
ECNT	40h	44h	48h	4Ch
PCNT	50h	54h	58h	5Ch

In the HET code the HTU request is enabled only for the last instruction (PCNT) of the WCAP-ECNT-PCNT block. When the PCNT condition is true, it will cause the generated HTU frame to perform three HTU element reads from the data fields of WCAP, ECNT, and PCNT.

32-Bit-Transfer of data fields:

Table 24-6 shows how the internal element counter, frame counter and the address registers change over time for the example described above. Every time the PCNT instruction captures a new value it generates a request to the HTU, which starts a frame. At the end of each frame the frame counter decrements.

Table 24-6. 32-Bit-Transfer of Data Fields[1]

Frame Counter	3			2			1		
Element Counter	3	2	1	3	2	1	3	2	1
Source Address (HET)	38h	48h	58h	38h	48h	58h	38h	48h	58h
Destination Address (main CPU RAM)	70h	74h	78h	7Ch	80h	84h	88h	8Ch	90h

[1] Shows the byte addresses

The destination buffer is filled with the WCAP, ECNT, and PCNT data field values as shown in Table 24-7.

Table 24-7. Destination Buffer Values

Address	Frame Count	Instruction	Value
70h	3	WCAP	3
74h	3	ECNT	1
78h	3	PCNT	2
7Ch	2	WCAP	6
80h	2	ECNT	2
84h	2	PCNT	3
88h	1	WCAP	10
8Ch	1	ECNT	3
90h	1	PCNT	4

The corresponding setup of the HTU control packet for this example is as follows:

```
IHADDR    = 0x38                 // points to WCAP data field
IFADDRA   = 0x70                 // points to buffer
ITCOUNT [frame count = 3] [element count = 3]
IHADDRCT = [DIR: Read HET and write to full address]
           [SIZE: 32 bit]
           [ADDMH: Increment HET address by 16 bytes]
           [ADDMF: Post increment full address mode]
           [Any transfer mode]
```

24.3.3 Example: 64-Bit-Transfer of Control Field and Data Fields

Table 24-8 shows how the internal element counter, frame counter and the address registers change over time assuming the same example as in Section 24.3.2, but now with a transfer size set to 64-bit. The HET address now points to the control field of the instruction, so CF and DF are transferred as 64 bit data.

Table 24-8. 64-Bit-Transfer of Control Field and Data Fields[1]

Frame Counter	3			2			1		
Element Counter	3	2	1	3	2	1	3	2	1
HET (Source) Address	34h	44h	54h	34h	44h	54h	34h	44h	54h
Full (Destination) Address	70h	78h	80h	88h	90h	98h	A0h	A8h	B0h

[1] Shows the byte addresses.

The destination buffer is filled with the WCAP, ECNT, and PCNT control and data field values as shown on the right in Table 24-9.

Table 24-9. Destination Buffer Values

Address	Frame Count	Instruction	Value
70h	3	WCAP	Control Field Value
74h	3	WCAP	3
78h	3	ECNT	Control Field Value
7Ch	3	ECNT	1
80h	3	PCNT	Control Field Value
84h	3	PCNT	2
88h	2	WCAP	Control Field Value
8Ch	2	WCAP	6
90h	2	ECNT	Control Field Value
94h	2	ECNT	2
98h	2	PCNT	Control Field Value
9Ch	2	PCNT	3
A0h	1	WCAP	Control Field Value
A4h	1	WCAP	10
A8h	1	ECNT	Control Field Value
ACh	1	ECNT	3
B0h	1	PCNT	Control Field Value
B4h	1	PCNT	4

The necessary setup of the HTU control packet (see Section 24.5) for this example is as follows:

```
IHADDR   = 0x34  (points to WCAP control field)
IFADDR   = 0x70  (points to buffer)
ITCOUNT [frame count = 3] [element count = 3]
IHADDRCT = [DIR: Read HET and write to full address]
           [SIZE: 64 bit]
           [ADDMH: Increment HET address by 16 bytes]
           [ADDMF: post increment full address mode]
           [Any transfer mode]
```

For different applications, which have the transfer direction set for reading the buffer and writing to HET fields, the 64-bit transfer could be used to change the conditional addresses together with a new data field.

24.4 HTU Control Registers

Table 24-10 provides a summary of the registers. The registers support 8-bit, 16-bit, and 32-bit writes. The offset is relative to the associated peripheral select. See the following sections for detailed descriptions of the registers. The base address for the control registers is FFF7 A400h for HTU1 and FFF7 A500h for HTU2. The address locations not listed, are reserved.

Table 24-10. HTU Control Registers

Offset	Acronym	Register Description	Section
00h	HTU GC	Global Control Register	Section 24.4.1
04h	HTU CPENA	Control Packet Enable Register	Section 24.4.2
08h	HTU BUSY0	Control Packet Busy Register 0	Section 24.4.3
0Ch	HTU BUSY1	Control Packet Busy Register 1	Section 24.4.4
10h	HTU BUSY2	Control Packet Busy Register 2	Section 24.4.5
14h	HTU BUSY3	Control Packet Busy Register 3	Section 24.4.6
18h	HTU ACPE	Active Control Packet and Error Register	Section 24.4.7
20h	HTU RLBECTRL	Request Lost and Bus Error Control Register	Section 24.4.8
24h	HTU BFINTS	Buffer Full Interrupt Enable Set Register	Section 24.4.9
28h	HTU BFINTC	Buffer Full Interrupt Enable Clear Register	Section 24.4.10
2Ch	HTU INTMAP	Interrupt Mapping Register	Section 24.4.11
34h	HTU INTOFF0	Interrupt Offset Register 0	Section 24.4.12
38h	HTU INTOFF1	Interrupt Offset Register 1	Section 24.4.13
3Ch	HTU BIM	Buffer Initialization Mode Register	Section 24.4.14
40h	HTU RLOSTFL	Request Lost Flag Register	Section 24.4.15
44h	HTU BFINTFL	Buffer Full Interrupt Flag Register	Section 24.4.16
48h	HTU BERINTFL	BER Interrupt Flag Register	Section 24.4.17
4Ch	HTU MP1S	Memory Protection 1 Start Address Register	Section 24.4.18
50h	HTU MP1E	Memory Protection 1 End Address Register	Section 24.4.19
54h	HTU DCTRL	Debug Control Register	Section 24.4.20
58h	HTU WPR	Watch Point Register	Section 24.4.21
5Ch	HTU WMR	Watch Mask Register	Section 24.4.22
60h	HTU ID	Module Identification Register	Section 24.4.23
64h	HTU PCR	Parity Control Register	Section 24.4.24
68h	HTU PAR	Parity Address Register	Section 24.4.25
70h	HTU MPCS	Memory Protection Control and Status Register	Section 24.4.26
74h	HTU MP0S	Memory Protection 0 Start Address Register	Section 24.4.27
78h	HTU MP0E	Memory Protection 0 End Address Register	Section 24.4.28

24.4.1 Global Control Register (HTU GC)

Figure 24-14. Global Control Register (HTU GC) [offset = 00]

31 25	24	23 17	16
Reserved	VBUSHOLD	Reserved	HTUEN
R-0	R/WP-0	R-0	R/WP-0

15 9	8	7 1	0
Reserved	DEBM	Reserved	HTURES
R-0	R/WP-0	R-0	R/WP-0

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

Table 24-11. Global Control Register (HTU GC) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	VBUSHOLD		Hold the VBUS bus
		0	The VBUS is not held.
		1	The VBUSHOLD bit holds the bus used to transfer data between the HTU and the N2HET module. When the BUS_BUSY bit is 0 then the bus is no longer busy. While the bus is held, requests will still be accepted. They will be acted upon when the VBUSHOLD is 0. Request lost conditions will be detected and interrupts generated if they are enabled.
23-17	Reserved	0	Reads return 0. Writes have no effect.
16	HTUEN		Transfer Unit Enable Bit
		0	The Transfer Unit is disabled.
		1	The Transfer Unit is enabled.
			The configuration registers and control packets should be set up first before the HTUEN bit is set to 1 to prevent it from carrying out unintended bus transactions. If the HTUEN bit is cleared to 0 during a frame is transferred, then the frame will be completed before the HTU is disabled. The HTUEN bit must be cleared to 0 and the parity functionality must be enabled (by PARITY_ENA) during the automatic DCP RAM initialization (see Initializing Parity Bits). If HTUEN is 1 when the initialization is triggered by the system module, then the initialization will not be performed and the HTU operation is not affected. If a 1 is written to HTUEN during the initialization, then the HTUEN bit will be set but the HTU will not be enabled before the initialization completes. **Note: If HTU is disabled during a frame transfer, then the ongoing current frame will be completed before the HTU module is disabled. If enabled again, then the transfer will restart from the initial frame count for the CP programmed.**
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	DEBM		Debug Mode
		0	The Transfer Unit is stopped in debug mode. The HTU will complete the current frame, but not start any new frames. It will also ignore all requests from the HET and not generate any request lost signals.
		1	The Transfer Unit continues operation in debug mode. **Note:** Since the HET has also an "ignore suspend" bit, there a several possibilities for the behavior of the HET and HTU in suspend mode.
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	HTURES		HTU Software Reset Request
		0	Reset request is not issued to the HTU module. Writing a 0 has no effect.
		1	Reset request is issued to the HTU module.
			Ongoing element transfers will be completed, before resetting the complete HTU module, similar to a hardware reset. The HTURES bit will also be cleared. The recommended order of operations is: • Set the software reset bit. This also clears HTUEN. • Wait for the HTURES bit to clear. • Configure the HTU registers and packets. • Set the HTUEN bit to begin operation.

24.4.2 Control Packet Enable Register (HTU CPENA)

This register enables or disables the individual double control packets (DCP).

Figure 24-15. Control Packet Enable Register (HTU CPENA) [offset = 04h]

31	16
Reserved	
R-0	

15	0
CPENA	
R/WP-0	

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

Table 24-12. Control Packet Enable Register (HTU CPENA) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	CPENA		CP Enable Bits Bits (2*x) and (2*x+1) of CPENA control the double control packet (DCP) x (whereby x must be within 0,1,....,7). See Table 24-13 for write rules. See Table 24-14 for read rules.

Table 24-13. CPENA Write Results

Bit (2*x+1)	Bit (2*x)	Control packets (CP) B and A of DCP x are affected as follows:
0	0	CP B and A are not changed.
0	1	CP B is disabled and CP A are enabled simultaneously.
1	0	CP B is enabled and CP A are disabled simultaneously.
1	1	CP B and CP A are both disabled simultaneously.

Table 24-14. CPENA Read Results

Bit (2*x+1)	Bit (2*x)	State of DCP:
0	0	The DCP is disabled.
0	1	CP B is disabled and CP A is enabled.
1	0	CP B is enabled and CP A is disabled.
1	1	Cannot be read.

- The conditions listed in Section 24.2.3 can automatically disable DCP x. In this case, bits (2*x) and (2*x+1) are both automatically set to 0.
- When bits (2*x) and (2*x+1) change from 00 to 01 or from 00 to 10 caused by a write access to CPENA, then old pending requests on the corresponding request line are cleared. This means only new requests which occur **after** this write access cause the first HTU transfer for this DCP. This is **not** the case when **switching** CPs (from 10 to 01 or from 01 to 10).
- CP A and/or CP B of a DCP can be configured to one-shot, circular or auto-switch transfer mode via the TMBA or TMBB bits in the IHADDRCT control packet configuration. If a write access to CPENA occurs during the last frame of a buffer (with frame counter = 1) then the action defined by the write access to CPENA and the action defined by TMBx can contradict. The priority rules for this case are given in Table 24-1.

24.4.3 Control Packet (CP) Busy Register 0 (HTU BUSY0)

This register displays the status of individual control packets.

Figure 24-16. Control Packet (CP) Busy Register 0 (HTU BUSY0) [offset = 08h]

31 25	24	23 17	16
Reserved	BUSY0A	Reserved	BUSY0B
R-0	R/W1CP-0	R-0	R/W1CP-0

15 9	8	7 1	0
Reserved	BUSY1A	Reserved	BUSY1B
R-0	R/W1CP-0	R-0	R/W1CP-0

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 in privilege mode only to clear the bit; -*n* = value after reset

Table 24-15. Control Packet (CP) Busy Register 0 (HTU BUSY0) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	BUSY0A		Busy Flag for CP A of DCP 0
23-17	Reserved	0	Reads return 0. Writes have no effect.
16	BUSY0B		Busy Flag for CP B of DCP 0
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	BUSY1A		Busy Flag for CP A of DCP 1
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	BUSY1B		Busy Flag for CP B of DCP 1

The bit is **set** when the frame on the according control packet **starts** (as shown in the diagram below, there could be a delay between the request and the start of the frame).

The bit is automatically **cleared** at any of the following conditions:

1. At the end of a frame.
2. Writing a 1 to a BUSY bit (of DCP x) if that bit is 1. This will:
   a. clear the element counter of DCP x
   b. stop all new element transfers on DCP x
   c. clear the busy bit
   d. and disable DCP x in the CPENA register.

   There is no effect, if the BUSY bit is 0.
3. At the conditions listed in Section 24.2.3.

A write access to the CPENA register can stop a control packet (CP) in single buffer mode or it can switch to the other CP of a DCP in dual buffer mode. If stopping or switching occurs while a frame runs on the currently active control packet, the CPU can poll the busy bit to determine when it is safe to read the buffer.

24.4.4 Control Packet (CP) Busy Register 1 (HTU BUSY1)

This register displays the status of individual control packets.

Figure 24-17. Control Packet (CP) Busy Register 1 (HTU BUSY1) [offset = 0Ch]

31 25	24	23 17	16
Reserved	BUSY2A	Reserved	BUSY2B
R-0	R/W1CP-0	R-0	R/W1CP-0

15 9	8	7 1	0
Reserved	BUSY3A	Reserved	BUSY3B
R-0	R/W1CP-0	R-0	R/W1CP-0

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 in privilege mode only to clear the bit; -*n* = value after reset

Table 24-16. Control Packet (CP) Busy Register 1 (HTU BUSY1) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	BUSY2A		Busy Flag for CP A of DCP 2
23-17	Reserved	0	Reads return 0. Writes have no effect.
16	BUSY2B		Busy Flag for CP B of DCP 2
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	BUSY3A		Busy Flag for CP A of DCP 3
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	BUSY3B		Busy Flag for CP B of DCP 3

See Section 24.4.3 for more details.

24.4.5 Control Packet (CP) Busy Register 2 (HTU BUSY2)

Figure 24-18. Control Packet (CP) Busy Register 2 (HTU BUSY2) [offset = 10h]

31 25	24	23 17	16
Reserved	BUSY4A	Reserved	BUSY4B
R-0	R/W1CP-0	R-0	R/W1CP-0

15 9	8	7 1	0
Reserved	BUSY5A	Reserved	BUSY5B
R-0	R/W1CP-0	R-0	R/W1CP-0

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 in privilege mode only to clear the bit; -*n* = value after reset

Table 24-17. Control Packet (CP) Busy Register 2 (HTU BUSY2) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	BUSY4A		Busy Flag for CP A of DCP 4
23-17	Reserved	0	Reads return 0. Writes have no effect.
16	BUSY4B		Busy Flag for CP B of DCP 4
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	BUSY5A		Busy Flag for CP A of DCP 5
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	BUSY5B		Busy Flag for CP B of DCP 5

24.4.6 Control Packet (CP) Busy Register 3 (HTU BUSY3)

Figure 24-19. Control Packet (CP) Busy Register 3 (HTU BUSY3) [offset = 14h]

31-25	24	23-17	16
Reserved	BUSY6A	Reserved	BUSY6B
R-0	R/W1CP-0	R-0	R/W1CP-0

15-9	8	7-1	0
Reserved	BUSY7A	Reserved	BUSY7B
R-0	R/W1CP-0	R-0	R/W1CP-0

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 in privilege mode only to clear the bit; -*n* = value after reset

Table 24-18. Control Packet (CP) Busy Register 3 (HTU BUSY3) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	BUSY6A		Busy Flag for CP A of DCP 6
23-17	Reserved	0	Reads return 0. Writes have no effect.
16	BUSY6B		Busy Flag for CP B of DCP 6
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	BUSY7A		Busy Flag for CP A of DCP 7
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	BUSY7B		Busy Flag for CP B of DCP 7

24.4.7 Active Control Packet and Error Register (HTU ACPE)

Figure 24-20. Active Control Packet and Error Register (HTU ACPE) [offset = 18h]

31	30-29	28-24	23-20	19-16
ERRF	Reserved	ERRETC	Reserved	ERRCPN
R/W1CP-0	R-0	R-0	R-0	R-0

15	14	13	12-8	7-4	3-0
TIPF	BUSBUSY	Rsvd	CETCOUNT	Reserved	NACP
R-0	R-0	R-0	R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 in privilege mode only to clear the bit; -*n* = value after reset

Table 24-19. Active Control Packet and Error Register (HTU ACPE) Field Descriptions

Bit	Field	Value	Description
31	ERRF		Error Flag
		0	No error occurred.
		1	This bit is set when one of the conditions listed at ERRETC is fulfilled and ERRETC and ERRCPN are captured. Once ERRF is set, it is cleared when reading the upper 16-bit word of the ACPE register or the complete 32-bit register. It is also cleared when writing a 1 to ERRF. ERRF can be used to indicate if ERRETC and ERRCPN contain new unread data.
30-29	Reserved	0	Reads return 0. Writes have no effect.

Table 24-19. Active Control Packet and Error Register (HTU ACPE) Field Descriptions (continued)

Bit	Field	Value	Description
28-24	ERRETC		Error Element Transfer Count If one of the following conditions happens the current element transfer counter of the control packet (specified by ERRCPN) is captured to ERRETC. Please see Section 24.2.3. • Request Lost Error of control packet specified by ERRCPN. This is independent of the CORL bit. • Parity Error of control packet specified by ERRCPN. This requires the parity check to be enabled, but is independent of the COPE bit. • Bus Error of control packet specified by ERRCPN. • Memory Protection Error of control packet specified by ERRCPN. This requires the memory protection to be enabled. • Writing a 1 to a BUSY bit, which belongs to the control packet specified by ERRCPN, if that bit is 1. There is no effect, if the BUSY bit is 0. ERRETC is frozen from being updated until the upper 16-bit word of the ACPE register or the complete 32-bit register is read by the CPU. After this read, the HTU will update ERRETC if one of the above conditions is fulfilled again. During debugging, ERRETC stays frozen even when reading the upper 16-bit word or the 32-bit register.
23-20	Reserved	0	Reads return 0. Writes have no effect.
19-16	ERRCPN		Error Control Packet Number If one of the conditions listed at ERRETC happens the number of the control packet, which caused the condition, is captured to ERRCPN. **Control Packet** — **ERRCPN Value** CP A of DCP x — 2 x CP B of DCP x — 2 x+1 With x = 0,1,...or 7 ERRCPN is frozen from being updated until the upper 16-bit word of the ACPE register or the complete 32-bit register is read by the CPU. After this read, the HTU will update ERRCPN if one of the above conditions is fulfilled again. During debugging, ERRCPN stays frozen even when reading the upper 16-bit word or the 32-bit register. If one of the conditions is fulfilled, ERRETC and ERRCPN are updated simultaneously.
15	TIPF		Transfer in Progress Flag
		0	No transfers are in progress.
		1	A transfer is currently active. This bit is the result of a logical OR function of all BUSYxx flags of the 4 BUSYx registers.
14	BUSBUSY		Bus is Busy
		0	Bus between N2HET and HTU is not busy.
		1	When BUSBUSY is 1, the bus is busy with a transfer. It is different from TIPF above because BUSBUSY will go low after VBUSHOLD is set to 1 and no transfers are pending between the HTU and the main memory. TIPF will remain 1, if a transfer is still pending and VBUSPHOLD is 1.
13	Reserved	0	Reads return 0. Writes have no effect.
12-8	CETCOUNT		Current Element Transfer Count CETCOUNT shows the current element transfer counter for the frame that is currently processed. If the HTU does not currently transfer any frame, CETCOUNT is 0. CETCOUNT is updated after the write part of a transfer. There is a period of up to 7 cycles between the time the CETCOUNT is 0 and the HTU is finished updating the current DCP (and the CPENA registers, if the required conditions are fulfilled).
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	NACP		Number of Active Control Packet Indicates which CP currently processes a frame. **Active or Recent DCP** — **NACP Value** CP A of DCP x — 2 x CP B of DCP x — 2 x+1 With x = 0,1,...or 7 NACP is updated at the time the frame starts on the according CP, and it is updated with a new value when a frame starts on a different CP. Note, that there can be a delay between the request and the start of the frame.

24.4.8 Request Lost and Bus Error Control Register (HTU RLBECTRL)

Figure 24-21. Request Lost and Bus Error Control Register (HTU RLBECTRL) [offset = 20h]

31 — 17	16
Reserved	BERINTENA
R-0	R/WP-0

15 — 9	8	7 — 1	0
Reserved	CORL	Reserved	RLINTENA
R-0	R/WP-0	R-0	R/WP-0

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

Table 24-20. Request Lost and Bus Error Control Register (HTU RLBECTRL) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reads return 0. Writes have no effect.
16	BERINTENA		Bus Error Interrupt Enable Bit
		0	The bus error interrupt is disabled for all DCPs.
		1	The bus error interrupt is enabled for all DCPs.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	CORL		Continue On Request Lost Error
		0	Stop current frame on request lost detection. Please see Section 24.2.3.
		1	If CORL is 1 and DCP x is enabled, then DCP x will stay enabled after a request lost condition on DCP x and element transfers will continue.
7-1	Reserved	0	Reads return 0. Writes have no effect.
0	RLINTENA		Request Lost Interrupt Enable Bit
		0	The request lost interrupt is disabled for all DCPs. Disabling RLINTENA will not clear the flags in the RLOSTFL register.
		1	The request lost interrupt is enabled for all DCPs. If bits are set in the RLOSTFL flag register at the time RLINTENA is (re-) enabled, then the according interrupt(s) will occur (in the order of the priority of the request lines).

24.4.9 Buffer Full Interrupt Enable Set Register (HTU BFINTS)

This registers allows to enable the buffer full interrupts for the different control packets. Reading registers BFINTS and BFINTC will return the same bits indicating the status which interrupt is enabled (1) or disabled (0).

Figure 24-22. Buffer Full Interrupt Enable Set Register (HTU BFINTS) [offset = 24h]

31	16
Reserved	
R-0	

15	0
BFINTENA	
R/WP-0	

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

Table 24-21. Buffer Full Interrupt Enable Set Register (HTU BFINTS) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	BFINTENA		Bus Full Interrupt Enable Bits. If the interrupt for CP A of a DCP is enabled, then the interrupt is generated once buffer A is full, that is, once the frame counter CFTCTA decrements to 0. The same applies for CP B (and CFTCTB).
		0	Interrupt is disabled. Writing a 0 has no effect.
		1	Writing to bit (2*x) enables the interrupt for CP A of DCP x. Writing to bit (2*x+1) enables the interrupt for CP B of DCP x.

24.4.10 Buffer Full Interrupt Enable Clear Register (HTU BFINTC)

This registers allows to disable the buffer full interrupts for the different control packets. Reading registers BFINTS and BFINTC will return the same bits indicating the status which interrupt is enabled (1) or disabled (0)

Figure 24-23. Buffer Full Interrupt Enable Clear Register (HTU BFINTC) [offset = 28h]

31	16
Reserved	
R-0	

15	0
BFINTDIS	
R/WP-0	

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

Table 24-22. Buffer Full Interrupt Enable Clear Register (HTU BFINTC) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	BFINTDIS		Buffer Full Interrupt Disable Bits
		0	Interrupt is disabled. Writing a 0 has no effect.
		1	Writing to bit (2*x) disables the interrupt for CP A of DCP x. Writing to bit (2*x+1) disables the interrupt for CP B of DCP x.

24.4.11 Interrupt Mapping Register (HTU INTMAP)

Figure 24-24. Interrupt Mapping Register (HTU INTMAP) [offset = 2Ch]

31	17	16
Reserved		MAPSEL
R-0		R/WP-0

15	0
CPINTMAP	
R/WP-0	

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

Table 24-23. Interrupt Mapping Register (HTU INTMAP) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reads return 0. Writes have no effect.
16	MAPSEL		Interrupt Mapping Select Bit
		0	If MAPSEL is 0, then one bit of CPINTMAP selects one of two interrupt priorities 0 or 1 for the **buffer full** interrupt for the according CP. The **request lost** and **bus error** interrupts of all CPs are set to priority 0, independent of CPINTMAP.
		1	If MAPSEL is 1, then one bit of CPINTMAP determines if the **buffer full**, **request lost** and **bus error** interrupts of the according CP are assigned either to interrupt line 0 or to 1.
15-0	CPINTMAP		CP Interrupt Mapping Bits
		0	Interrupt of CP A (bit 2-x) of DCP x is mapped to interrupt line 0. Interrupt of CP B (bit 2*x+1) of DCP x is mapped to interrupt line 0.
		1	Interrupt of CP A (bit 2-x) of DCP x is mapped to interrupt line 1. Interrupt of CP B (bit 2*x+1) of DCP x is mapped to interrupt line 1.

24.4.12 Interrupt Offset Register 0 (HTU INTOFF0)

The INTOFF0 register reflects the highest priority interrupt flag bit set in the BERINTFL, RLOSTFL, or BFINTFL flag registers with the appropriate CPINTMAP bit set to 0. The priority order (from high to low) is: BER, RLOST, buffer-full. Interrupts for request lines with lower number have higher priority.

Figure 24-25. Interrupt Offset Register 0 (HTU INTOFF0) [offset = 34h]

31–16
Reserved
R-0

15–10	9–8	7–4	3–0
Reserved	INTTYPE0	Reserved	CPOFF0
R-0	R-0	R-0	R-0

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

Table 24-24. Interrupt Offset Register 0 (HTU INTOFF0) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return 0. Writes have no effect.
9-8	INTTYPE0		Interrupt Type of Interrupt Line 0. Indicates whether a buffer-full, RLOST, or BER interrupt, assigned to interrupt line 0, is currently pending.
		0	No interrupt.
		1h	Interrupt caused by full buffer on CP/DCP specified by CPOFF0.
		2h	RLOST interrupt generated by CP/DCP specified by CPOFF0.
		3h	BER interrupt generated by CP/DCP specified by bits CPOFF0.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	CPOFF0		CP Offset. Indicates for which control packet the interrupt is pending, which is classified by INTTYPE0 and is assigned to interrupt line 0.
		0	DCP 0, CP A
		1h	DCP 0, CP B
		2h	DCP 1, CP A
		3h	DCP 1, CP B
		4h	DCP 2, CP A
		5h	DCP 2, CP B
		6h	DCP 3, CP A
		7h	DCP 3, CP B
		8h	DCP 4, CP A
		9h	DCP 4, CP B
		Ah	DCP 5, CP A
		Bh	DCP 5, CP B
		Ch	DCP 6, CP A
		Dh	DCP 6, CP B
		Eh	DCP 7, CP A
		Fh	DCP 7, CP B

NOTE: Reading CPOFF0 will clear the bit generating the current interrupt from appropriate flag register (BERINTFL, RLOSTFL, or BFINTFL), except when in debug mode where reading CPOFF0 will have no effect on the flag registers.

In order to read INTTYPE0 and CPOFF0 simultaneously, always read this register using word or half-word but not using byte accesses.

24.4.13 Interrupt Offset Register 1 (HTU INTOFF1)

This register is organized identically to the INTOFF0 register. The difference is that INTOFF1 reflects the highest priority interrupt flag bit set in the BERINTFL, RLOSTFL, or BFINTFL flag registers with the appropriate CPINTMAP bit set to 1.

Figure 24-26. Interrupt Offset Register 1 (HTU INTOFF1) [offset = 38h]

31-16
Reserved
R-0

15-10	9-8	7-4	3-0
Reserved	INTTYPE1	Reserved	CPOFF1
R-0	R-0	R-0	R-0

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

Table 24-25. Interrupt Offset Register 1 (HTU INTOFF1) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return 0. Writes have no effect.
9-8	INTTYPE1		Interrupt Type of Interrupt Line 1. Indicates whether a buffer-full, RLOST, or BER interrupt, assigned to interrupt line 1, is currently pending.
		0	No interrupt.
		1h	Interrupt caused by full buffer on CP/DCP specified by CPOFF1.
		2h	RLOST interrupt generated by CP/DCP specified by CPOFF1.
		3h	BER interrupt generated by CP/DCP specified by bits CPOFF1.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	CPOFF1		CP Offset. Indicates for which DCP / CP the interrupt is pending, which is classified by INTTYPE1 and is assigned to interrupt line 1.
		0	DCP 0, CP A
		1h	DCP 0, CP B
		2h	DCP 1, CP A
		3h	DCP 1, CP B
		4h	DCP 2, CP A
		5h	DCP 2, CP B
		6h	DCP 3, CP A
		7h	DCP 3, CP B
		8h	DCP 4, CP A
		9h	DCP 4, CP B
		Ah	DCP 5, CP A
		Bh	DCP 5, CP B
		Ch	DCP 6, CP A
		Dh	DCP 6, CP B
		Eh	DCP 7, CP A
		Fh	DCP 7, CP B

NOTE: Reading CPOFF1 will clear the bit generating the current interrupt from appropriate flag register (BERINTFL, RLOSTFL, or BFINTFL), except when in debug mode where reading CPOFF1 will have no effect on the flag registers.

In order to read INTTYPE1 and CPOFF1 simultaneously, always read this register using word or half-word but not using byte accesses.

24.4.14 Buffer Initialization Mode Register (HTU BIM)

This register enables special applications, where one CP is temporarily disabled, but after having re-enabled the CP, filling the buffer should not start back at its beginning, but should continue after the last element of the previous run.

Table 24-27 shows more details on the BIM usage.

Figure 24-27. Buffer Initialization Mode Register (HTU BIM) [offset = 3Ch]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		BIM	
R-0		R/WP-0	

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

Table 24-26. Buffer Initialization Mode Register (HTU BIM) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	BIM		Buffer Initialization Mode The BIM bits and the TMBx bits determine when a buffer is initialized, that means when its initial full address IFADDRx and its initial frame counter IFTCOUNT is used. When initializing (restarting) a buffer the information in the corresponding initial DCP RAM is loaded to a internal state machine but not to the current DCP RAM (CFADDRx, CFTCTx). The current DCP RAM is updated the first time when the first frame has finished. A buffer is initialized: • In circular buffer transfer mode (defined by TMBx) when the end of the buffer is reached. • When CPs are switched or enabled according to Buffer Initialization. The CPENA bits (2*x+1) and (2*x) are changed by write access to CPENA. For the first two rows of the table, the change of the CPENA bits could also be the result of the auto switch feature (as defined by TMBx). BIM bit x only affects DCP x (with x = 0,1,...or 7).

Table 24-27. Buffer Initialization

Case	Change of CPENA bits (2*x+1) and (2*x)		Action on buffer A or B (of DCP x)		
	Old state [1]	New state [2]		BIM bit x = 0 (normal mode)	BIM bit x = 1 (special mode)
A	01	10	Switch from CP A to B	Next frame starts at the initial address of buffer B [3]	Same as for BIM bit x = 0
B	10	01	Switch from CP B to A	Next frame starts at the initial address of buffer A [3]	Same as for BIM bit x = 0
C	01	01	Stay at CP A	Write to CPENA bits (2*x+1) and (2*x) is ignored	Same as for BIM bit x = 0
E	10	10	Stay at CP B	Write to CPENA bits (2*x+1) and (2*x) is ignored	Same as for BIM bit x = 0
E	00	01	Enable CP A	Next frame starts at the **initial** address of buffer A	*Next frame continues at the **current** address of buffer A*
F	00	10	Enable CP B	Next frame starts at the **initial** address of buffer B	*Next frame continues at the **current** address of buffer B*
G	xx	11	Disable both CPs	Stop DCP x	Same as for BIM bit x = 0

[1] See read table of CPENA register (Table 24-14).

[2] See write table of CPENA register (Table 24-13).

[3] This is regardless of whether the switch is done by a write access to CPENA or by the auto-switch feature.

NOTE: For cases E and F above, after the last frame of a buffer, the HTU sets CFTCTx to 0 and CFADDRx to the next address after the buffer. If the DCP was disabled during this state, then both CFTCTx and CFADDRx would contain invalid initialization values. Therefore, if a DCP should continue at its current address, then the software should use one of the following two procedures before it (re-) enables the DCP (as per Table 24-27):

1.
```
If CFTCTx ≠ 0 then set BIM=1
If CFTCTx = 0 then set
                BIM=0
```
2.
```
If CFTCTx ≠ 0 then set
                BIM=1
If CFTCTx = 0 then {set
                BIM=1;
```

```
set CFTCTx = IFTCOUNT;
set CFADDRx = IFADDRx}
```

But note that these procedures are only required for the cases E and F and not for all the other cases shown in Table 24-27. Also, when a buffer reaches its end in circular mode, it uses the initial DCP information to restart independently of the BIM setting (assuming it is not temporarily disabled during CFTCTx = 0).

NOTE: Similarly, care needs to be taken when BIM is set to 1 and a DCP is enabled for the very first time. Also, in this case, CFTCTx and CFADDRx usually contain invalid initialization values. The software can either solve this by setting BIM = 0 for the first time or setting CFADDRx to IFADDRx and CFTCTx to IFTCOUNT before the DCP is enabled.

NOTE: If

- the HTUEN bit is changed to 1 after the HTU was disabled HTUEN = 0
- the CPENA bit pair is 01 or 10 (during this HTUEN change)

then the corresponding BIM bit will decide if the corresponding buffer continues at its initial or current address. Cases E and F in Table 24-27 also apply for this situation. The software should use the procedures explained in the first note before setting HTUEN.

24.4.15 Request Lost Flag Register (HTU RLOSTFL)

Figure 24-28. Request Lost Flag Register (HTU RLOSTFL) [offset = 40h]

31	16
Reserved	
R-0	

15	0
CPRLFL	
R/W1CP-0	

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 in privilege mode to clear the bit; -*n* = value after reset

Table 24-28. Request Lost Flag Register (HTU RLOSTFL) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	CPRLFL		CP Request Lost Flags
		0	No request was lost. Writing a 0 has no effect.
		1	If bit (2*x) is set, a request was lost on CP A of DCP x. If bit (2*x+1) is set, a request was lost on CP B of DCP x. Reading from INTOFFx in case of a RLOST interrupt clears the corresponding flag. The state of the flag bit can be polled even if RLINTENA is cleared. • Reading CPRLFL will not clear the flags or • Reading from INTOFFx clears the corresponding flag. • Writing a 1 clears the corresponding flag.

24.4.16 Buffer Full Interrupt Flag Register (HTU BFINTFL)

Figure 24-29. Buffer Full Interrupt Flag Register (HTU BFINTFL) [offset = 44h]

31	16
Reserved	
R-0	

15	0
BFINTFL	
R/W1CP-0	

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 in privilege mode to clear the bit; -*n* = value after reset

Table 24-29. Buffer Full Interrupt Flag Register (HTU BFINTFL) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	BFINTFL		Buffer Full Interrupt Flags
		0	No buffer full condition is detected. Writing a 0 has no effect.
		1	If bit (2*x) is set, a buffer full condition on CP A of DCP x has been detected. If bit (2*x+1) is set, a buffer full condition on CP B of DCP x has been detected. The BFINTFL flag is set after the last frame finishes on the corresponding buffer regardless of whether the buffer is configured to one shot, circular or auto-switch mode. If BFINTFL is set in circular mode, then a circular overrun has occurred on the corresponding buffer. This can be used to indicate whether the buffer section after the frozen full address contains valid data or not. Reading from INTOFFx in case of a buffer-full interrupt clears the corresponding flag. The state of the flag bit can be polled even if the corresponding interrupt enable bit is cleared. • Reading BFINTFL will not clear the flags or • Reading INTOFFx will clear the corresponding flags or • Writing a 1 clears the corresponding flag.

24.4.17 BER Interrupt Flag Register (HTU BERINTFL)

A bus error interrupt results due to an address error or a timeout condition on the main memory access. A bus error will stop the frame transfer. Please see Section 24.2.3.

Figure 24-30. BER Interrupt Flag Register (HTU BERINTFL) [offset = 48h]

31	16
Reserved	
R-0	

15	0
BERINTFL	
R/W1CP-0	

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 in privilege mode to clear the bit; -*n* = value after reset

Table 24-30. BER Interrupt Flag Register (HTU BERINTFL) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	BERINTFL		Bus Error Interrupt Flags
		0	No bus error condition is detected. Writing a 0 has no effect.
		1	If bit (2*x) is set, then a BER interrupt is pending on CP A of DCP x. If bit (2*x+1) is set, then a BER interrupt is pending on CP B of DCP x. The state of the flag bit can be polled even if BERINTENA is cleared. • Reading BERINTFL will not clear the flags or • Reading from INTOFFx in case of a BER interrupt clears the corresponding flag or • Writing a 1 clears the corresponding flag.

24.4.18 Memory Protection 1 Start Address Register (HTU MP1S)

This register configures the start address of memory protection region 1.

Figure 24-31. Memory Protection 1 Start Address Register (HTU MP1S) [offset = 4Ch]

31	16
STARTADDRESS1	
R/WP-0	

15	2	1	0
STARTADDRESS1		0	0
R/WP-0			

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

Table 24-31. Memory Protection 1 Start Address Register (HTU MP1S) Field Descriptions

Bit	Field	Description
31-0	STARTADDRESS1	The start address defines at which main memory address the region begins. A memory protection error will be triggered, if the HTU accesses an address smaller than STARTADDRESS1 and the MPCS bit REG01ENA register is configured accordingly. The address is 32-bit aligned, so the 2 LSBs are not significant and will always read 0.

24.4.19 Memory Protection 1 End Address Register (HTU MP1E)

Figure 24-32. Memory Protection 1 End Address Register (HTU MP1E) [offset = 50h]

31	16
ENDADDRESS1	
R/WP-0	

15	2	1	0
ENDADDRESS1		0	0
R/WP-0			

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

Table 24-32. Memory Protection 1 End Address Register (HTU MP1E) Field Descriptions

Bit	Field	Description
31-0	ENDADDRESS1	The end address defines at which address the region ends. A memory protection error will be triggered, if the HTU accesses an address bigger than ENDADDRESS1 and the register bit REG01ENA is configured accordingly. The address is 32-bit aligned, so the 2 LSBs are not significant and will always read 0. The effective end address is rounded up to the nearest word end address, that is, 0x200 = 0x203.

24.4.20 Debug Control Register (HTU DCTRL)

This register allows to create watch points on access to a certain location. It is intended to help debug the application execution during program development.

Figure 24-33. Debug Control Register (HTU DCTRL) [offset = 54h]

31 28	27 24	23 17	16
Reserved	CPNUM	Reserved	HTUDBGS
R-0	R-0	R-0	R/W1CS-0

15 1	0
Reserved	DBREN
R-0	R/WS-0

LEGEND: R/W = Read/Write; R = Read only; W1CS = Write 1 in suspend mode to clear the bit; WS = Write in suspend mode only; -*n* = value after reset

Table 24-33. Debug Control Register (HTU DCTRL) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reads return 0. Writes have no effect.
27-24	CPNUM		CP Number. These bit fields indicate the CP that should cause the watch point to match.
		0	CP A of DCP0
		1h	CP B of DCP0
		2h	CP A of DCP1
		3h	CP B of DCP1
		4h	CP A of DCP2
		5h	CP B of DCP2
		6h	CP A of DCP3
		7h	CP B of DCP3
		8h	CP A of DCP4
		9h	CP B of DCP4
		Ah	CP A of DCP5
		Bh	CP B of DCP5
		Ch	CP A of DCP6
		Dh	CP B of DCP6
		Eh	CP A of DCP7
		Fh	CP B of DCP7
23-17	Reserved	0	Reads return 0. Writes have no effect.
16	HTUDBGS		HTU Debug Status. When the main memory address is equal to the unique address defined by WPR, or lies in the specified range resulting from WMR, then the HTUDBGS is set. If in addition DBREN is set, then the application code execution will be stopped. A 1 must be written to this bit in order to clear it and to release the CPU from debug halting state.
		0	Read: No watch point condition was detected. Write: No effect.
		1	Read: A watch point condition was detected. Write: Clears the bit.
15-1	Reserved	0	Reads return 0. Writes have no effect.
0	DBREN		Debug Request Enable If a watch point matches and DBREN is set, then the application code execution will be stopped. This bit can only be set or cleared when in debug mode. This bit and all other bits of the DCTRL, WPR and WMR registers are reset by the test reset (nTRST) but not by the normal device reset.

24.4.21 Watch Point Register (HTU WPR)

This register defines the main memory address of the watch point.

Figure 24-34. Watch Point Register (HTU WPR) [offset = 58h]

31	16
WP	
R/WS-0	

15	0
WP	
R/WS-0	

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

Table 24-34. Watch Point Register (HTU WPR) Field Descriptions

Bit	Field	Description
31-0	WP	Watch Point Register A 32-bit address can be programmed into this register as a watch point. The WPR register is used along with the Watch Mask Register (WMR). When the main memory address is equal to the unique address defined by WPR, or lies in the specified range resulting from WMR, then the HTUDBGS is set. If in addition DBREN is set, then the application code execution is stopped. This register can only be programmed during debug mode. This register and all other bits of the DCTRL and WMR registers are reset by the test reset (nTRST) but not by the normal device reset.

24.4.22 Watch Mask Register (HTU WMR)

This register defines a mask of the main memory address of the watch point. It can be used to define a memory range in conjunction with the WPR register.

Figure 24-35. Watch Mask Register (HTU WMR) [offset = 5Ch]

31	16
WM	
R/WS-0	

15	0
WM	
R/WS-0	

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

Table 24-35. Watch Mask Register (HTU WMR) Field Descriptions

Bit	Field	Description
31-0	WM	Watch Mask Register Setting a bit in the WMR register to 1 has the effect of masking the corresponding bit in of the main memory address, so that this bit is ignored for the address comparison. This register can only be programmed during debug mode. This register and all other bits of the DCTRL and WPR registers are reset by the test reset (nTRST) but not by the normal device reset.

24.4.23 Module Identification Register (HTU ID)

This register is for TI internal purposes and allows to keep track of the HTU module version on different devices.

Figure 24-36. Module Identification Register (HTU ID) [offset = 60h]

31 … 24	23 … 16
Reserved	CLASS
R-0	R - Module Class Number

15 … 8	7 … 0
TYPE	REV
R - Class Subtype Number	R - Module Revision Number

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

Table 24-36. Module Identification Register (HTU ID) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23-16	CLASS		Module Class This field defines the module class number as read-only constant value for the HTU module. Writes have no effect.
15-8	TYPE		Subtype within a Class This field defines the subtype within a class as read-only constant value for the HTU module. Writes have no effect.
7-0	REV		Module Revision Number This field defines the module revision number as read-only constant value for the HTU module. Writes have no effect.

24.4.24 Parity Control Register (HTU PCR)

Figure 24-37. Parity Control Register (HTU PCR) [offset = 64h]

31–17	16
Reserved	COPE
R-0	R/WP-0

15–9	8	7–4	3–0
Reserved	TEST	Reserved	PARITY_ENA
R-0	R/WP-0	R-0	R/WP-5h

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

Table 24-37. Parity Control Register (HTU PCR) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reads return 0. Writes have no effect.
16	COPE		Continue on Parity Error
		0	The HTU performs parity checks every time it reads the RAM section of DCP x (with x = 0, 1,... or 7), before the next frame (of DCP x) is started. If a parity error is detected during this read access and if the parity check is enabled, then the frame will not be started and DCP x will be automatically disabled in the CPENA register. If a master different than the HTU (for example, CPU) reads the RAM section of DCP x and a parity error is detected during this read access, while the parity check is enabled, then the DCP x will automatically be disabled in the CPENA register. If a frame is active on DCP x during this read access, then in addition the element counter of DCP x is cleared and all new element transfers on DCP x are stopped and the active busy bit of DCP x is cleared.
		1	The difference to COPE = 0 is, that the data transfer on a active DCP continues after a parity error was detected on this DCP. So, neither the DCP with the parity error will be disabled nor the frame will be stopped.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	TEST		Test. When this bit is set, the parity bits are mapped into the peripheral RAM frame to make them accessible by the CPU.
		0	Parity bits are not memory-mapped.
		1	Parity bits are memory-mapped.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	PARITY_ENA		Enable/Disable Parity Checking. This bit field enables or disables the parity check on read operations and the parity calculation on write operations. If parity checking is enabled and a parity error is detected, then the PEFT flag is set, PAOFF is captured if it is not currently frozen and an interrupt is generated if it is enabled.
		5h	Parity check is disabled.
		All Others	Parity check is enabled. **Note:** It is recommended to write Ah to enable error detection, to guard against single bit changes from flipping PARITY_ENA to a disable state.

24.4.25 Parity Address Register (HTU PAR)

Figure 24-38. Parity Address Register (HTU PAR) [offset = 68h]

31	17	16
Reserved		PEFT
R-0		R/W1CP-0

15	9	8	0
Reserved		PAOFF	
R-0		R-X	

LEGEND: R/W = Read/Write; R = Read only; W1CP = Write 1 in privilege mode to clear the bit; -*n* = value after reset; X = undefined

Table 24-38. Parity Address Register (HTU PAR) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return 0. Writes have no effect.
16	PEFT		Parity Error Fault Flag. This bit is set, when the HTU detects a parity error and parity checking is enabled.
		0	No fault is detected.
		1	Fault is detected. **Note:** Once PEFT is set, a read access to the lower 16 bits or to the complete 32-bit HTUPAR register will clear the PEFT flag in non-debug mode. Another possibility to clear PEFT is to write a 1 to the PEFT bit.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8-0	PAOFF		Parity Error Address Offset. This bit field holds the address of the first parity error, which is detected in the DCP RAM. PAOFF provides the offset address of the erroneous byte counted from the beginning of the DCP memory. This error address is frozen from being updated until a read access to the lower 16 bits or to the complete 32-bit HTUPAR register happens. During debug mode, this address is frozen even when read. **Note:** The Parity Error Address bits will not be reset, neither by PORRST nor by any other reset source.

24.4.26 Memory Protection Control and Status Register (HTU MPCS)

Figure 24-39. Memory Protection Control and Status Register (HTU MPCS) [offset = 70h]

31-28	27-24
Reserved	CPNUM0
R-0	R-0

23-18	17	16
Reserved	MPEFT1	MPEFT0
R-0	R/W1CP-0	R/W1CP-0

15-12	11-8
Reserved	CPNUM1
R-0	R-0

7-6	5	4	3	2	1	0
Reserved	INT ENA01	ACCR01	REG01ENA	INT ENA0	ACCR0	REG0ENA
R-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; W1CP = Write 1 in privilege mode to clear the bit; -*n* = value after reset

Table 24-39. Memory Protection Control and Status Register (HTU MPCS) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reads return 0. Writes have no effect.
27-24	CPNUM0		Control Packet Number for single memory protection region configuration. CPNUM0 holds the number of the CP, which has caused the first memory protection error when only one memory protection region is used. This number is not updated for multiple access violations until it is read by the CPU. During debug mode, CPNUM0 is frozen even when read.
		0	CP A of DCP0
		1h	CP B of DCP0
		2h	CP A of DCP1
		3h	CP B of DCP1
		4h	CP A of DCP2
		5h	CP B of DCP2
		6h	CP A of DCP3
		7h	CP B of DCP3
		8h	CP A of DCP4
		9h	CP B of DCP4
		Ah	CP A of DCP5
		Bh	CP B of DCP5
		Ch	CP A of DCP6
		Dh	CP B of DCP6
		Eh	CP A of DCP7
		Fh	CP B of DCP7
23-18	Reserved	0	Reads return 0. Writes have no effect.
17	MPEFT1		Memory Protection Error Fault Flag 1. This bit is set, when the HTU performs an access outside the region defined by the MP0S and MP0E and the MP1S and MP1E registers, when the access violates the rights defined by ACCR01, and when the REG01ENA bit is set.
		0	No fault detected. Writing a 0 has no effect.
		1	Fault detected. Writing a 1 will clear the bit.
16	MPEFT0		Memory Protection Error Fault Flag 0. This bit is set, when the HTU performs an access outside the region defined by the MP0S and MP0E registers, when the access violates the rights defined by ACCR, and when the REG0ENA bit is set.
		0	No fault detected. Writing a 0 has no effect.
		1	Fault detected. Writing a 1 will clear the bit.

Table 24-39. Memory Protection Control and Status Register (HTU MPCS) Field Descriptions (continued)

Bit	Field	Value	Description
15-12	Reserved	0	Reads return 0. Writes have no effect.
11-8	CPNUM1		Control Packet Number for single memory protection region configuration. CPNUM1 holds the number of the CP, which has caused the first memory protection error when only one memory protection region is used. This number is not updated for multiple access violations until it is read by the CPU. During debug mode, CPNUM1 is frozen even when read.
		0	CP A of DCP0
		1h	CP B of DCP0
		2h	CP A of DCP1
		3h	CP B of DCP1
		4h	CP A of DCP2
		5h	CP B of DCP2
		6h	CP A of DCP3
		7h	CP B of DCP3
		8h	CP A of DCP4
		9h	CP B of DCP4
		Ah	CP A of DCP5
		Bh	CP B of DCP5
		Ch	CP A of DCP6
		Dh	CP B of DCP6
		Eh	CP A of DCP7
		Fh	CP B of DCP7
7-6	Reserved	0	Reads return 0. Writes have no effect.
5	INTENA01		Interrupt Enable 01. This bit needs to be set when working with two memory-mapped regions and a error should be generated to the ESM module on an access violation.
		0	Error signaling is disabled.
		1	Error signaling is enabled.
4	ACCR01		Access Rights 01. This bit defines the access rights for the HTU for accesses outside the region defined by the MP0S and MP0E and the MP1S and MP1E registers.
		0	HTU read access is allowed but write access will be signaled.
		1	Any access performed by the HTU is forbidden and will be signaled.
3	REG01ENA		Region Enable 01. This bit needs to be set when working with two memory-mapped regions. **REG0ENA must be cleared to 0** if this bit is set to a 1. Memory region 0 must be less than memory region 1.
		0	The protection outside the memory region defined by the MP0S and MP0E and the MP1S and MP1E registers is not enabled. This means the HTU can access any implemented memory space. REG0ENA could still enabled to give protection outside the MP0S:MP0E region.
		1	The protection outside the memory region defined by the MP0S and MP0E and the MP1S and MP1E registers is enabled. This means the HTU can perform any access within the regions, but if it attempts to perform a forbidden access outside of both of the regions (according to the ACCR01 configuration), the access is signaled by the MPEFT1 flag. The number of the CP, which has caused the memory protection error, is captured to CPNUM1 if it is not currently frozen and an error is generated if it is enabled.
2	INTENA0		Interrupt Enable 0. This bit needs to be set when working with one memory-mapped region and a error should be generated to the ESM module on an access violation.
		0	Error signaling is disabled.
		1	Error signaling is enabled.
1	ACCR		Access Rights 0. This bit defines the access rights for the HTU for accesses outside the region defined by the MP0S and MP0E registers for a single memory protection region configuration.
		0	HTU read access is allowed but write access will be signaled.
		1	Any access performed by the HTU is forbidden and will be signaled.

Table 24-39. Memory Protection Control and Status Register (HTU MPCS) Field Descriptions (continued)

Bit	Field	Value	Description
0	REG0ENA		Region Enable 0
		0	The protection outside the memory region defined by the MP0S and MP0E registers is not enabled. This means the HTU can access any implemented memory space.
		1	The protection outside the memory region defined by the MP0S and MP0E registers is enabled. This means the HTU can perform any access within the region, but if it attempts to perform a forbidden access outside the region (according to the ACCR configuration), the access is signaled by the MPEFT0 flag, the number of the CP, which has caused the memory protection error, is captured to CPNUM0 if it is not currently frozen and an error is generated if it is enabled.

24.4.27 Memory Protection Start Address Register 0 (HTU MP0S)

This register configures the start address of memory protection region 0

Figure 24-40. Memory Protection Start Address Register 0 (HTU MP0S) [offset = 74h]

31	16
STARTADDRESS0	
R/WP-0	

15	2	1	0
STARTADDRESS0		0	0
R/WP-0			

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

Table 24-40. Memory Protection 0 Start Address Register (HTU MP0S) Field Descriptions

Bit	Field	Description
31-0	STARTADDRESS0	The start address defines at which main memory address the region begins. A memory protection error will be triggered, if the HTU accesses an address smaller than STARTADDRESS0 and the MPCS register is configured accordingly. The address is 32-bit aligned, so the 2 LSBs are not significant and will always read 0.

24.4.28 Memory Protection End Address Register (HTU MP0E)

Figure 24-41. Memory Protection End Address Register (HTU MP0E) [offset = 78h]

31	16
ENDADDRESS0	
R/WP-0	

15	2	1	0
ENDADDRESS0		0	0
R/WP-0			

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

Table 24-41. Memory Protection End Address Register (HTU MP0E) Field Descriptions

Bit	Field	Description
31-0	ENDADDRESS0	The end address defines at which address the region ends. A memory protection error will be triggered, if the HTU accesses an address bigger than ENDADDRESS0 and the register bit MPCS register is configured accordingly. The address is 32-bit aligned, so the 2 LSBs are not significant and will always read 0. The effective end address is rounded up to the nearest word end address, that is, 0x200 = 0x203.

24.5 Double Control Packet Configuration Memory

All bits marked "reserved' are implemented in RAM and will be initialized to unknown values after power on. Reserved locations can be written and read but should be written with 0 to ensure future compatibility. The HTU RAM can be cleared with the system RAM initialization function.

Table 24-42 provides a summary of the memory configuration. There are eight sets of DCP registers and eight sets of CF registers. The base address for the DCP registers is FF4E 0000h for HTU1 and FF4C 0000h for HTU2.

Table 24-42. Double Control Packet Memory Map

Offset	Acronym	Register Description	Section
00h	HTU DCP0 IFADDRA	Initial Full Address A Register	Section 24.5.1
04h	HTU DCP0 IFADDRB	Initial Full Address B Register	Section 24.5.2
08h	HTU DCP0 IHADDRCT	Initial N2HET Address and Control Register	Section 24.5.3
0Ch	HTU DCP0 ITCOUNT	Initial Transfer Count Register	Section 24.5.4
10h	HTU DCP1 IFADDRA	Initial Full Address A Register	Section 24.5.1
14h	HTU DCP1 IFADDRB	Initial Full Address B Register	Section 24.5.2
18h	HTU DCP1 IHADDRCT	Initial N2HET Address and Control Register	Section 24.5.3
1Ch	HTU DCP1 ITCOUNT	Initial Transfer Count Register	Section 24.5.4
:	:	:	
70h	HTU DCP7 IFADDRA	Initial Full Address A Register	Section 24.5.1
74h	HTU DCP7 IFADDRB	Initial Full Address B Register	Section 24.5.2
78h	HTU DCP7 IHADDRCT	Initial N2HET Address and Control Register	Section 24.5.3
7Ch	HTU DCP7 ITCOUNT	Initial Transfer Count Register	Section 24.5.4
100h	HTU CDCP0 CFADDRA	Current Full Address A Register	Section 24.5.5
104h	HTU CDCP0 CFADDRB	Current Full Address B Register	Section 24.5.6
108h	HTU CDCP0 CFCOUNT	Current Frame Count Register	Section 24.5.7
110h	HTU CDCP1 CFADDRA	Current Full Address A Register	Section 24.5.5
114h	HTU CDCP1 CFADDRB	Current Full Address B Register	Section 24.5.6
118h	HTU CDCP1 CFCOUNT	Current Frame Count Register	Section 24.5.7
:	:	:	
170h	HTU CDCP7 CFADDRA	Current Full Address A Register	Section 24.5.5
174h	HTU CDCP7 CFADDRB	Current Full Address B Register	Section 24.5.6
178h	HTU CDCP7 CFCOUNT	Current Frame Count Register	Section 24.5.7

24.5.1 Initial Full Address A Register (HTU IFADDRA)

Figure 24-42. Initial Full Address A Register (HTU IFADDRA)

31	16
IFADDRA	
R/WP-X	

15	0
IFADDRA	
R/WP-X	

LEGEND: R/W = Read/Write; WP = Write in privilege mode only; -*n* = value after reset; X = Unknown

Table 24-43. Initial Full Address A Register (HTU IFADDRA) Field Descriptions

Bit	Field	Description
31-0	IFADDRA	Initial Address of Buffer A in main memory. Initial (byte) address of buffer A placed in the main memory address range. Bits 0 and 1 are ignored by the logic, due to 32-bit alignment.

24.5.2 Initial Full Address B Register (HTU IFADDRB)

Figure 24-43. Initial Full Address B Register (HTU IFADDRB)

31	16
IFADDRB	
R/WP-X	

15	0
IFADDRB	
R/WP-X	

LEGEND: R/W = Read/Write; WP = Write in privilege mode only; -*n* = value after reset; X = Unknown

Table 24-44. Initial Full Address B Register (HTU IFADDRB) Field Descriptions

Bit	Field	Description
31-0	IFADDRB	Initial Address of Buffer B in main memory. Initial (byte) address of buffer B placed in the main memory address range. Bits 0 and 1 are ignored by the logic, due to 32-bit alignment.

24.5.3 Initial N2HET Address and Control Register (HTU IHADDRCT)

Figure 24-44. Initial N2HET Address and Control Register (HTU IHADDRCT)

31-24
Reserved
R-0

23	22	21	20	19-18	17-16
DIR	SIZE	ADDMH	ADDMF	TMBA	TMBB
R/WP-X	R/WP-X	R/WP-X	R/WP-X	R/WP-X	R/WP-X

15-13	12-2	1-0
Reserved	IHADDR	Reserved
R-0	R/WP-X	R-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset; X = Unknown

Table 24-45. Initial N2HET Address and Control Register (HTU IHADDRCT) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23	DIR		Direction of Transfer
		0	N2HET address is read and main memory address is written.
		1	Main memory address is read and N2HET address is written.
22	SIZE		Size of Transferred Data
		0	32-bit transfer
		1	64-bit transfer
			64-bit transfer examples: If the N2HET address points to the N2HET instruction Control Field (CF), then the CF and Data Field (DF) will be transferred. If the N2HET address points to the Program Field (PF), then the PF and CF will be transferred.
21	ADDMH		Addressing Mode N2HET Address. This bit determines the N2HET address index from one to the next element of a frame.
		0	Increment by 16 bytes.
			Examples:
			If the initial N2HET address points to data field of instruction (n). Then the N2HET fields to be transferred by the elements of a frame are: data field of instruction (n), data field of instruction (n+1), data field of instruction (n+2) and so on. If the initial N2HET address points to control field of instruction (n), then the N2HET fields to be transferred by the elements of a frame are: control field of instruction (n), control field of instruction (n+1), control field of instruction (n+2) and so on.
		1	Increment by 8 bytes.
			This mode is intended to be used together with the 64-bit transfer size to load short N2HET instruction blocks into the N2HET RAM. So the sequence of transferred 64-bit elements could be: [PF and CF of instruction (n)], [DF and RF of instruction (n)], [PF and CF of instruction (n+1)], [DF and RF of instruction (n+1)] and so on.
20	ADDMF		Addressing Mode Main Memory Address
		0	Post-increment
			Note: When post-increment is selected the HTU will automatically increment by 4 bytes for a 32-bit data size and by 8 bytes for a 64-bit data size.
		1	Constant
19-18	TMBA		Transfer Mode for Buffer A
		0	One-Shot buffer mode
		1h	Circular buffer mode
		2h-3h	Auto Switch mode
17-16	TMBB		Transfer Mode for Buffer B
		0	One-Shot buffer mode
		1h	Circular buffer mode
		2h-3h	Auto Switch mode

Table 24-45. Initial N2HET Address and Control Register (HTU IHADDRCT) Field Descriptions (continued)

Bit	Field	Value	Description
15-13	Reserved	0	Reads return 0. Writes have no effect.
12-2	IHADDR		Initial N2HET Address The initial N2HET Address points to the N2HET field, which is the first element of the frame. The N2HET address (bits 12:2) increments by 1 for each 32-bit N2HET field and starts with 0 at the first 32-bit field in the N2HET RAM. **Note:** When the HTU addresses the N2HET RAM it uses only the number of address bits required for the actual N2HET RAM size. If the N2HET address exceeds the actual N2HET RAM size, the unused MSB bits of the address will be ignored and the address rolls over to the start of the N2HET RAM.
1-0	Reserved	0	Reads return 0. Writes have no effect.

24.5.4 Initial Transfer Count Register (HTU ITCOUNT)

Figure 24-45. Initial Transfer Count Register (HTU ITCOUNT)

31 — 21	20 — 16
Reserved	IETCOUNT
R-0	R/WP-X

15 — 8	7 — 0
Reserved	IFTCOUNT
R-0	R/WP-X

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset; X = Unknown

Table 24-46. Initial Transfer Count Register (HTU ITCOUNT) Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	0	Reads return 0. Writes have no effect.
20-16	IETCOUNT		Initial Element Transfer Count Defines the number of element transfers.
15-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	IFTCOUNT		Initial Frame Transfer Count Defines the number of frame transfers.

24.5.5 Current Full Address A Register (HTU CFADDRA)

Figure 24-46. Current Full Address A Register (HTU CFADDRA)

31	16
CFADDRA	
R/WP-X	

15	0
CFADDRA	
R/WP-X	

LEGEND: R/W = Read/Write; WP = Write in privilege mode only; -*n* = value after reset; X = Unknown

Table 24-47. Current Full Address A Register (HTU CFADDRA) Field Descriptions

Bit	Field	Description
31-0	CFADDRA	Current (byte) Address of Buffer A The current main memory address register is updated at the end of each frame. Therefore it points to the start address of the frame, which is the next to transfer, if currently no frame is transferred on this DCP. For an ongoing frame transfer, it points to the start address of this frame. After the last element of a buffer was transferred it will point to the buffer end address plus 0x4. The main purpose of the current full address registers for buffer A and buffer B (see next section) is to enable the software to find out the recently transferred element in the frozen buffer while the address of the active buffer increments. **Note:** A frame can be automatically stopped if any of the events listed in Conditions for Frame Transfer Interruption happens. If a frame is stopped before it could complete, then the current full address register is not updated and it will point to the start of the bad frame after the DCP was automatically disabled.

To transfer the first frame of buffer x, the information in the corresponding initial DCP RAM (IFADDRx, IHADDRCT, ITCOUNT) is loaded to an internal state machine but not to the current DCP RAM (CFADDRx, CFTCTx).

This is valid for all of the following modes:

- Buffer x has reached it's end in circular mode and rolls back to its start address.
- CP x is enabled by a CPENA access (and corresponding BIM bit is 0).
- A CPENA access or auto-switch mode causes a switch from CP y to CP x.

This means after starting the transfer to/from buffer x, CFADDRx and CFTCTx is not updated before the end of the first frame. So before the software switches from CP y to CP x using a write access to the CPENA register, it needs to initialize CFADDRx, CFTCTx. This allows the software to find out if the next request on CP x after the switching to CP x was delayed or never occurring.

24.5.6 Current Full Address B Register (HTU CFADDRB)

Figure 24-47. Current Full Address B Register (HTU CFADDRB)

31	16
CFADDRB	
R/WP-X	

15	0
CFADDRB	
R/WP-X	

LEGEND: R/W = Read/Write; WP = Write in privilege mode only; -*n* = value after reset; X = Unknown

Table 24-48. Current Full Address B Register (HTU CFADDRB) Field Descriptions

Bit	Field	Description
31-0	CFADDRB	Current (byte) Address of Buffer B The current main memory address register is updated at the end of each frame. Therefore it points to the start address of the frame, which is the next to transfer, if currently no frame is transferred on this DCP. If currently a frame is transferred, then it points to the start address of this frame. After the last element of a buffer was transferred it will point to the buffer end address plus 0x4. The main purpose of the current full address registers for buffer A and buffer B (see next section) is to enable the software to find out the recently transferred element in the frozen buffer while the address of the active buffer increments. **Note:** A frame can be automatically stopped if any of the events listed in Conditions for Frame Transfer Interruption happens. If a frame is stopped before it could complete, then the current full address register is not updated and it will point to the start of the bad frame after the DCP was automatically disabled.

To transfer the first frame of buffer x, the information in the corresponding initial DCP RAM (IFADDRx, IHADDRCT, ITCOUNT) is loaded to an internal state machine but not to the current DCP RAM (CFADDRx, CFTCTx).

This is valid for all of the following modes:

- Buffer x has reached it's end in circular mode and rolls back to its start address.
- CP x is enabled by a CPENA access (and corresponding BIM bit is 0).
- A CPENA access or auto-switch mode causes a switch from CP y to CP x.

This means after starting the transfer to/from buffer x, CFADDRx and CFTCTx is not updated before the end of the first frame. So before the software switches from CP y to CP x using a write access to the CPENA register, it needs to initialize CFADDRx, CFTCTx. This allows the software to find out if the next request on CP x after the switching to CP x was delayed or never occurring.

24.5.7 Current Frame Count Register (HTU CFCOUNT)

The current frame count register enables the software to find out the recent frame in the buffer while the counter of the active buffer decrements.

Figure 24-48. Current Frame Count Register (HTU CFCOUNT)

31 24	23 16
Reserved	CFTCTA
R-0	R/WP-X

15 8	7 0
Reserved	CFTCTB
R-0	R/WP-X

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privilege mode only; -*n* = value after reset; X = Unknown

Table 24-49. Current Frame Count Register (HTU CFCOUNT) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23-16	CFTCTA		Current Frame Transfer Count for CP A. It is updated at the end of each frame.
15-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	CFTCTB		Current Frame Transfer Count for CP B. It is updated at the end of each frame.

24.6 Examples

24.6.1 Application Examples for Setting the Transfer Modes of CP A and B of a DCP

Table 24-50. Application Examples for Setting the Transfer Modes of CP A and B of a DCP

CP A	CP B	
One shot	Not used	Buffer A can be used as a "one shot" buffer. A buffer full interrupt enabled for CP A can signal reaching the end of the buffer.
Auto switch	One shot	Can double the buffer size for a "one shot" buffer. A buffer full interrupt enabled for CP B can signal reaching the end of the buffer.
Circular	Circular	The CPU can switch the buffers at arbitrary times. It will fill or read the frozen buffer during the other buffer is filled or read by the HTU. Interrupts are not required for this case.
Auto switch	Auto switch	Buffer full interrupts (enabled for CP A and B) signal when the end of a buffer is reached. After one buffer is completed the according CPU interrupt routine will read or refill this buffer. At the same time the other buffer is read or filled by the HTU. Here the time when the buffer must be read is determined by the time of the interrupt (determined by the frequency of the N2HET transfer requests).

24.6.2 Software Example Sequence Assuming Circular Mode for Both CP A and B

The example assumes the N2HET address to be read and the main memory address to be written.

I1 CPU initializes initial DCP: IFADDRA, IFADDRB, IHADDRCT, ITCOUNT

I2 CPU clears current DCP: CFADDRA, CFADDRB, CFTCTA, CFTCTB

I3 CPU clears BFINTFL flag of CP A and B

I4 Enable CP A with the CPENA register. Now the HTU fills buffer A

After some time the CPU intends to read buffer A:

A1 CPU enables CP B and disables CP A by writing to the CPENA register. After this switch, the HTU fills buffer B. Filling buffer B starts with its initial full address and initial frame counter.

A2 CPU waits for CP A busy bit equals 0

A3 Optional: CPU verifies that the CP A request lost flag is not set. The bus error flag of CP A could also be checked.

A4 CPU reads the frozen CFTCTA, which indicates the fill level in the buffer

A5 CPU sets current CP A (CFTCTA and/or CFADDRA) to 0. This allows to find out if any request has happened during the next time buffer A is active.

A6 CPU reads BFINTFL flag of buffer A

A7 CPU clears the BFINTFL flag of buffer A. This is an initialization for the next time buffer A is used.

A8 CPU reads valid values of frozen buffer A. After reading the CPU does not need to clear the frozen buffer A.

After some time the CPU intends to read buffer B:

B1 CPU enables CP A and disables CP B by writing to the CPENA register. After this switch, the HTU fills buffer A. Filling buffer A starts with its initial full address and initial frame counter.

B2 CPU waits for CP B busy bit equals 0

B3 Optional: CPU verifies that the CP B request lost flag is not set. The bus error flag of CP B could also be checked.

B4 CPU reads the frozen CFTCTB, which indicates the fill level in the buffer

B5 CPU sets current CP B (CFTCTB and/or CFADDRB) to 0. This allows to find out if any request has happened during the next time buffer B is active.

B6 CPU reads BFINTFL flag of buffer B

B7 CPU clears the BFINTFL flag of buffer B. This is an initialization for the next time buffer B is used.

B8 CPU reads valid values of frozen buffer B. After reading the CPU does not need to clear the frozen buffer B.

After some time the CPU intends to read buffer A:

A1) ... see above...

NOTE: The buffer full interrupt doesn't need to be enabled. The BFINTFL flag is used to indicate a circular overrun of the buffer. If the BFINTFL flag is set, also the buffer section after the frozen full address could be read.

Steps A3 and B3 in the example sequence above imply that request lost interrupts are disabled. The example below assumes that request lost interrupts are enabled.

Request lost detection with interrupt enabled.

24.6.3 Example of an Interrupt Dispatch Flow for a Request Lost Interrupt

- A request lost occurs and the interrupt routine starts.
- Reading INTOFFx.INTYPEx shows that RLOSTFL is the interrupt source.
- Reading INTOFFx.CPOFFx = Ah shows that DCP 5 / CP A has caused the RLOSTFL interrupt. The hardware automatically clears bit (2·5+0) in RLOSTFL.
- Reading RLOSTFL= 84h shows that also another request lost event happened on DCP 1 / CP A [bit (2·1+0)] and on DCP 3 / CP B [bit (2·3+1)] at the same time or after the request lost occurred on DCP 5 / CP A.
- Writing back 84h to RLOSTFL clears bits 2 and 7 and the according pending interrupts.

Chapter 25

SPNU514C – March 2018

General-Purpose Input/Output (GIO) Module

This chapter describes the general-purpose input/output (GIO) module. The GIO module provides the family of devices with input/output (I/O) capability. The I/O pins are bidirectional and bit-programmable. The GIO module also supports external interrupt capability.

Topic **Page**

25.1 Overview 1302
25.2 Quick Start Guide 1303
25.3 Functional Description of GIO Module 1305
25.4 Device Modes of Operation 1308
25.5 GIO Control Registers 1309
25.6 I/O Control Summary 1327

25.1 Overview

The GIO module offers general-purpose input and output capability. It supports up to eight 8-bit ports for a total of up to 64 GIO terminals. Each of these 64 terminals can be independently configured as input or output and configured as required by the application. The GIO module also supports generation of interrupts whenever a rising edge or falling edge or any toggle is detected on up to 32 of these GIO terminals. Refer to the device datasheet for identifying the number of GIO ports supported and the GIO terminals capable of generating an interrupt.

The main features of the GIO module are summarized as follows:

- Allows each GIO terminal to be configured for general-purpose input or output functions
- Supports programmable pull directions on each input GIO terminal
- Supports GIO output in push/pull or open-drain modes
- Allows up to 32 GIO terminals to be used for generating interrupt requests

25.2 Quick Start Guide

The GIO module comprises two separate components: an input/output (I/O) block and an interrupt generation block. Figure 25-1 and Figure 25-2 show what you should do after reset to configure the GIO module as I/O or for generating interrupts.

In GIO interrupt service routine, you shall read the GIO offset register (GIOOFF1 or GIOOFF2, depending on high-/low-level interrupt) to clear the flag and find the pending interrupt GIO channel.

Figure 25-1. I/O Function Quick Start Flow Chart

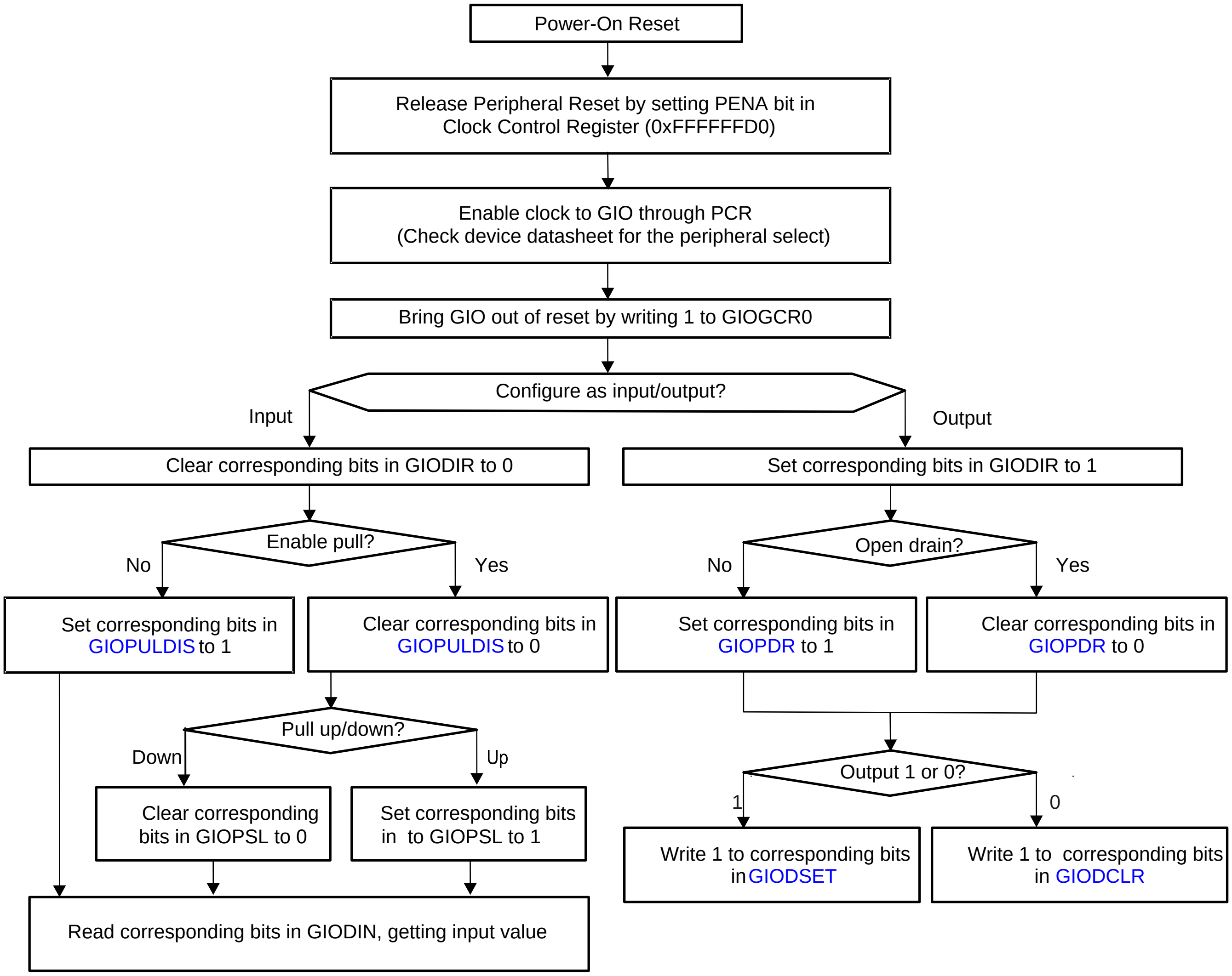
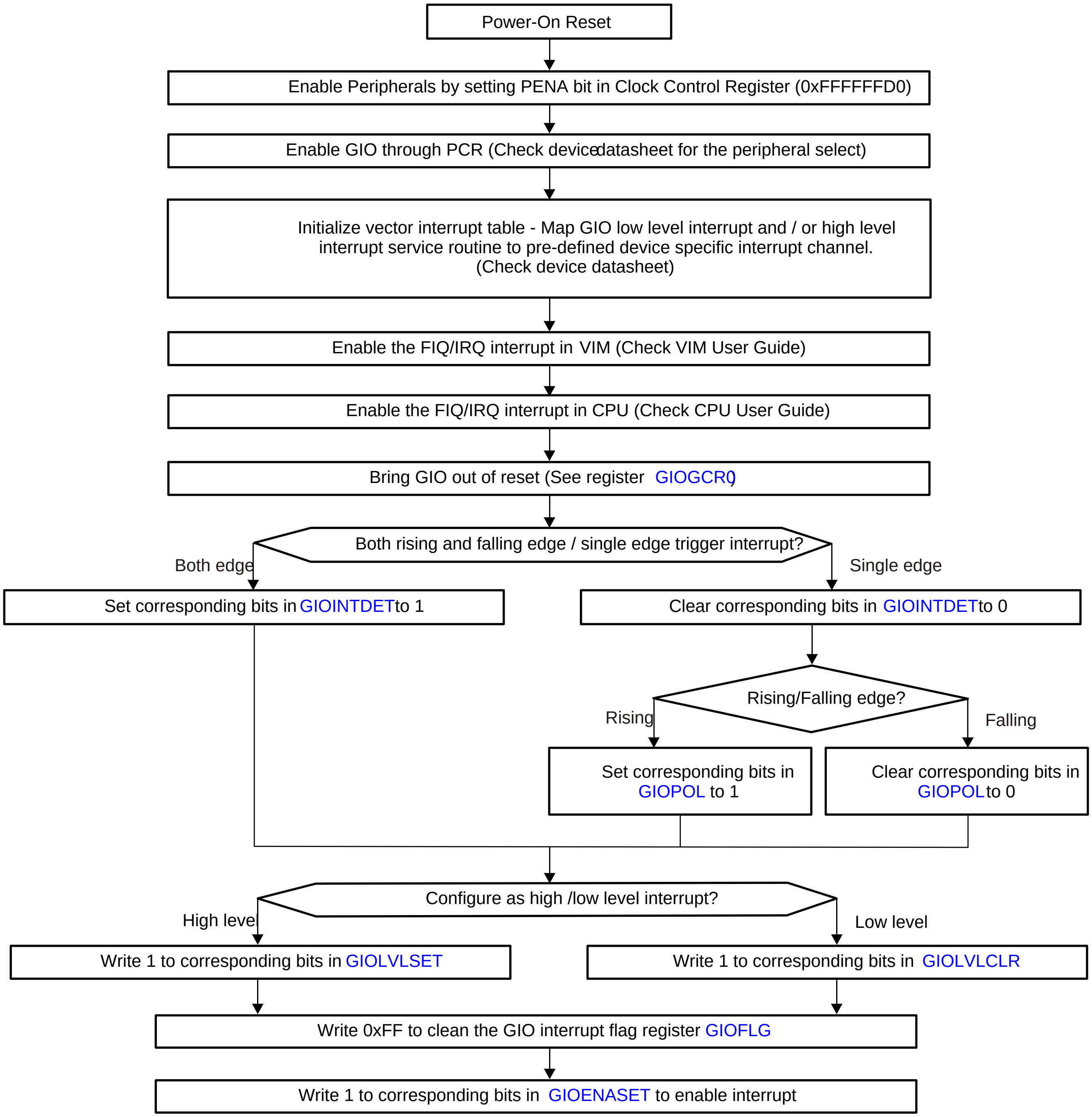

Power-On Reset
Release Peripheral Reset by setting PENA bit in Clock Control Register (0xFFFFFFD0)
Enable clock to GIO through PCR (Check device datasheet for the peripheral select)
Bring GIO out of reset by writing 1 to GIOGCR0
Configure as input/output?
Input
Output
Clear corresponding bits in GIODIR to 0
Set corresponding bits in GIODIR to 1
Enable pull?
No
Yes
Open drain?
No
Yes
Set corresponding bits in GIOPULDIS to 1
Clear corresponding bits in GIOPULDIS to 0
Set corresponding bits in GIOPDR to 1
Clear corresponding bits in GIOPDR to 0
Pull up/down?
Down
Up
Output 1 or 0?
1
0
Clear corresponding bits in GIOPSL to 0
Set corresponding bits in to GIOPSL to 1
Write 1 to corresponding bits in GIODSET
Write 1 to corresponding bits in GIODCLR
Read corresponding bits in GIODIN, getting input value

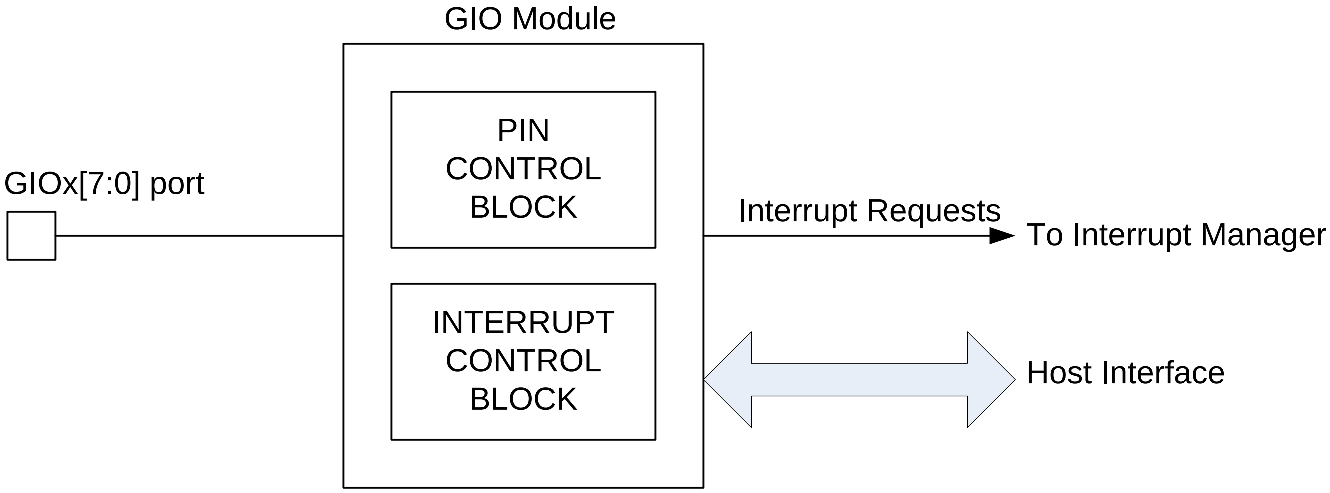
Figure 25-2. Interrupt Generation Function Quick Start Flow Chart


Power-On Reset
Enable Peripherals by setting PENA bit in Clock Control Register (0xFFFFFFD0)
Enable GIO through PCR (Check devicedatasheet for the peripheral select)
Initialize vector interrupt table - Map GIO low level interrupt and / or high level interrupt service routine to pre-defined device specific interrupt channel. (Check device datasheet)
Enable the FIQ/IRQ interrupt in VIM (Check VIM User Guide)
Enable the FIQ/IRQ interrupt in CPU (Check CPU User Guide)
Bring GIO out of reset (See register GIOGCR0)
Both rising and falling edge / single edge trigger interrupt?
Both edge
Single edge
Set corresponding bits in GIOINTDETto 1
Clear corresponding bits in GIOINTDETto 0
Rising/Falling edge?
Rising
Falling
Set corresponding bits in GIOPOL to 1
Clear corresponding bits in GIOPOLto 0
Configure as high /low level interrupt?
High level
Low level
Write 1 to corresponding bits in GIOLVLSET
Write 1 to corresponding bits in GIOLVLCLR
Write 0xFF to clean the GIO interrupt flag register GIOFLG
Write 1 to corresponding bits in GIOENASET to enable interrupt

25.3 Functional Description of GIO Module

As shown in Figure 25-3, the GIO module comprises of two separate components: an input/output (I/O) block and an interrupt block.

Figure 25-3. GIO Module Diagram



25.3.1 I/O Functions

The I/O block allows each GIO terminal to be configured for use as a general-purpose input or output in the application. The GIO module supports multiple registers to control the various aspects of the input and output functions. These are described as follows.

- Data direction (GIODIR)

  Configures GIO terminal(s) as input (default) or output through the GIODIRx registers.

- Data input (GIODIN)

  Reflects the logic level on GIO terminals in the GIODINx registers. A high voltage (V_{IH} or greater) applied to the pin causes a high value (1) in the data input register (GIODIN[7:0]). When a low voltage (V_{IL} or less) is applied to the pin, the data input register reads a low value (0). The V_{IH} and V_{IL} values are device specific and can be found in the device datasheet.

- Data output (GIODOUT)

  Configures the logic level to be output on GIO terminal(s) configured as outputs. A low value (0) written to the data output register forces the pin to a low output voltage (V_{OL} or lower). A high value (1) written to the data output register (GIODOUTx) forces the pin to a high output voltage (V_{OH} or higher) if the open drain functionality is disabled (GIOPDRx[7:0]). If open drain functionality is enabled, a high value (1) written to the data output register forces the pin to a high-impedance state (Z).

- Data set (GIODSET)

  Allows logic HIGH to be output on GIO terminal(s) configured as outputs by writing 1's to the required bits in the GIODSETx registers. If open drain functionality is enabled, a high value (1) written to the data output register forces the pin to a high-impedance state (Z). The GIODSETx registers eliminate the need for the application to perform a read-modify-write operation when it needs to set one or more GIO pin(s).

- Data clear (GIODCLR)

  Allows logic LOW to be output on GIO terminal(s) configured as outputs by writing 1s to the required bits in the GIODCLRx registers. The GIODCLRx registers eliminate the need for the application to perform a read-modify-write operation when it needs to clear one or more GIO pin(s).

- Open drain (GIOPDR)

  Open drain functionality is enabled or disabled (default) using the open drain register GIOPDR[7:0] register. If open-drain mode output is enabled on a pin, a high value (1) written to the data output register (GIODOUTx[7:0]) forces the pin to a high impedance state (Z).

- Pull disable (GIOPULDIS)

  Disables the internal pull on GIO terminal(s) configured as inputs by writing to the GIOPULDISx registers.
- Pull select (GIOPSL)

  Selects internal pull down (default) or pull up on GIO terminal(s) configured as inputs by writing to the GIOPULSELx registers.

Refer to the specific device's datasheet to identify the number of GIO ports as well as the input and output functions supported. Some devices may not support the programmable pull controls. In that case, the pull disable and the pull select register controls will not work.

25.3.2 Interrupt Function

The GIO module supports up to 32 terminals to be configured for generating an interrupt to the host processor through the Vectored Interrupt Manager (VIM). The main functions of the interrupt block are:

- Select the GIO pin(s) that is/are used to generate interrupt(s)

  This is done via the interrupt enable set and clear registers, GIOENASET and GIOENACLR.
- Select the edge on the selected GIO pin(s) that is/are used to generate interrupt(s): rising/falling/both

  Rising or falling edge can be selected via the GIOPOL register. If interrupt is required to be generated on both rising and falling edges, this can be configured via the GIOINTDET register.
- Select the interrupt priority

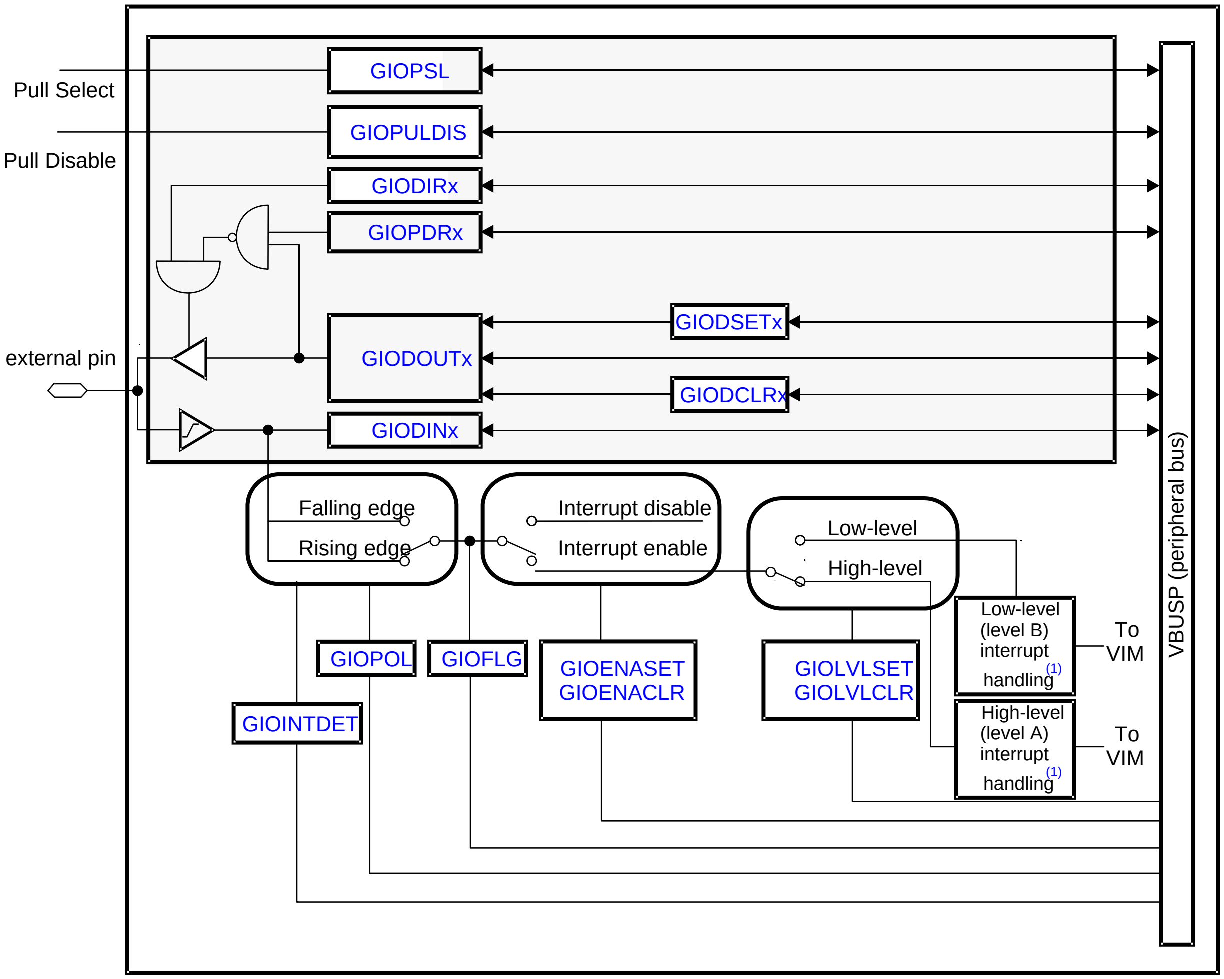
  Low- or high-level interrupt can be selected through the GIOLVLSET and GIOLVLCLR registers.
- Individual interrupt flags are set in the GIOFLG register

The terminals on GIO ports A through D are all interrupt-capable and can be used to handle either general I/O functions or interrupt requests. Each interrupt request can be connected to the VIM at one of two different levels – High (or A) and Low (or B), depending on the VIM channel number. The VIM has an inherent priority scheme so that a request on a lower number channel has a higher priority than a request on a higher number channel. Refer the device datasheet to identify the VIM channel numbers for the GIO level A and level B interrupt requests. Also note that the interrupt priority of level A and level B interrupt handling blocks can be re-programmed in the VIM.

25.3.3 GIO Block Diagram

The GIO block diagram (Figure 25-4) represents the flow of information through a pin. The shaded area corresponds to the I/O block; the unshaded area corresponds to the interrupt block.

Figure 25-4. GIO Block Diagram



(1) A single low-level-interrupt-handling block and a single high-level-interrupt-handling block service all of the interrupt-capable external pins, but only one pin can be serviced by an interrupt block at a time.

25.4 Device Modes of Operation

The GIO module behaves differently in different modes of operation. There are two main modes:

- Emulation mode
- Power-down mode (low-power mode)

25.4.1 Emulation Mode

Emulation mode is used by debugger tools to stop the CPU at breakpoints to read registers.

NOTE: Emulation Mode and Emulation Registers

Emulation mode is a mode of operation of the device and is separate from the GIO emulation registers (GIOEMU1 and GIOEMU2). The contents of these emulation registers are identical to the contents of GIO offset registers (GIOOFF1 and GIOOFF2). Both emulation registers and GIO offset registers are NOT cleared when they are read in emulation mode. GIO offset registers are cleared when they are read in normal mode (other than emulation mode). The emulation registers are NOT cleared when they are read in normal mode. The intention for the emulation registers is that software can use them without clearing the flags.

During emulation mode:

- External interrupts are not captured because the VIM is unable to service interrupts.
- Any register can be read without affecting the state of the system.
- A write to a register still does affect the state of the system.

25.4.2 Power-Down Mode (Low-Power Mode)

In power-down mode, the clock signal to the GIO module is disabled. Thus, there is no switching and the only current draw comes from leakage current. In power-down mode, interrupt pins become level-sensitive rather than edge-sensitive. The polarity bit changes function from falling-edge-triggered to low-level-triggered and rising-edge-triggered to high-level-triggered. A corresponding level on an interrupt pin pulls the module out of low-power mode, if the interrupt is also enabled to wake up the device out of a low-power mode.

25.4.2.1 Module-Level Power Down

The GIO module can be placed into a power down state by disabling the GIO peripheral module via the appropriate bit in the peripheral power down register. Please refer to the Peripheral Central Resource Registers (Section 2.5.3) for details.

25.4.2.2 Device-Level Power Down

The entire device can be placed in one of the pre-defined low-power modes: doze, snooze, or sleep using the clock source and clock domain disable registers in the system module.

25.5 GIO Control Registers

Table 25-1 shows the summary of the GIO registers. The registers are accessible in 8-, 16-, and 32-bit reads or writes.

The start address for the GIO module is FFF7 BC00h.

The GIO module supports up to 8 ports. Refer to your device-specific data manual to identify the actual number of GIO ports and the number of pins in each GIO port implemented on this device.

The GIO module supports up to 4 interrupt-capable ports. Refer to the device datasheet to identify the actual number of interrupt-capable GIO ports and the number of pins in each GIO port implemented on this device.

Table 25-1. GIO Control Registers

Offset	Acronym	Register Description	Section
00h	GIOGCR0	GIO Global Control Register	Section 25.5.1
08h	GIOINTDET	GIO Interrupt Detect Register	Section 25.5.2
0Ch	GIOPOL	GIO Interrupt Polarity Register	Section 25.5.3
10h	GIOENASET	GIO Interrupt Enable Set Register	Section 25.5.4.1
14h	GIOENACLR	GIO Interrupt Enable Clear Register	Section 25.5.4.2
18h	GIOLVLSET	GIO Interrupt Priority Set Register	Section 25.5.5.1
1Ch	GIOLVLCLR	GIO Interrupt Priority Clear Register	Section 25.5.5.2
20h	GIOFLG	GIO Interrupt Flag Register	Section 25.5.6
24h	GIOOFF1	GIO Offset 1 Register	Section 25.5.7
28h	GIOOFF2	GIO Offset 2 Register	Section 25.5.8
2Ch	GIOEMU1	GIO Emulation 1 Register	Section 25.5.9
30h	GIOEMU2	GIO Emulation 2 Register	Section 25.5.10
34h	GIODIRA	GIO Data Direction Register	Section 25.5.11
38h	GIODINA	GIO Data Input Register	Section 25.5.12
3Ch	GIODOUTA	GIO Data Output Register	Section 25.5.13
40h	GIODSETA	GIO Data Set Register	Section 25.5.14
44h	GIODCLRA	GIO Data Clear Register	Section 25.5.15
48h	GIOPDRA	GIO Open Drain Register	Section 25.5.16
4Ch	GIOPULDISA	GIO Pull Disable Register	Section 25.5.17
50h	GIOPSLA	GIO Pull Select Register	Section 25.5.18
54h	GIODIRB	GIO Data Direction Register	Section 25.5.11
58h	GIODINB	GIO Data Input Register	Section 25.5.12
5Ch	GIODOUTB	GIO Data Output Register	Section 25.5.13
60h	GIODSETB	GIO Data Set Register	Section 25.5.14
64h	GIODCLRB	GIO Data Clear Register	Section 25.5.15
68h	GIOPDRB	GIO Open Drain Register	Section 25.5.16
6Ch	GIOPULDISB	GIO Pull Disable Register	Section 25.5.17
70h	GIOPSLB	GIO Pull Select Register	Section 25.5.18

25.5.1 GIO Global Control Register (GIOGCR0)

The GIOGCR0 register contains one bit that controls the module reset status. Writing a 0 to this bit puts the module in a reset state. After system reset, this bit must be set to 1 before configuring any other register of the GIO module. Figure 25-5 and Table 25-2 describe this register.

Figure 25-5. GIO Global Control Register (GIOGCR0) [offset = 00h]

31–16
Reserved
R-0

15–1	0
Reserved	RESET
R-0	R/WP-0

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

Table 25-2. GIO Global Control Register (GIOGCR0) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	RESET		GIO reset.
		0	The GIO is in reset state.
		1	The GIO is operating normally.

NOTE: Note that putting the GIO module in reset state is not the same as putting it in a low-power state.

25.5.2 GIO Interrupt Detect Register (GIOINTDET)

The GIO module supports generation of an interrupt request to CPU when a rising edge, falling edge, or both edges is detected on one or more GIO pin(s). The GIOINTDET register allows both rising and falling edges to be detected, while the GIOPOL register allows the application to define whether a rising edge or a falling edge is to be detected. Figure 25-6 and Table 25-3 describe this register.

Figure 25-6. GIO Interrupt Detect Register (GIOINTDET) [offset = 08h]

31-24	23-16
GIOINTDET 3	GIOINTDET 2
R/W-0	R/W-0

15-8	7-0
GIOINTDET 1	GIOINTDET 0
R/W-0	R/W-0

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

Table 25-3. GIO Interrupt Detect Register (GIOINTDET) Field Descriptions

Bit	Field	Value	Description
31-24	GIOINTDET 3		Interrupt detection select for pins GIOD[7:0]
		0	The flag sets on either a falling or a rising edge on the corresponding pin, depending on the polarity setup in the polarity register (GIOPOL).
		1	The flag sets on both the rising and falling edges on the corresponding pin.
23-16	GIOINTDET 2		Interrupt detection select for pins GIOC[7:0]
		0	The flag sets on either a falling or a rising edge on the corresponding pin, depending on the polarity setup in the polarity register (GIOPOL).
		1	The flag sets on both the rising and falling edges on the corresponding pin.
15-8	GIOINTDET 1		Interrupt detection select for pins GIOB[7:0]
		0	The flag sets on either a falling or a rising edge on the corresponding pin, depending on the polarity setup in the polarity register (GIOPOL).
		1	The flag sets on both the rising and falling edges on the corresponding pin.
7-0	GIOINTDET 0		Interrupt detection select for pins GIOA[7:0]
		0	The flag sets on either a falling or a rising edge on the corresponding pin, depending on the polarity setup in the polarity register (GIOPOL).
		1	The flag sets on both the rising and falling edges on the corresponding pin.

25.5.3 GIO Interrupt Polarity Register (GIOPOL)

The GIOPOL register configures the polarity of the edge, rising edge or falling edge, that needs to be detected. When the device is in low-power mode, the GIOPOL register controls the ***level***, high or low, which will be detected by the GIO module. Figure 25-7 and Table 25-4 describe this register.

Figure 25-7. GIO Interrupt Polarity Register (GIOPOL) [offset = 0Ch]

31	24	23	16
GIOPOL 3		GIOPOL 2	
R/W-0		R/W-0	

15	8	7	0
GIOPOL 1		GIOPOL 0	
R/W-0		R/W-0	

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

Table 25-4. GIO Interrupt Polarity Register (GIOPOL) Field Descriptions

Bit	Field	Value	Description
31-24	GIOPOL 3		Interrupt polarity select for pins GIOD[7:0]
			Normal operation (user or privileged mode):
		0	The flag is set on the falling edge on the corresponding pin.
		1	The flag is set on the rising edge on the corresponding pin.
			Low-power mode (GIO module clocks off):
		0	The interrupt is triggered on the low level.
		1	The interrupt is triggered on the high level.
23-16	GIOPOL 2		Interrupt polarity select for pins GIOC[7:0]
			Normal operation (user or privileged mode):
		0	The flag is set on the falling edge on the corresponding pin.
		1	The flag is set on the rising edge on the corresponding pin.
			Low-power mode (GIO module clocks off):
		0	The interrupt is triggered on the low level.
		1	The interrupt is triggered on the high level.
15-8	GIOPOL 1		Interrupt polarity select for pins GIOB[7:0]
			Normal operation (user or privileged mode):
		0	The flag is set on the falling edge on the corresponding pin.
		1	The flag is set on the rising edge on the corresponding pin.
			Low-power mode (GIO module clocks off):
		0	The interrupt is triggered on the low level.
		1	The interrupt is triggered on the high level.
7-0	GIOPOL 0		Interrupt polarity select for pins GIOA[7:0]
			Normal operation (user or privileged mode):
		0	The flag is set on the falling edge on the corresponding pin.
		1	The flag is set on the rising edge on the corresponding pin.
			Low-power mode (GIO module clocks off):
		0	The interrupt is triggered on the low level.
		1	The interrupt is triggered on the high level.

25.5.4 GIO Interrupt Enable Registers (GIOENASET and GIOENACLR)

The GIOENASET and GIOENACLR registers control which interrupt-capable pins are actually configured as interrupts. If the interrupt is enabled, the rising edge, falling edge, or both edges on the selected pin lead to an interrupt request.

25.5.4.1 GIOENASET Register

Figure 25-8 and Table 25-5 describe this register.

> **NOTE: Enabling Interrupt at the Device Level**
>
> The interrupt channel in the Vectored Interrupt Manager (VIM) must be enabled for the interrupt request to be forwarded to the CPU. Additionally, the ARM CPU (CPSR bit 7 or 6) must be cleared to respond to interrupt requests (IRQ/FIQ).

Figure 25-8. GIO Interrupt Enable Set Register (GIOENASET) [offset = 10h]

31	24	23	16
GIOENASET 3		GIOENASET 2	
R/W-0		R/W-0	
15	8	7	0
GIOENASET 1		GIOENASET 0	
R/W-0		R/W-0	

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

Table 25-5. GIO Interrupt Enable Set Register (GIOENASET) Field Descriptions

Bit	Field	Value	Description
31-24	GIOENASET 3		Interrupt enable for pins GIOD[7:0]
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: Enables the interrupt.
23-16	GIOENASET 2		Interrupt enable for pins GIOC[7:0]
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: Enables the interrupt.
15-8	GIOENASET 1		Interrupt enable for pins GIOB[7:0]
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: Enables the interrupt.
7-0	GIOENASET 0		Interrupt enable for pins GIOA[7:0]
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: Enables the interrupt.

25.5.4.2 GIOENACLR Register

This register disables the interrupt. Figure 25-9 and Table 25-6 describe this register.

Figure 25-9. GIO Interrupt Enable Clear Register (GIOENACLR) [offset = 14h]

31 24	23 16
GIOENACLR 3	GIOENACLR 2
R/W-0	R/W-0

15 8	7 0
GIOENACLR 1	GIOENACLR 0
R/W-0	R/W-0

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

Table 25-6. GIO Interrupt Enable Clear Register (GIOENACLR) Field Descriptions

Bit	Field	Value	Description
31-24	GIOENACLR 3		Interrupt disable for pins GIOD[7:0]
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: Disables the interrupt.
23-16	GIOENACLR 2		Interrupt disable for pins GIOC[7:0]
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: Disables the interrupt.
15-8	GIOENACLR 1		Interrupt disable for pins GIOB[7:0]
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: Disables the interrupt.
7-0	GIOENACLR 0		Interrupt disable for pins GIOA[7:0]
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: Disables the interrupt.

25.5.5 GIO Interrupt Priority Registers (GIOLVLSET and GIOLVLCLR)

The GIOLVLSET and GIOLVLCLR registers configure the interrupts as high-level (level A) or low-level (level B) going to the Vectored Interrupt Manager (VIM). Each interrupt is individually configured.

- The high-level interrupts are recorded to GIOOFF1 and GIOEMU1.
- The low-level interrupts are recorded to GIOOFF2 and GIOEMU2.

NOTE: The GIO module can generate two interrupt requests. These are connected to two separate channels on the Vectored Interrupt Manager (VIM). The lower-numbered VIM channels are higher priority. The GIO interrupt connected to a lower-number channel is the high-level (also called level A) GIO interrupt, while the GIO interrupt connected to a higher-number channel is the low-level (also called level B) GIO interrupt.

25.5.5.1 GIOLVLSET Register

The GIOLVLSET register is used to configure an interrupt as a high-level interrupt going to the VIM. An interrupt can be configured as a high-level interrupt by writing a 1 into the corresponding bit of the GIOLVLSET register. Writing a 0 has no effect. Figure 25-10 and Table 25-7 describe this register.

Figure 25-10. GIO Interrupt Priority Register (GIOLVLSET) [offset = 18h]

31			16
GIOLVLSET 3		GIOLVLSET 2	
R/W-0		R/W-0	

15	8	7	0
GIOLVLSET 1		GIOLVLSET 0	
R/W-0		R/W-0	

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

Table 25-7. GIO Interrupt Priority Register (GIOLVLSET) Field Descriptions

Bit	Field	Value	Description
31-24	GIOLVLSET 3		GIO high-priority interrupt for pins GIOD[7:0].
		0	Read: The interrupt is a low-level interrupt. The low-level interrupts are recorded to GIOOFF2 and GIOEMU2. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is set as a high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1. Write: Sets the interrupt as a high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1.
23-16	GIOLVLSET 2		GIO high-priority interrupt for pins GIOC[7:0].
		0	Read: The interrupt is a low-level interrupt. The low-level interrupts are recorded to GIOOFF2 and GIOEMU2. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is set as a high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1. Write: Sets the interrupt as a high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1.
15-8	GIOLVLSET 1		GIO high-priority interrupt for pins GIOB[7:0].
		0	Read: The interrupt is a low-level interrupt. The low-level interrupts are recorded to GIOOFF2 and GIOEMU2. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is set as a high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1. Write: Sets the interrupt as a high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1.

Table 25-7. GIO Interrupt Priority Register (GIOLVLSET) Field Descriptions (continued)

Bit	Field	Value	Description
7-0	GIOLVLSET 0		GIO high-priority interrupt for pins GIOA[7:0].
		0	Read: The interrupt is a low-level interrupt. The low-level interrupts are recorded to GIOOFF2 and GIOEMU2. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is set as a high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1. Write: Sets the interrupt as a high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1.

25.5.5.2 GIOLVLCLR Register

The GIOLVLCLR register is used to configure an interrupt as a low-level interrupt going to the VIM. An interrupt can be configured as a low-level interrupt by writing a 1 into the corresponding bit of the GIOLVLCLR register. Writing a 0 has no effect. Figure 25-11 and Table 25-8 describe this register.

Figure 25-11. GIO Interrupt Priority Register (GIOLVLCLR) [offset = 1Ch]

31	16
GIOLVLCLR 3	GIOLVLCLR 2
R/W-0	R/W-0

15 8	7 0
GIOLVLCLR 1	GIOLVLCLR 0
R/W-0	R/W-0

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

Table 25-8. GIO Interrupt Priority Register (GIOLVLCLR) Field Descriptions

Bit	Field	Value	Description
31-24	GIOLVLCLR 3		GIO low-priority interrupt for pins GIOD[7:0]
		0	Read: The interrupt is a low-level interrupt. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is set as a high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1. Write: Sets the interrupt as a low-level interrupt. The low-level interrupts are recorded to GIOOFF2 and GIOEMU2.
23-16	GIOLVLCLR 2		GIO low-priority interrupt for pins GIOC[7:0]
		0	Read: The interrupt is a low-level interrupt. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is set as a high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1. Write: Sets the interrupt as a low-level interrupt. The low-level interrupts are recorded to GIOOFF2 and GIOEMU2.
15-8	GIOLVLCLR 1		GIO low-priority interrupt for pins GIOB[7:0]
		0	Read: The interrupt is a low-level interrupt. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is set as a high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1. Write: Sets the interrupt as a low-level interrupt. The low-level interrupts are recorded to GIOOFF2 and GIOEMU2.
7-0	GIOLVLCLR 0		GIO low-priority interrupt for pins GIOA[7:0]
		0	Read: The interrupt is a low-level interrupt. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is set as a high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1. Write: Sets the interrupt as a low-level interrupt. The low-level interrupts are recorded to GIOOFF2 and GIOEMU2.

25.5.6 GIO Interrupt Flag Register (GIOFLG)

The GIOFLG register contains flags indicating that the transition edge (as set in GIOINTDET and GIOPOL registers) has occurred. The flag can be cleared by the CPU writing a 1 to the flag that is set. The flag is also cleared by reading the appropriate interrupt offset register (GIOOFF1 or GIOOFF2). Figure 25-12 and Table 25-9 describe this register.

Figure 25-12. GIO Interrupt Flag Register (GIOFLG) [offset = 20h]

<table>
<tr><td>31</td><td>24</td><td>23</td><td>16</td></tr>
<tr><td colspan="2">GIOFLG 3</td><td colspan="2">GIOFLG 2</td></tr>
<tr><td colspan="2">R/W1C-0</td><td colspan="2">R/W1C-0</td></tr>
<tr><td>15</td><td>8</td><td>7</td><td>0</td></tr>
<tr><td colspan="2">GIOFLG 1</td><td colspan="2">GIOFLG 0</td></tr>
<tr><td colspan="2">R/W1C-0</td><td colspan="2">R/W1C-0</td></tr>
</table>

LEGEND: R/W = Read/Write; W1C = Write 1 to clear; -*n* = value after reset

Table 25-9. GIO Interrupt Flag Register (GIOFLG) Field Descriptions

Bit	Field	Value	Description
31-24	GIOFLG 3	0	GIO flag for pins GIOD[7:0] Read: A transition has not occurred since the last clear. Write: Writing a 0 to this bit has no effect.
		1	Read: The selected transition on the corresponding pin has occurred. Write: The corresponding bit is cleared to 0. **Note: This bit is also cleared by a read to the corresponding bit in the appropriate offset register.**
23-16	GIOFLG 2	0	GIO flag for pins GIOC[7:0] Read: A transition has not occurred since the last clear. Write: Writing a 0 to this bit has no effect.
		1	Read: The selected transition on the corresponding pin has occurred. Write: The corresponding bit is cleared to 0. **Note: This bit is also cleared by a read to the corresponding bit in the appropriate offset register.**
15-8	GIOFLG 1	0	GIO flag for pins GIOB[7:0] Read: A transition has not occurred since the last clear. Write: Writing a 0 to this bit has no effect.
		1	Read: The selected transition on the corresponding pin has occurred. Write: The corresponding bit is cleared to 0. **Note: This bit is also cleared by a read to the corresponding bit in the appropriate offset register.**
7-0	GIOFLG 0	0	GIO flag for pins GIOA[7:0] Read: A transition has not occurred since the last clear. Write: Writing a 0 to this bit has no effect.
		1	Read: The selected transition on the corresponding pin has occurred. Write: The corresponding bit is cleared to 0. **Note: This bit is also cleared by a read to the corresponding bit in the appropriate offset register.**

NOTE: An interrupt flag gets set when the selected transition happens on the corresponding GIO pin ***regardless of whether the interrupt generation is enabled or not***. It is recommended to clear a flag before enabling the interrupt generation for a transition on the corresponding GIO pin.

25.5.7 GIO Offset Register 1 (GIOOFF1)

The GIOOFF1 register provides a numerical offset value that represents the pending external interrupt with high priority. The offset value can be used to locate the position of the interrupt routine in a vector table in application software. Figure 25-13 and Table 25-10 describe this register.

NOTE: Reading this register clears it, GIOEMU1 and the corresponding flag bit in the GIOFLG register. However, in emulation mode, a read to this register does not clear any register or flag. If more than one GIO interrupts are pending, then reading the GIOOFF1 register will change the contents of GIOOFF1 and GIOEMU1 to show the offset value for the next highest-priority pending interrupt. The application can choose to service all GIO interrupts from the same service routine by continuing to read the GIOOFF1 register until it reads zeros.

Figure 25-13. GIO Offset 1 Register (GIOOFF1) [offset = 24h]

31–16
Reserved
R-0

15–6	5–0
Reserved	GIOOFF1
R-0	R-0

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

Table 25-10. GIO Offset 1 Register (GIOOFF1) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reads return 0. Writes have no effect.
5-0	GIOOFF1		GIO offset 1. These bits index the currently pending high-priority interrupt. This register and the flag bit (in the GIOFLG register) are also cleared when this register is read, except in emulation mode.
		0	No interrupt is pending.
		1h	Interrupt 0 (corresponding to GIOA0) is pending with a high priority.
		:	:
		8h	Interrupt 7 (corresponding to GIOA7) is pending with a high priority.
		9h	Interrupt 8 (corresponding to GIOB0) is pending with a high priority.
		:	:
		10h	Interrupt 16 (corresponding to GIOB7) is pending with a high priority.
		:	:
		20h	Interrupt 32 (corresponding to GIOD7) is pending with a high priority.
		21h-3Fh	Reserved

25.5.8 GIO Offset B Register (GIOOFF2)

The GIOOFF2 register provides a numerical offset value that represents the pending external interrupt with low priority. The offset value can be used to locate the position of the interrupt routine in a vector table in application software. Figure 25-14 and Table 25-11 describe this register.

NOTE: Reading this register clears it, GIOEMU2 and the corresponding flag bit in the GIOFLG register. However, in emulation mode, a read to this register does not clear any register or flag. If more than one GIO interrupts are pending, then reading the GIOOFF1 register will change the contents of GIOOFF2 and GIOEMU2 to show the offset value for the next highest-priority pending interrupt. The application can choose to service all GIO interrupts from the same service routine by continuing to read the GIOOFF1 register until it reads zeros.

Figure 25-14. GIO Offset 2 Register (GIOOFF2) [offset = 28h]

<table>
<tr><td>31</td><td>16</td></tr>
<tr><td colspan="2">Reserved</td></tr>
<tr><td colspan="2">R-0</td></tr>
</table>

<table>
<tr><td>15</td><td>6</td><td>5</td><td>0</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="2">GIOOFF2</td></tr>
<tr><td colspan="2">R-0</td><td colspan="2">R-0</td></tr>
</table>

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

Table 25-11. GIO Offset 2 Register (GIOOFF2) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reads return 0. Writes have no effect.
5-0	GIOOFF2		GIO offset 2. These bits index the currently pending low-priority interrupt. This register and the flag bit (in the GIOFLG register) are also cleared when this register is read, except in emulation mode.
		0	No interrupt is pending.
		1h	Interrupt 0 (corresponding to GIOA0) is pending with a low priority.
		:	:
		8h	Interrupt 7 (corresponding to GIOA7) is pending with a low priority.
		9h	Interrupt 8 (corresponding to GIOB0) is pending with a low priority.
		:	:
		10h	Interrupt 16 (corresponding to GIOB7) is pending with a low priority.
		:	:
		20h	Interrupt 32 (corresponding to GIOD7) is pending with a low priority.
		21h-3Fh	Reserved

25.5.9 GIO Emulation A Register (GIOEMU1)

The GIOEMU1 register is a read-only register. The contents of this register are identical to the contents of GIOOFF1. The intention for the this register is that software can use it without clearing the flags. Figure 25-15 and Table 25-12 describe this register.

NOTE: The corresponding flag in the GIOFLG register is not cleared when the GIOEMU1 register is read.

Figure 25-15. GIO Emulation 1 Register (GIOEMU1) [offset = 2Ch]

31–16
Reserved
R-0

15–6	5–0
Reserved	GIOEMU1
R-0	R-0

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

Table 25-12. GIO Emulation 1 Register (GIOEMU1) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reads return 0. Writes have no effect.
5-0	GIOEMU1		GIO offset emulation 1. These bits index the currently pending high-priority interrupt. No register or flag is cleared by reading this register.
		0	No interrupt is pending.
		1h	Interrupt 0 (corresponding to GIOA0) is pending with a high priority.
		:	:
		8h	Interrupt 7 (corresponding to GIOA7) is pending with a high priority.
		9h	Interrupt 8 (corresponding to GIOB0) is pending with a high priority.
		:	:
		10h	Interrupt 16 (corresponding to GIOB7) is pending with a high priority.
		:	:
		20h	Interrupt 32 (corresponding to GIOD7) is pending with a high priority.
		21h-3Fh	Reserved

25.5.10 GIO Emulation B Register (GIOEMU2)

The GIOEMU2 register is a read-only register. The contents of this register are identical to the contents of GIOOFF2. The intention for the this register is that software can use it without clearing the flags. Figure 25-16 and Table 25-13 describe this register.

NOTE: The corresponding flag in the GIOFLG register is not cleared when the GIOEMU2 register is read.

Figure 25-16. GIO Emulation 2 Register (GIOEMU2) [offset = 30h]

31-16
Reserved
R-0

15-6	5-0
Reserved	GIOEMU2
R-0	R-0

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

Table 25-13. GIO Emulation 2 Register (GIOEMU2) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reads return 0. Writes have no effect.
5-0	GIOEMU2		GIO offset emulation 2. These bits index the currently pending low-priority interrupt. No register or flag is cleared by reading this register.
		0	No interrupt is pending.
		1h	Interrupt 0 (corresponding to GIOA0) is pending with a low priority.
		:	:
		8h	Interrupt 7 (corresponding to GIOA7) is pending with a low priority.
		9h	Interrupt 8 (corresponding to GIOB0) is pending with a low priority.
		:	:
		10h	Interrupt 16 (corresponding to GIOB7) is pending with a low priority.
		:	:
		20h	Interrupt 32 (corresponding to GIOD7) is pending with a low priority.
		21h-3Fh	Reserved

25.5.11 GIO Data Direction Registers (GIODIR[A-B])

The GIODIR register controls whether the pins of a given port are configured as inputs or outputs. Figure 25-17 and Table 25-14 describe this register.

Figure 25-17. GIO Data Direction Registers (GIODIR[A-B]) [offset = 34h, 54h]

31-16
Reserved
R-0

15-8	7-0
Reserved	GIODIR[7:0]
R-0	R/W-0

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

Table 25-14. GIO Data Direction Registers (GIODIR[A-B]) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	GIODIR[*n*]		GIO data direction of port n, pins [7:0]
		0	The GIO pin is an input. Note: If the pin direction is set as an input, the output buffer is tristated.
		1	The GIO pin is an output.

25.5.12 GIO Data Input Registers (GIODIN[A-B])

Values in the GIODIN register reflect the current state (high = 1 or low = 0) on the pins of the port. Figure 25-18 and Table 25-15 describe this register.

Figure 25-18. GIO Data Input Registers (GIODIN[A-B]) [offset = 38h, 58h]

31-16
Reserved
R-0

15-8	7-0
Reserved	GIODIN[7:0]
R-0	R/W-0

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

Table 25-15. GIO Data Input Registers (GIODIN[A-B]) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	GIODIN[*n*]		GIO data input for port n, pins [7:0]
		0	The pin is at logic low (0).
		1	The pin is at logic high (1).

25.5.13 GIO Data Output Registers (GIODOUT[A-B])

Values in the GIODOUT register specify the output state (high = 1 or low = 0) of the pins of the port when they are configured as outputs. Figure 25-19 and Table 25-16 describe this register.

NOTE: Values in the GIODSET register set the data output control register bits to 1 regardless of the current value in the GIODOUT bits.

Figure 25-19. GIO Data Output Registers (GIODOUT[A-B]) [offset = 3Ch, 5Ch]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		GIODOUT[7:0]	
R-0		R/W-0	

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

Table 25-16. GIO Data Output Registers (GIODOUT[A-B]) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	GIODOUT[*n*]		GIO data output of port n, pins[7:0].
		0	The pin is driven to logic low (0).
		1	The pin is driven to logic high (1).
			Note: Output is in high impedance state if the GIOPDRx bit = 1 and GIODOUTx bit = 1.
			Note: GIO pin is placed in output mode by setting the GIODIRx bit to 1.

25.5.14 GIO Data Set Registers (GIODSET[A-B])

Values in this register set the data output control register bits to 1 regardless of the current value in the GIODOUT bits. The contents of this register reflect the contents of GIODOUT. Figure 25-20 and Table 25-17 describe this register.

Figure 25-20. GIO Data Set Registers (GIODSET[A-B]) [offset = 40h, 60h]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		GIODSET[7:0]	
R-0		R/W-0	

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

Table 25-17. GIO Data Set Registers (GIODSET[A-B]) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	GIODSET[*n*]		GIO data set for port n, pins[7:0]. This bit drives the output of GIO pin high.
		0	Write: Writing a 0 has no effect.
		1	Write: The corresponding GIO pin is driven to logic high (1).
			Note: The current logic state of the GIODOUT bit will also be displayed by this bit.
			Note: GIO pin is placed in output mode by setting the GIODIRx bit to 1.

25.5.15 GIO Data Clear Registers (GIODCLR[A-B])

Values in this register clear the data output register (GIO Data Output Register [A-H]) bit to 0 regardless of its current value. The contents of this register reflect the contents of GIODOUT. Figure 25-21 and Table 25-18 describe this register.

Figure 25-21. GIO Data Clear Registers (GIODCLR[A-B]) [offset = 44h, 64h]

31			16
Reserved			
R-0			
15	8	7	0
Reserved		GIODCLR[7:0]	
R-0		R/W-0	

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

Table 25-18. GIO Data Clear Registers (GIODCLR[A-B]) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	GIODCLR[*n*]		GIO data clear for port n, pins[7:0]. This bit drives the output of GIO pin low.
		0	Write: Writing a 0 has no effect.
		1	Write: The corresponding GIO pin is driven to logic low (0).
			Note: The current logic state of the GIODOUT bit will also be displayed by this bit.
			Note: GIO pin is placed in output mode by setting the GIODIRx bit to 1.

25.5.16 GIO Open Drain Registers (GIOPDR[A-B])

Values in this register enable or disable the open drain capability of the data pins. Figure 25-22 and Table 25-19 describe this register.

Figure 25-22. GIO Open Drain Registers (GIOPDR[A-B]) [offset = 48h, 68h]

31			16
Reserved			
R-0			
15	8	7	0
Reserved		GIOPDR[7:0]	
R-0		R/W-0	

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

Table 25-19. GIO Open Drain Registers (GIOPDR[A-B]) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	GIOPDR[*n*]		GIO open drain for port n, pins[7:0]
		0	The GIO pin is configured in push/pull (normal GIO) mode. The output voltage is V_{OL}or lower if GIODOUT bit = 0 and V_{OH} or higher if GIODOUT bit = 1.
		1	The GIO pin is configured in open drain mode. The GIODOUTx bit controls the state of the GIO output buffer: GIODOUTx = 0, the GIO output buffer is driven low; GIODOUTx = 1, the GIO output buffer is tristated.

25.5.17 GIO Pull Disable Registers (GIOPULDIS[A-B])

Values in this register enable or disable the pull control capability of the pins. Figure 25-23 and Table 25-20 describe this register.

Figure 25-23. GIO Pull Disable Registers (GIOPULDIS[A-B]) [offset = 4Ch, 6Ch]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		GIOPULDIS[7:0]	
R-0		R/W-0	

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

Table 25-20. GIO Pull Disable Registers (GIOPULDIS[A-B]) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	GIOPULDIS[*n*]		GIO pull disable for port n, pins[7:0]. Writes to this bit will only take effect when the GIO pin configured as an input pin.
		0	The pull functionality is enabled.
		1	The pull functionality is disabled.
			Note: The GIO pin is placed in input mode by clearing the GIODIRx bit to 0.

25.5.18 GIO Pull Select Registers (GIOPSL[A-B])

Values in this register select the pull up or pull down functionality of the pins. Figure 25-24 and Table 25-21 describe this register.

Figure 25-24. GIO Pull Select Registers (GIOPSL[A-B]) [offset = 50h, 70h]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		GIOPSL[7:0]	
R-0		R/W-0	

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

Table 25-21. GIO Pull Select Registers (GIOPSL[A-B]) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	GIOPSL[*n*]		GIO pull select for port n, pins[7:0]
		0	The pull down functionality is select, when pull up/pull down logic is enabled.
		1	The pull up functionality is select, when pull up/pull down logic is enabled.
			Note: The pull up/pull down functionality is enabled by clearing corresponding bit in GIOPULDIS to 0.

25.6 I/O Control Summary

The behavior of the output buffer and the pull control is summarized in Table 25-22.

Table 25-22. Output Buffer and Pull Control Behavior for GIO Pins

Module under Reset?	Pin Direction (GIODIR)[1][2]	Open Drain Enable (GIOPDR)[1][3]	Pull Disable (GIOPULDIS)[1][4]	Pull Select (GIOPSL)[1][5]	Pull Control	Output Buffer[6]
Yes	X	X	X	X	Enabled	Disabled
No	0	X	0	0	Pull down	Disabled
No	0	X	0	1	Pull up	Disabled
No	0	X	1	0	Disabled	Disabled
No	0	X	1	1	Disabled	Disabled
No	1	0	X	X	Disabled	Enabled
No	1	1	X	X	Disabled	Enabled

[1] X = Don't care

[2] GIODIR = 0 for input; = 1 for output

[3] See Section 25.5.16

[4] GIOPULDIS = 0 for enabling pull control; = 1 for disabling pull control

[5] GIOPSL= 0 for pull-down functionality; = 1 for pull-up functionality

[6] If open drain is enabled, output buffer will be disabled if a high level (1) is being output.

Controller Area Network (DCAN) Module

This chapter describes the controller area network (DCAN) module.

NOTE: This chapter describes a superset implementation of the DCAN module. Consult your device-specific datasheet to identify the applicability of the DMA-related features, the number of instantiations of the DCAN IP, and the number of mailboxes supported on your specific device being used.

Topic **Page**

26.1 Overview ... 1329
26.2 CAN Blocks ... 1330
26.3 CAN Bit Timing ... 1332
26.4 CAN Module Configuration ... 1336
26.5 Message RAM ... 1338
26.6 Message Interface Register Sets ... 1342
26.7 Message Object Configurations ... 1345
26.8 Message Handling ... 1347
26.9 CAN Message Transfer ... 1352
26.10 Interrupt Functionality ... 1353
26.11 Global Power-Down Mode ... 1355
26.12 Local Power-Down Mode ... 1356
26.13 GIO Support ... 1357
26.14 Test Modes ... 1358
26.15 Parity Check Mechanism ... 1362
26.16 Debug/Suspend Mode ... 1363
26.17 DCAN Control Registers ... 1363

26.1 Overview

The Controller Area Network is a high-integrity, serial, multi-master communication protocol for distributed real-time applications. This CAN module is implemented according to ISO 11898-1 and is suitable for industrial, automotive and general embedded communications.

26.1.1 Features

The DCAN module provides the following features:

Protocol

- Supports CAN protocol version 2.0 part A, B

Speed

- Bit rates up to 1 MBit/s

MailBox

- Configurable Message objects
- Individual identifier masks for each message object
- Programmable FIFO mode for message objects

High Speed MailBox Access

- DMA access to Message RAM.

Power

- Global power down and wakeup support
- Local power down and wakeup support

Debug

- Suspend mode for debug support
- Programmable loop-back modes for self-test operation
- Direct access to Message RAM in test mode
- Supports Two interrupt lines - Level 0 and Level 1

Others

- Automatic Message RAM initialization
- Automatic bus on after Bus-Off state by a programmable 32-bit timer
- CAN Rx / Tx pins configurable as general purpose IO pins
- Software module reset
- Message RAM parity check mechanism
- Dual clock source to reduce jitter

26.1.2 Functional Description

The CAN protocol is an ISO standard (ISO 11898) for serial data communication. This protocol uses Non-Return To Zero (NRZ) with bit-stuffing. And the communication is carried over a two-wire balanced signaling scheme.

The DCAN data communication happens through the CAN_TX and CAN_RX pins. An additional transceiver hardware is required for the connection to the physical layer (CAN bus) CAN_High and CAN_Low.

The DCAN register set can be accessed directly by the CPU. These registers are used to control and configure the CAN module and the Message RAM.

Individual CAN message objects should be configured for communication over a CAN network. The message objects and identifier masks are stored in the Message RAM.

The CAN module internally handles functions such acceptance filtering, transfer of messages from and to the Message RAM, handling of transmission requests as well as the generation of interrupts or DMA requests.

26.2 CAN Blocks

The DCAN Module, shown in Figure 26-1, comprises of the following basic blocks.

26.2.1 CAN Core

The CAN Core consists of the CAN Protocol Controller and the Rx/Tx Shift Register. It handles all ISO 11898-1 protocol functions.

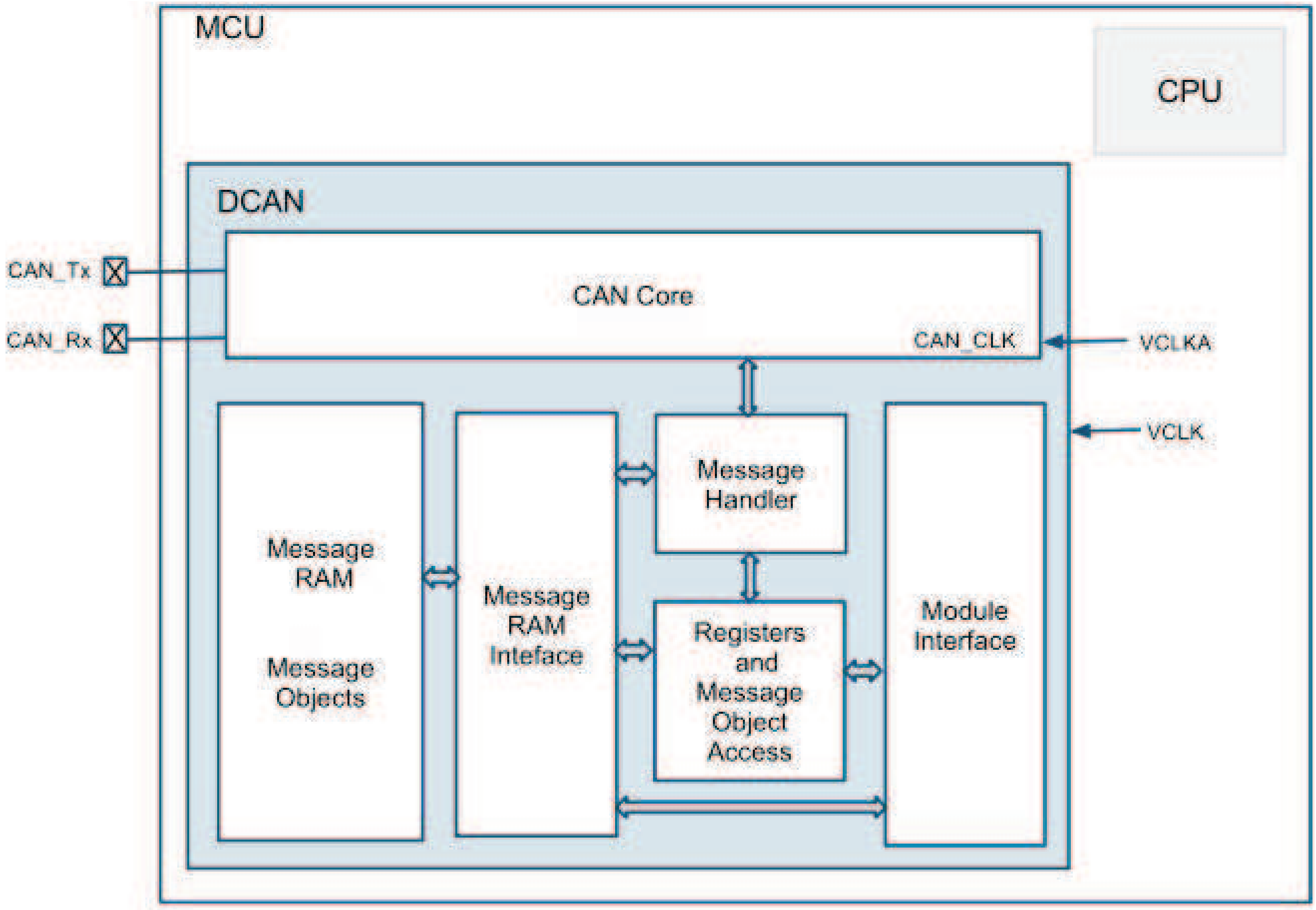
26.2.2 Message RAM

The DCAN Message RAM enables storage of CAN messages.

26.2.3 Message Handler

The Message Handler is a state machine that controls the data transfer between the single ported Message RAM and the CAN Core’s Rx/Tx Shift Register. It also handles acceptance filtering and the interrupt/DMA request generation as programmed in the control registers.

Figure 26-1. Block Diagram



26.2.4 Message RAM Interface

The Interface Register sets control the CPU read and write accesses to the Message RAM.

There are three interface registers IF1, IF2, and IF3.

IF1 and IF2 Interface Registers sets for read and write access.

IF3 Interface Register set for read access only.

The Interface Registers have the same word-length as the Message RAM.

Additional information can be found in Section 26.6.

26.2.5 Register and Message Object Access

During normal operation, data consistency of the message objects is guaranteed by indirectly accessing the message objects through the interface registers IF1 and IF2.

In order to be able to perform tests on the message object memory, a dedicated test mode has been implemented, that allows direct access by either the CPU or DMA. During normal operation direct access has to be avoided.

26.2.6 Dual Clock Source

Two clock domains are provided to the DCAN module:

1. VCLK - The peripheral synchronous clock domain as the general module clock source.
2. VCLKA - The peripheral asynchronous clock source domain provided to the CAN core as clock source (CAN_CLK) for generating the CAN Bit Timing.

If a frequency modulated clock output from FMPLL is used as the VCLK source, then VCLKA should be derived from an unmodulated clock source (for example, OSCIN source).

The clock source for VCLKA is selected by the Peripheral Asynchronous Clock Source Register in the system module.

Both clock domains can be derived from the same clock source (so that VCLK = VCLKA). However, if frequency modulation in the FMPLL is enabled (spread spectrum clock), then due to the high precision clocking requirements of the CAN Core, the FMPLL clock source should not be used for VCLKA. Alternatively, a separate clock without any modulation (for example, derived directly from the OSCIN clock) should be used for VCLKA.

Please refer to the system module reference guide and the device datasheet for more information how to configure the relevant clock source registers in the system module.

Between the two clock domains, a synchronization mechanism is implemented in the DCAN module in order to ensure correct data transfer.

NOTE: If the dual clock functionality is used, then VCLK must always be higher or equal to CAN_CLK (derived from the asynchronous clock source), in order to achieve a stable functionality of the DCAN. Here also the frequency shift of the modulated VCLK has to be considered:

$$f_{0,\ \mathrm{VCLK}} \pm \Delta f_{\mathrm{FM,VCLK}} \geq f_{\mathrm{CANCLK}}$$

The CAN Core has to be programmed to at least 8 clock cycles per bit time. To achieve a transfer rate of 1 MBaud when using the asynchronous clock domain as the clock source for CAN_CLK, an oscillator frequency of 8MHz or higher has to be used.

26.3 CAN Bit Timing

The DCAN supports bit rates between less than 1 kBit/s and 1000 kBit/s.

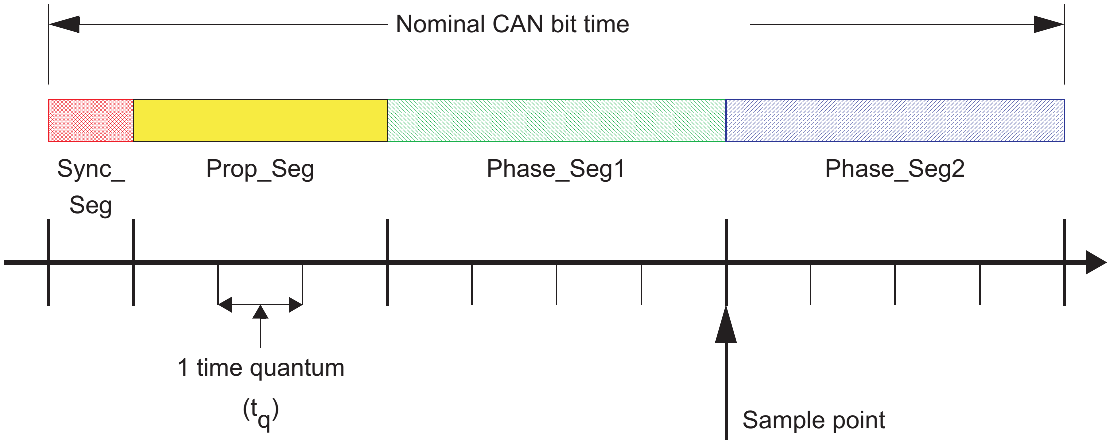
Each member of the CAN network has its own clock generator, typically derived from a crystal oscillator. The Bit timing parameters can be configured individually for each CAN node, creating a common Bit rate even though the CAN nodes' oscillator periods (f_{osc}) may be different.

26.3.1 Bit Time and Bit Rate

According to the CAN specification, the Bit time is divided into four segments (see Figure 26-2):

- Synchronization Segment (Sync_Seg)
- Propagation Time Segment (Prop_Seg)
- Phase Buffer Segment 1 (Phase_Seg1)
- Phase Buffer Segment 2 (Phase_Seg2)

Figure 26-2. Bit Timing



Each segment consists of a specific number of time quanta. The length of one time quantum, (t_q), which is the basic time unit of the bit time, is given by the CAN_CLK and the Baud Rate Prescalers (BRPE and BRP). With these two Baud Rate Prescalers combined, divider values from 1 to 1024 can be programmed:

t_q = Baud Rate Prescaler / CAN_CLK

Apart from the fixed length of the synchronization segment, these numbers are programmable.

Table 26-1 describes the minimum programmable ranges required by the CAN protocol. A given bit rate may be met by different Bit time configurations.

NOTE: For proper functionality of the CAN network, the physical delay times and the oscillator's tolerance range have to be considered.

Table 26-1. Parameters of the CAN Bit Time

Parameter	Range	Remark
Sync_Seg	1 t_q (fixed)	Synchronization of bus input to CAN_CLK
Prop_Seg	[1 … 8] t_q	Compensates for the physical delay times
Phase_Seg1	[1 … 8] t_q	May be lengthened temporarily by synchronization
Phase_Seg2	[1 … 8] t_q	May be shortened temporarily by synchronization
Synchronization Jump Width (SJW)	[1 … 4] t_q	May not be longer than either Phase Buffer Segment

26.3.1.1 Synchronization Segment

The Synchronization Segment (Sync_Seg) is the part of the bit time where edges of the CAN bus level are expected to occur. If an edge occurs outside of Sync_Seg, its distance to the Sync_Seg is called the phase error of this edge.

26.3.1.2 Propagation Time Segment

This part of the bit time is used to compensate physical delay times within the CAN network. These delay times consist of the signal propagation time on the bus and the internal delay time of the CAN nodes.

26.3.1.3 Phase Buffer Segments and Synchronization

The Phase Buffer Segments (Phase_Seg1 and Phase_Seg2) and the Synchronization Jump Width (SJW) are used to compensate for the oscillator tolerance.

The Phase Buffer Segments surround the sample point. The Phase Buffer Segments may be lengthened or shortened by synchronization.

The Synchronization Jump Width (SJW) defines how far the resynchronizing mechanism may move the sample point inside the limits defined by the Phase Buffer Segments to compensate for edge phase errors.

Synchronizations occur on edges from recessive to dominant. Their purpose is to control the distance between edges and sample points.

Edges are detected by sampling the actual bus level in each time quantum and comparing it with the bus level at the previous sample point. A synchronization may be done only if a recessive bit was sampled at the previous sample point and if the actual time quantum’s bus level is dominant.

An edge is synchronous if it occurs inside of Sync_Seg, otherwise its distance to the Sync_Seg is the edge phase error, measured in time quanta. If the edge occurs before Sync_Seg, the phase error is negative, else it is positive.

26.3.1.4 Oscillator Tolerance Range

With the introduction of CAN protocol version 1.2, the option to synchronize on edges from dominant to recessive became obsolete. Only edges from recessive to dominant are considered for synchronization. The protocol update to version 2.0 (A and B) had no influence on the oscillator tolerance.

The tolerance range df for an oscillator’s frequency f_{osc} around the nominal frequency f_{nom} with:

$$(1\text{-df}) \text{ x } f_{nom} \le f_{osc} \le (1\text{+df}) \text{ x } f_{nom} \tag{36}$$

depends on the proportions of Phase_Seg1, Phase_Seg2, SJW, and the bit time. The maximum tolerance df is defined by two conditions (both shall be met):

$$\text{I:} \quad df \le \frac{\min(\text{Phase\_Seg1, Phase\_Seg2})}{[2x(13x\text{bit\_time} - \text{Phase\_Seg2})]}$$

$$\text{II:} \quad df \le \frac{\text{SJW}}{20x\text{bit\_time}} \tag{37}$$

It has to be considered that SJW may not be larger than the smaller of the Phase Buffer Segments and that the Propagation Time Segment limits that part of the bit time that may be used for the Phase Buffer Segments.

The combination Prop_Seg = 1 and Phase_Seg1 = Phase_Seg2 = SJW = 4 allows the largest possible oscillator tolerance of 1.58%. This combination with a Propagation Time Segment of only 10% of the bit time is not suitable for short bit times; it can be used for bit rates of up to 125 kBit/s (bit time = 8 μs) with a bus length of 40 m.

26.3.2 DCAN Bit Timing Registers

In the DCAN, the bit timing configuration is programmed in two register bytes, additionally a third byte for a baud rate prescaler extension of 4 bits (BREP) is provided. The sum of Prop_Seg and Phase_Seg1 (as TSEG1) is combined with Phase_Seg2 (as TSEG2) in one byte, SJW and BRP (plus BRPE in third byte) are combined in the other byte

In this bit timing register, the components TSEG1, TSEG2, SJW and BRP have to be programmed to a numerical value that is one less than its functional value; so instead of values in the range of [1… n], values in the range of [0 … n-1] are programmed. That way, SJW (functional range of [1 … 4]) is represented by only two bits.

Therefore the length of the Bit time is (programmed values) [TSEG1 + TSEG2 + 3] t_q or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] t_q.

The data in the Bit Timing Register (BTR) is the configuration input of the CAN protocol controller. The Baud Rate Prescaler (configured by BRPE/BRP) defines the length of the time quantum (the basic time unit of the bit time); the Bit Timing Logic (configured by TSEG1, TSEG2, and SJW) defines the number of time quanta in the bit time.

26.3.2.1 Calculation of the Bit Timing Parameters

Usually, the calculation of the bit timing configuration starts with a desired bit rate or bit time. The resulting Bit time (1 / Bit rate) must be an integer multiple of the CAN clock period.

NOTE: 8 MHz is the minimum CAN clock frequency required to operate the DCAN at a bit rate of 1 MBit/s.

The bit time may consist of 8 to 25 time quanta. The length of the time quantum t_q is defined by the Baud Rate Prescaler with t_q = (Baud Rate Prescaler) / CAN_CLK. Several combinations may lead to the desired bit time, allowing iterations of the following steps.

First part of the bit time to be defined is the Prop_Seg. Its length depends on the delay times measured in the system. A maximum bus length as well as a maximum node delay has to be defined for expandible CAN bus systems. The resulting time for Prop_Seg is converted into time quanta (rounded up to the nearest integer multiple of t_q).

The Sync_Seg is 1 t_q long (fixed), leaving (bit time – Prop_Seg – 1) t_q for the two Phase Buffer Segments. If the number of remaining t_q is even, the Phase Buffer Segments have the same length, Phase_Seg2 = Phase_Seg1, else Phase_Seg2 = Phase_Seg1 + 1.

The minimum nominal length of Phase_Seg2 has to be regarded as well. Phase_Seg2 may not be shorter than any CAN controller’s Information Processing Time in the network, which is device dependent and can be in the range of [0 … 2] t_q.

The length of the Synchronization Jump Width is set to its maximum value, which is the minimum of 4 and Phase_Seg1.

If more than one configurations are possible to reach a certain Bit rate, it is recommended to choose the configuration that allows the highest oscillator tolerance range.

CAN nodes with different clocks require different configurations to come to the same bit rate. The calculation of the propagation time in the CAN network, based on the nodes with the longest delay times, is done once for the whole network.

The CAN system’s oscillator tolerance range is limited by the node with the lowest tolerance range.

The calculation may show that bus length or bit rate have to be decreased or that the oscillator frequencies’ stability has to be increased in order to find a protocol compliant configuration of the CAN bit timing.

The resulting configuration is written into the Bit Timing Register:

Tseg2 = Phase_Seg2-1
Tseg1 = Phase_Seg1+ Prop_Seg-1
SJW = SynchronizationJumpWidth-1
BRP = Prescaler-1

26.3.2.2 Example for Bit Timing at High Baudrate

In this example, the frequency of CAN_CLK is 10 MHz, BRP is 0, the bit rate is 1 MBit/s.

t_q	100	ns	=	t_{CAN_CLK}	
delay of bus driver	60	ns			
delay of receiver circuit	40	ns			
delay of bus line (40m)	220	ns			
t_{Prop}	700	ns	=	INT (2 × delays + 1) = 7 × t_q	
t_{SJW}	100	ns	=	1 × t_q	
t_{TSeg1}	800	ns	=	$t_{Prop} + t_{SJW}$	
t_{TSeg2}	100	ns	=	Information Processing Time + 1 × t_q	
$t_{Sync\text{-}Seg}$	100	ns	=	1 × t_q	
bit time	1000	ns	=	$t_{Sync\text{-}Seg} + t_{TSeg1} + t_{TSeg2}$	
tolerance for CAN_CLK	1.58	%	=	$\frac{\text{min(TSeg1, TSeg2)}}{\text{[2 x (13 x bit\_time - TSeg2)]}}$	(38)
			=	$\frac{0.1\mu s}{[2 \text{ x } (13 \text{ x } 1\mu s - 0.1\mu s)]}$	(39)
			=	0.38%	

In this example, the concatenated bit time parameters are $(1\text{-}1)_3$ & $(8\text{-}1)_4$ & $(1\text{-}1)_2$ & $(1\text{-}1)_6$, so the Bit Timing Register is programmed to 0000 0700h.

26.3.2.3 Example for Bit Timing at Low Baudrate

In this example, the frequency of CAN_CLK is 2 MHz, BRP is 1, the bit rate is 100 KBit/s.

t_q	1	µs	=	2 × t_{CAN_CLK}	
delay of bus driver	200	ns			
delay of receiver circuit	80	ns			
delay of bus line (40m)	220	ns			
t_{Prop}	1	µs	=	1 × t_q	
t_{SJW}	4	µs	=	4 × t_q	
t_{TSeg1}	5	µs	=	$t_{Prop} + t_{SJW}$	
t_{TSeg2}	3	µs	=	Information Processing Time + 3 × t_q	
$t_{Sync\text{-}Seg}$	1	µs	=	1 × t_q	
bit time	9	µs	=	$t_{Sync\text{-}Seg} + t_{TSeg1} + t_{TSeg2}$	
tolerance for CAN_CLK	0.43	%	=	$\frac{\text{min(TSeg1, TSeg2)}}{\text{[2 x (13 x bit\_time - TSeg2)]}}$	(40)
			=	$\frac{3\mu s}{[2 \text{ x } (13 \text{ x } 9\mu s - 3\mu s)]}$	(41)
			=	1.32%	

In this example, the concatenated bit time parameters are $(3\text{-}1)_3$ & $(5\text{-}1)_4$ & $(4\text{-}1)_2$ & $(2\text{-}1)_6$, so the Bit Timing Register is programmed to 0000 24C1h.

26.4 CAN Module Configuration

After a hardware reset all CAN protocol functions are disabled. The CAN module must be initialized and configured before it can participate on the CAN bus.

26.4.1 DCAN RAM Initialization through Hardware

To start with a clean DCAN RAM ,the complete DCAN RAM can be initialized with zeros and the ecc/parity bits set accordingly by configuring the following registers in the system module:

1. Memory Hardware Initialization Global Control Register (MINITGCR)
2. Memory Initialization Enable Register (MSINENA)

For more details on RAM hardware initialization support, refer to the system module reference guide.

26.4.2 CAN Module Initialization

To initialize the CAN Controller, you have to set up the CAN Bit timing and those message objects that have to be used for CAN communication. Message objects that are not needed, can be deactivated.

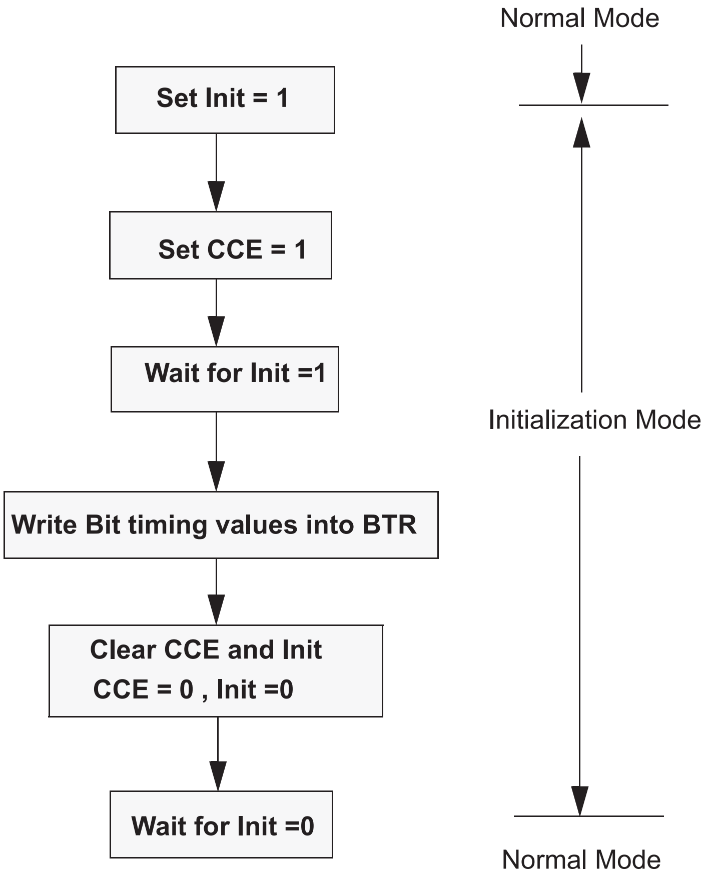
So the two critical steps are:

1. Configuration of CAN Bit Timings
2. Configuration of Message Objects

26.4.2.1 Software Configuration of CAN Bit Timings

This step involves configuring the CAN baud rate register with the calculated CAN bit timing value. The calculation procedure of CAN bit timing values for BTR register are mentioned in Section 26.3. Refer to Figure 26-3 for CAN bit timing software configuration flow.

Figure 26-3. CAN Bit-timing Configuration



Step 1: Enter “initialization mode” by setting the Init (Initialization) bit in the CAN Control Register.

While the Init bit is set, the message transfer from and to the CAN bus is stopped, and the status of the CAN_TX output is recessive (high).

The CAN error counters are not updated. Setting the Init bit does not change any other configuration register.

Also note that the CAN module is also in initialization mode on hardware reset and during Bus-Off.

Step 2: Set the CCE (Configure Change Enable) bit in the CAN Control Register.

The access to the Bit Timing Register for the configuration of the Bit timing is enabled when both Init and CCE bits in the CAN Control Register are set.

Step 3: Wait for the Init bit to get set. This would make sure that the module has entered “Initialization mode”.

Step 4: Write the Bit-Timing values into the Bit-Timing Register (BTR).

Refer to Section 26.3.2.1 for BTR value calculation for a given bit-timing.

Step 5: Clear the CCE bit followed by Init bit.

Step 6: Wait for the Init bit to clear. This would make sure that the module has come out of “initialization mode”.

After step 6 (Init bit cleared), the module will attempt a synchronization on the CAN bus, provided that the BTR settings are meeting the CAN bus parameters.

NOTE: The module would not come out of the “initialization mode” if any incorrect BTR values are written in step 4.

26.4.2.2 Configuration of Message Objects

The whole Message RAM should be configured before putting the CAN into operation. All the message objects are deactivated by default. The user should configure the message object that are to be used to a particular identifier. The user can change the configuration of any message object or deactivate it when required.

The message objects can be configured only through the Interface registers (IFx) and the CPU does not have direct access to the message object (Message RAM) when DCAN is in operation.

To configure the message objects, the user must know about:

1. The message object structure (Section 26.5)
2. The interface register set (IFx) (Section 26.6)

NOTE: The message objects initialization is independent of the bit-timing configuration procedure.

26.5 Message RAM

The DCAN Message RAM contains message objects and parity bits for the message objects.

26.5.1 Structure of Message Objects

Figure 26-4 shows the structure of a message object.

The grayed fields are those parts of the message object that are represented in dedicated registers. For example, the transmit request flags of all message objects are represented in centralized transmit request registers.

Figure 26-4. Structure of a Message Object

Message Object												
UMask	Msk[28:0]	MXtd	MDir	EoB	unused	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
MsgVal	ID[28:0]	Xtd	Dir	DLC[3:0]	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7

Table 26-2. Message Object Field Descriptions

Name	Value	Description
MsgVal		Message valid
	0	The message object is ignored by the Message Handler.
	1	The message object is to be used by the Message Handler.
		Note: The CPU should reset the MsgVal bit of all unused Messages Objects during the initialization before it resets bit Init in the CAN Control Register. MsgVal must also be reset if the messages object is no longer used in operation. For reconfiguration of message objects during normal operation see Section 26.7.6 and Section 26.7.7.
UMask		Use Acceptance Mask
	0	Mask bits (Msk[28:0], MXtd and MDir) are ignored and not used for acceptance filtering.
	1	Mask bits are used for acceptance filtering.
		Note: If the UMask bit is set to 1, the message object's mask bits have to be programmed during initialization of the message object before MsgVal is set to 1.
ID[28:0]		Message Identifier
	ID[28:0]	29-bit ("extended") identifier bits
	ID[28:18]	11-bit ("standard") identifier bits
Msk[28:0]		Identifier Mask
	0	The corresponding bit in the message identifier is not used for acceptance filtering (don't care).
	1	The corresponding bit in the message identifier is used for acceptance filtering.
Xtd		Extended Identifier
	0	The 11-bit ("standard") identifier will be used for this message object.
	1	The 29-bit ("extended") identifier will be used for this message object.
MXtd		Mask Extended Identifier
	0	The extended identifier bit (IDE) has no effect on the acceptance filtering.
	1	The extended identifier bit (IDE) is used for acceptance filtering.
		Note: When 11-bit ("standard") identifiers are used for a message object, the identifiers of received data frames are written into bits ID[28:18]. For acceptance filtering, only these bits together with mask bits Msk[28:18] are considered.
Dir		Message Direction
	0	Direction = receive: On TxRqst, a remote frame with the identifier of this message object is transmitted. On reception of a data frame with matching identifier, the message is stored in this message object.
	1	Direction = transmit: On TxRqst, a data frame is transmitted. On reception of a remote frame with matching identifier, the TxRqst bit of this message object is set (if RmtEn = 1).
MDir		Mask Message Direction
	0	The message direction bit (Dir) has no effect on the acceptance filtering.
	1	The message direction bit (Dir) is used for acceptance filtering.

Table 26-2. Message Object Field Descriptions (continued)

Name	Value	Description
EOB		End of Block
	0	The message object is part of a FIFO Buffer block and is not the last message object of this FIFO Buffer block.
	1	The message object is a single message object or the last message object in a FIFO Buffer Block.
		Note: This bit is used to concatenate multiple message objects to build a FIFO Buffer. For single message objects (not belonging to a FIFO Buffer), this bit must always be set to 1.
NewDat		New Data
	0	No new data has been written into the data bytes of this message object by the Message Handler since the last time when this flag was cleared by the CPU.
	1	The Message Handler or the CPU has written new data into the data bytes of this message object.
MsgLst		Message Lost (only valid for message objects with direction = receive)
	0	No message was lost since the last time when this bit was reset by the CPU.
	1	The Message Handler stored a new message into this message object when NewDat was still set, so the previous message has been overwritten.
RxIE		Receive Interrupt Enable
	0	IntPnd will not be triggered after the successful reception of a frame.
	1	IntPnd will be triggered after the successful reception of a frame.
TxIE		Transmit Interrupt Enable
	0	IntPnd will not be triggered after the successful transmission of a frame.
	1	IntPnd will be triggered after the successful transmission of a frame.
IntPnd		Interrupt Pending
	0	This message object is not the source of an interrupt.
	1	This message object is the source of an interrupt. The Interrupt Identifier in the Interrupt Register will point to this message object if there is no other interrupt source with higher priority.
RmtEn		Remote Enable
	0	At the reception of a Remote Frame, TxRqst is not changed.
	1	At the reception of a Remote Frame, TxRqst is set.
TxRqst		Transmit Request
	0	This message object is not waiting for a transmission.
	1	The transmission of this message object is requested and is not yet done.
DLC[3:0]		Data Length Code
	0-8	Data Frame has 0-8 data bytes.
	9-15	Data Frame has 8 data bytes.
		Note: The Data Length Code of a message object must be defined to the same value as in the corresponding objects with the same identifier at other nodes. When the Message Handler stores a data frame, it will write the DLC to the value given by the received message.
Data 0		1st data byte of a CAN Data Frame
Data 1		2nd data byte of a CAN Data Frame
Data 2		3rd data byte of a CAN Data Frame
Data 3		4th data byte of a CAN Data Frame
Data 4		5th data byte of a CAN Data Frame
Data 5		6th data byte of a CAN Data Frame
Data 6		7th data byte of a CAN Data Frame
Data 7		8th data byte of a CAN Data Frame
		Note: Byte Data 0 is the first data byte shifted into the shift register of the CAN Core during a reception, byte Data 7 is the last. When the Message Handler stores a data frame, it will write all the eight data bytes into a message object. If the Data Length Code is less than 8, the remaining bytes of the message object may be overwritten by undefined values.

26.5.2 Addressing Message Objects in RAM

The starting location of a particular message object in RAM is:

Message RAM base address + (message object number) × 0x20.

This means that Message Object 1 starts at offset 0x0020; Message Object 2 starts at offset 0x0040, and so on.

NOTE: 0 is not a valid message object number. At address 0x0000, message object number 64 is located. Writing to the address of an unimplemented message object may overwrite an implemented message object.

The base address for DCAN1 RAM is FF1E 0000h, for DCAN2 RAM is FF1C 0000h, and for DCAN3 RAM is FF1A 0000h.

Message Object number 1 has the highest priority.

Table 26-3. Message RAM Addressing in Debug/Suspend and RDA Mode

Message Object Number	Base Address Offset	Word Number	Debug/Suspend mode, see Section 26.5.3	RDA mode, see Section 26.5.4
1	0x0020	1	Parity	Data Bytes 4-7
	0x0024	2	MXtd, MDir, Mask	Data Bytes 0-3
	0x0028	3	Xtd, Dir, ID	ID[27:0], DLC
	0x002C	4	Ctrl	Mask, Xtd, Dir, ID[28]
	0x0030	5	Data Bytes 3-0	Parity, Ctrl, MXtd, MDir
	0x0034	6	Data Bytes 7-4	--
:	:	:	:	:
31	0x03E0	1	Parity	Data Bytes 4-7
	0x03E4	2	MXtd, MDir, Mask	Data Bytes 0-3
	0x03E8	3	Xtd, Dir, ID	ID[27]:0, DLC
	0x03EC	4	Ctrl	Mask, Xtd, Dir, ID[28]
	0x03F0	5	Data Bytes 3-0	Parity, Ctrl, MXtd, MDir
	0x03F4	6	Data Bytes 7-4	--
:	:	:	:	:
63	0x07E0	1	Parity	Data Bytes 4-7
	0x07E4	2	MXtd, MDir, Mask	Data Bytes 0-3
	0x07E8	3	Xtd, Dir, ID	ID[27:0], DLC
	0x07EC	4	Ctrl	Mask, Xtd, Dir, ID[28]
	0x07F0	5	Data Bytes 3-0	Parity, Ctrl, MXtd, MDir
	0x07F4	6	Data Bytes 7-4	--
64	0x0000	1	Parity	Data Bytes 4-7
	0x0004	2	MXtd, MDir, Mask	Data Bytes 0-3
	0x0008	3	Xtd, Dir, ID	ID[27]:0, DLC
	0x000C	4	Ctrl	Mask, Xtd, Dir, ID[28]
	0x0010	5	Data Bytes 3-0	Parity, Ctrl, MXtd, MDir
	0x0014	6	Data Bytes 7-4	--

26.5.3 Message RAM Representation in Debug/Suspend Mode

In Debug/Suspend mode, the Message RAM will be memory mapped. This allows the external debug unit to access the Message RAM.

NOTE: During Debug/Suspend Mode, the Message RAM cannot be accessed via the IFx register sets.

Figure 26-5. Message RAM Representation in Debug/Suspend Mode

<table>
<tr><th>Bit</th><th>31
15</th><th>30
14</th><th>29
13</th><th>28
12</th><th>27
11</th><th>26
10</th><th>25
9</th><th>24
8</th><th>23
7</th><th>22
6</th><th>21
5</th><th>20
4</th><th>19
3</th><th>18
2</th><th>17
1</th><th>16
0</th></tr>
<tr><td rowspan="2">MsgAddr + 0x00</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="11">Reserved</td><td colspan="5">Parity[4:0]</td></tr>
<tr><td rowspan="2">MsgAddr + 0x04</td><td>MXtd</td><td>MDir</td><td>Rsvd</td><td colspan="13">Msk[28:16]</td></tr>
<tr><td colspan="16">Msk[15:0]</td></tr>
<tr><td rowspan="2">MsgAddr + 0x08</td><td>Rsvd</td><td>Xtd</td><td>Dir</td><td colspan="13">ID[28:16]</td></tr>
<tr><td colspan="16">ID[15:0]</td></tr>
<tr><td rowspan="2">MsgAddr + 0x0C</td><td colspan="16">Reserved</td></tr>
<tr><td>Rsvd</td><td>MsgLst</td><td>Rsvd</td><td>UMask</td><td>TxIE</td><td>RxIE</td><td>RmtEn</td><td>Rsvd</td><td>EOB</td><td colspan="3">Reserved</td><td colspan="4">DLC[3:0]</td></tr>
<tr><td rowspan="2">MsgAddr + 0x10</td><td colspan="8">Data 3</td><td colspan="8">Data 2</td></tr>
<tr><td colspan="8">Data 1</td><td colspan="8">Data 0</td></tr>
<tr><td rowspan="2">MsgAddr + 0x14</td><td colspan="8">Data 7</td><td colspan="8">Data 6</td></tr>
<tr><td colspan="8">Data 5</td><td colspan="8">Data 4</td></tr>
</table>

26.5.4 Message RAM Representation in Direct Access Mode

When the RDA bit in Test Register is set while the DCAN module is in Test Mode (Test bit in CAN control register is set), the CPU has direct access to the Message RAM. Due to the 32-bit bus structure, the RAM is split into word lines to support this feature. The CPU has access to one word line at a time only.

In RAM Direct Access mode, the RAM is represented by a continuous memory space within the address frame of the DCAN module, starting at the Message RAM base address.

NOTE: During Direct Access Mode, the Message RAM cannot be accessed via the IFx register sets. Before entering RDA mode, it must be ensured that the Init bit is set to avoid any conflicts with the message handler accessing the message RAM.

Any read or write to the RAM addresses for RamDirectAccess during normal operation mode (TestMode bit or RDA bit is not set) will be ignored.

Writes to Reserved bits have no effect.

Figure 26-6. Message RAM Representation in RAM Direct Access Mode

<table>
<tr><th>Bit</th><th>31
15</th><th>30
14</th><th>29
13</th><th>28
12</th><th>27
11</th><th>26
10</th><th>25
9</th><th>24
8</th><th>23
7</th><th>22
6</th><th>21
5</th><th>20
4</th><th>19
3</th><th>18
2</th><th>17
1</th><th>16
0</th></tr>
<tr><td rowspan="2">MsgAddr + 0x00</td><td colspan="8">Data 4</td><td colspan="8">Data 5</td></tr>
<tr><td colspan="8">Data 6</td><td colspan="8">Data 7</td></tr>
<tr><td rowspan="2">MsgAddr + 0x04</td><td colspan="8">Data 0</td><td colspan="8">Data 1</td></tr>
<tr><td colspan="8">Data 2</td><td colspan="8">Data 3</td></tr>
<tr><td rowspan="2">MsgAddr + 0x08</td><td colspan="16">ID[27:12]</td></tr>
<tr><td colspan="12">ID[11:0]</td><td colspan="4">DLC[3:0]</td></tr>
<tr><td rowspan="2">MsgAddr + 0x0C</td><td colspan="16">Msk[28:13]</td></tr>
<tr><td colspan="13">Msk[12:0]</td><td>Xtd</td><td>Dir</td><td>ID[28]</td></tr>
<tr><td rowspan="2">MsgAddr + 0x10</td><td colspan="16">Reserved</td></tr>
<tr><td colspan="4">Parity[3:0]</td><td colspan="4">Reserved</td><td>MsgLst</td><td>UMask</td><td>TxIE</td><td>RxIE</td><td>RmtEn</td><td>EOB</td><td>MXtd</td><td>MDir</td></tr>
</table>

26.6 Message Interface Register Sets

Accesses to the Message RAM are performed via the Interface Register sets.

1. Interface Register 1 and 2
2. Interface Register 3

The third Interface Register set IF3 can be configured to automatically receive control and user data from the Message RAM when a message object has been updated after reception of a CAN message. The CPU does not need to initiate the transfer from Message RAM to IF3 Register set.

The Message Handler avoids potential conflicts between concurrent accesses to Message RAM and CAN frame reception/transmission.

There are two modes where the Message RAM can be directly accessed by the CPU:

1. In Debug/Suspend mode (see Section 26.5.3)
2. In RAM Direct Access (RDA) mode (see Section 26.5.4)

For the Message RAM Base address, please refer to the device datasheet.

A complete message object (see Section 26.5.1) or parts of the message object may be transferred between the Message RAM and the IF1/IF2 Register set (see Section 26.17.20) in one single transfer.

26.6.1 Message Interface Register Sets 1 and 2

The Interface Register sets IF1 and IF2 provide indirect read/write access from the CPU to the Message RAM. The IF1 and IF2 register sets can buffer control and user data to be transferred to and from the message objects.

Table 26-4. Message Interface Register Sets 1 and 2

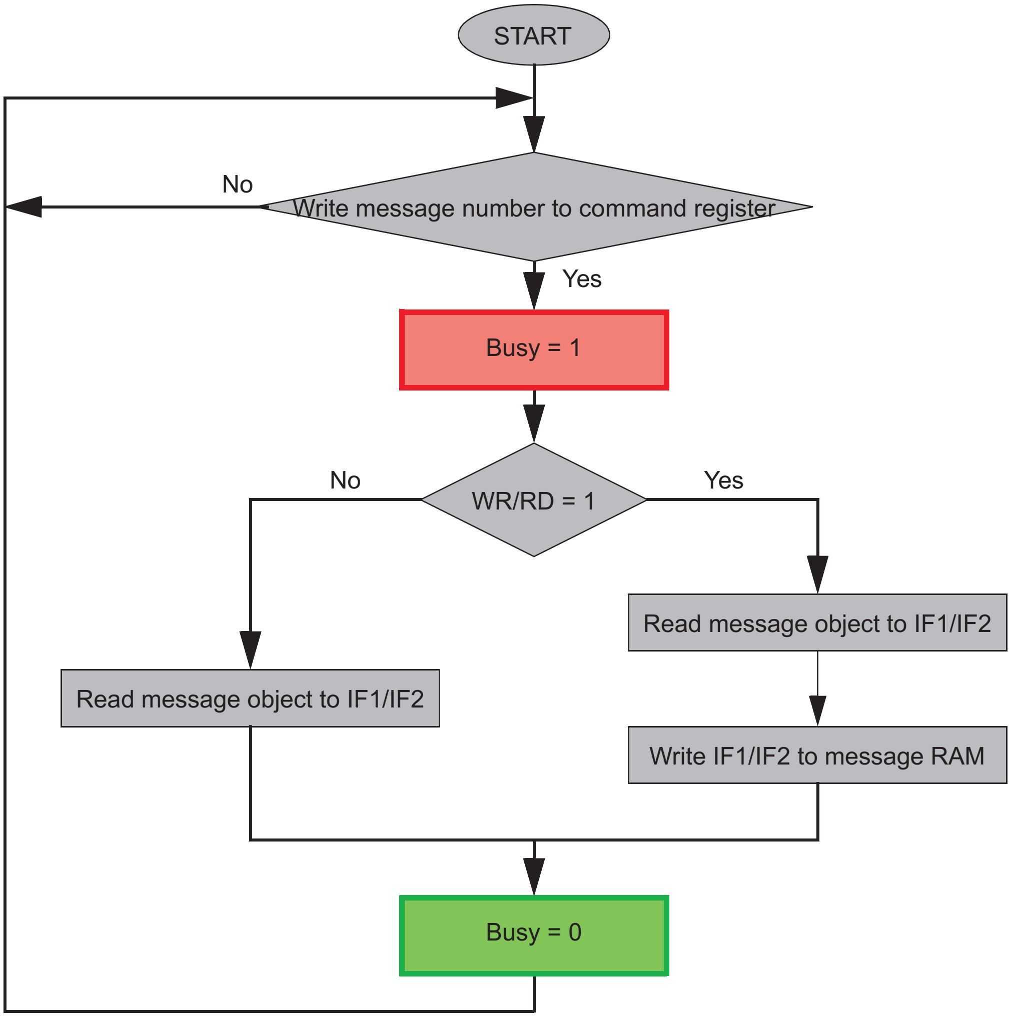
Address	IF1 Register Set		Address	IF2 Register Set	
[CAN Base +]	31 16	15 0	[CAN Base +]	31 16	15 0
0x100	IF1 Command Mask	IF1 Command Request	0x120	IF2 Command Mask	IF2 Command Request
0x104	IF1 Mask 2	IF1 Mask 1	0x124	IF2 Mask 2	IF2 Mask 1
0x108	IF1 Arbitration 2	IF1 Arbitration 1	0x128	IF2 Arbitration 2	IF2 Arbitration 1
0x10C	Rsvd	IF1 Message Control	0x12C	Rsvd	IF2 Message Control
0x110	IF1 Data A 2	IF1 Data A 1	0x130	IF2 Data A 2	IF2 Data A 1
0x114	IF1 Data B 2	IF1 Data B 1	0x134	IF2 Data B 2	IF2 Data B 1

26.6.2 Using Message Interface Register Sets 1 and 2

The Command Register addresses the desired message object in the Message RAM and specifies whether a complete message object or only parts should be transferred. The data transfer is initiated by writing the message number to the bits [7:0] of the Command Register.

When the CPU initiates a data transfer between the IF1/IF2 Registers and Message RAM, the Message Handler sets the Busy bit in the respective Command Register to 1. After the transfer has completed, the Busy bit is set back to 0 (see Figure 26-7).

Figure 26-7. Data Transfer Between IF1 / IF2 Registers and Message RAM


START
No
Write message number to command register
Yes
Busy = 1
No
WR/RD = 1
Yes
Read message object to IF1/IF2
Read message object to IF1/IF2
Write IF1/IF2 to message RAM
Busy = 0

26.6.3 Message Interface Register 3

The IF3 register set can automatically be updated with received message objects without the need to initiate the transfer from Message RAM by CPU. The intention of this feature of IF3 is to provide an interface for the DMA to read packets efficiently.

Table 26-5. Message Interface Register 3

Address	**IF3 Register Set**	
[CAN Base +]	**31 16**	**15 0**
0x140	reserved	IF3 Observation
0x144	IF3 Mask 2	IF3 Mask 1
0x148	IF3 Arbitration 2	IF3 Arbitration 1
0x14C	reserved	IF3 Message Control
0x150	IF3 Data A 2	IF3 Data A 1
0x154	IF3 Data B 2	IF3 Data B 1
...		
0x160	IF3 Update Enable 2	IF3 Update Enable 1
0x164	IF3 Update Enable 4	IF3 Update Enable 3
0x168	IF3 Update Enable 6	IF3 Update Enable 5
0x16C	IF3 Update Enable 8	IF3 Update Enable 7

The automatic update functionality can be programmed for each message object (see IF3 Update Enable Register, Section 26.17.29).

All valid message objects in Message RAM that are configured for automatic update, will be checked for active NewDat flags. If such a message object is found, it will be transferred to the IF3 register (if no previous DMA transfers are ongoing), controlled by IF3 Observation register. If more than one NewDat flag is active, the message object with the lowest number has the highest priority for automatic IF3 update.

The NewDat bit in the message object will be reset by a transfer to IF3.

If DCAN internal IF3 update is complete, a DMA request is generated. The DMA request stays active until first read access to one of the IF3 registers. The DMA functionality has to be enabled by setting bit DE3 in CAN Control register. Please refer to the device datasheet to find out if this DMA source is available.

NOTE: The IF3 register set can not be used for transferring data into message objects.

26.7 Message Object Configurations

This section describes the possible message object configurations for CAN communication.

26.7.1 Configuration of a Transmit Object for Data Frames

Figure 26-8 shows how a Transmit Object can be initialized.

Figure 26-8. Initialization of a Transmit Object

MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	1	0	0	0	appl.	0	appl.	0

The Arbitration bits (ID[28:0] and Xtd bit) are given by the application. They define the identifier and type of the outgoing message. If an 11-bit Identifier (Standard Frame) is used (Xtd = 0), it is programmed to ID[28:18]. In this case, ID[17:0] can be ignored.

The Data Registers (DLC and Data0-7) are given by the application, TxRqst and RmtEn should not be set before the data is valid.

If the TxIE bit is set, the IntPnd bit will be set after a successful transmission of the message object.

If the RmtEn bit is set, a matching received Remote Frame will cause the TxRqst bit to be set; the Remote Frame will autonomously be answered by a Data Frame.

The Mask bits (Msk[28:0], UMask, MXtd, and MDir bits) may be used (UMask= 1) to allow groups of Remote Frames with similar identifiers to set the TxRqst bit. The Dir bit should not be masked. For details, see Section 26.8.8. Identifier masking must be disabled (UMask = 0) if no Remote Frames are allowed to set the TxRqst bit (RmtEn = 0).

26.7.2 Configuration of a Transmit Object for Remote Frames

It is not necessary to configure Transmit Objects for the transmission of Remote Frames. Setting TxRqst for a Receive Object will cause the transmission of a Remote Frame with the same identifier as the Data Frame for which this receive Object is configured.

26.7.3 Configuration of a Single Receive Object for Data Frames

Figure 26-9 shows how a Receive Object for Data Frames can be initialized.

Figure 26-9. Initialization of a Single Receive Object for Data Frames

MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	0	0	0	appl.	0	0	0	0

The Arbitration bits (ID[28:0] and Xtd bit) are given by the application. They define the identifier and type of accepted received messages. If an 11-bit Identifier (Standard Frame) is used (Xtd = 0), it is programmed to ID[28:18]. In this case, ID[17:0] can be ignored. When a Data Frame with an 11-bit Identifier is received, ID[17:0] will be set to 0.

The Data Length Code (DLC) is given by the application. When the Message Handler stores a Data Frame in the message object, it will store the received Data Length Code and eight data bytes. If the Data Length Code is less than 8, the remaining bytes of the message object may be overwritten by non specified values.

The Mask bits (Msk[28:0], UMask, MXtd, and MDir bits) may be used (UMask = 1) to allow groups of Data Frames with similar identifiers to be accepted. The Dir bit should not be masked in typical applications. If some bits of the Mask bits are set to “don’t care”, the corresponding bits of the Arbitration Register will be overwritten by the bits of the stored Data Frame.

If the RxIE bit is set, the IntPnd bit will be set when a received Data Frame is accepted and stored in the message object.

If the TxRqst bit is set, the transmission of a Remote Frame with the same identifier as actually stored in the Arbitration bits will be triggered. The content of the Arbitration bits may change if the Mask bits are used (UMask = 1) for acceptance filtering.

26.7.4 Configuration of a Single Receive Object for Remote Frames

Figure 26-10 shows how a Receive Object for Remote Frames can be initialized.

Figure 26-10. Initialization of a Single Receive Object for Remote Frames

MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	1	0	0	appl.	0	0	0	0

Receive Objects for Remote Frames may be used to monitor Remote Frames on the CAN bus. The Remote Frame stored in the Receive Object will not trigger the transmission of a Data Frame. Receive Objects for Remote Frames may be expanded to a FIFO buffer, see Section 26.7.5.

UMask must be set to 1. The Mask bits (Msk[28:0], UMask, MXtd, and MDir bits) may be set to “must-match” or to “don’t care”, to allow groups of Remote Frames with similar identifiers to be accepted. The Dir bit should not be masked in typical applications. For details, see Section 26.8.8.

The Arbitration bits (ID[28:0] and Xtd bit) may be given by the application. They define the identifier and type of accepted received Remote Frames. If some bits of the Mask bits are set to “don’t care”, the corresponding bits of the Arbitration bits will be overwritten by the bits of the stored Remote Frame. If an 11-bit Identifier (Standard Frame) is used (Xtd = 0), it is programmed to ID[28:18]. In this case, ID[17:0] can be ignored. When a Remote Frame with an 11-bit Identifier is received, ID[17:0] will be set to 0.

The Data Length Code (DLC) may be given by the application. When the Message Handler stores a Remote Frame in the message object, it will store the received Data Length Code. The data bytes of the message object will remain unchanged.

If the RxIE bit is set, the IntPnd bit will be set when a received Remote Frame is accepted and stored in the message object.

26.7.5 Configuration of a FIFO Buffer

With the exception of the EoB bit, the configuration of Receive Objects belonging to a FIFO Buffer is the same as the configuration of a single Receive Object.

To concatenate multiple message objects to a FIFO Buffer, the identifiers and masks (if used) of these message objects have to be programmed to matching values. Due to the implicit priority of the message objects, the message object with the lowest number will be the first message object of the FIFO Buffer. The EoB bit of all message objects of a FIFO Buffer except the last one have to be programmed to zero. The EoB bits of the last message object of a FIFO Buffer is set to 1, configuring it as the end of the block.

26.7.6 Reconfiguration of Message Objects for the Reception of Frames

A message object with Dir = 0 is configured for the reception of data frames, with Dir = 1 AND Umask = 1 AND RmtEn = 0 it is configured for the reception of remote frames.

It is necessary to reset MsgVal to not valid before changing any of the following configuration and control bits: ID[28:0], Xtd, Dir, DLC, RxIE, TxIE, RmtEn, EoB, Umask, Msk[28:0], MXtd, and MDir.

These parts of a message object may be changed without clearing MsgVal: Data[7:0], TxRqst, NewDat, MsgLst, and IntPnd.

26.7.7 Reconfiguration of Message Objects for the Transmission of Frames

A message object with Dir = 1 AND (Umask = 0 OR RmtEn = 1) is configured for the transmission of data frames.

It is necessary to reset MsgVal to not valid before changing any of the following configuration and control bits: Dir, RxIE, TxIE, RmtEn, EoB, Umask, Msk[28:0], MXtd, and MDir.

These parts of a message object may be changed without clearing MsgVal: ID[28-0], Xtd, DLC, Data[7:0], TxRqst, NewDat, MsgLst, and IntPnd.

26.8 Message Handling

When initialization is finished, the DCAN module synchronizes itself to the traffic on the CAN bus. It does acceptance filtering on received messages and stores those frames that are accepted into the designated message objects.

The application has to update the data of the messages to be transmitted and enable and request their transmission. The transmission is requested automatically when a matching Remote Frame is received.

The application may read messages that are received and accepted. Messages that are not read before the next messages is accepted for the same message object will be overwritten.

Messages may be read based on interrupts or by polling.

26.8.1 Message Handler Overview

The Message Handler state machine controls the data transfer between the Rx/Tx Shift Register of the CAN Core and the Message RAM. It performs the following tasks:

- Data Transfer from Message RAM to CAN Core (messages to be transmitted).
- Data Transfer from CAN Core to the Message RAM (received messages).
- Data Transfer from CAN Core to the Acceptance Filtering unit.
- Scanning of Message RAM for a matching message object (acceptance filtering).
- Scanning the same message object after being changed by IF1/IF2 registers when priority is same or higher as message the object found by last scanning.
- Handling of TxRqst flags.
- Handling of interrupt flags.

The Message Handler registers contains status flags of all message objects grouped into the following topics:

- Transmission Request flags
- New Data flags
- Interrupt Pending Flags
- Message Valid Registers

Instead of collecting the listed status information of each message object via IFx registers separately, these Message Handler registers provides a fast and easy way to get an overview (for example, about all pending transmission requests).

All Message Handler registers are read-only.

26.8.2 Receive/Transmit Priority

The receive/transmit priority for the message objects is attached to the message number, not to the CAN identifier. Message object 1 has the highest priority, while the last implemented message object has the lowest priority. If more than one transmission request is pending, they are serviced due to the priority of the corresponding message object, so messages with the highest priority can be placed in the message objects with the lowest numbers.

The acceptance filtering for received Data Frames or Remote Frames is also done in ascending order of message objects, so a frame that has been accepted by a message object cannot be accepted by another message object with a higher Message Number. The last message object may be configured to accept any Data Frame or Remote Frame that was not accepted by any other message object, for nodes that need to log the complete message traffic on the CAN bus.

26.8.3 Transmission of Messages in Event Driven CAN Communication

If the shift register of the CAN Core is ready for loading and if there is no data transfer between the IFx Registers and Message RAM, the d bits in the Message Valid Register and the TxRqst bits in the Transmission Request Register are evaluated. The valid message object with the highest priority pending transmission request is loaded into the shift register by the Message Handler and the transmission is started. The message object's NewDat bit is reset.

After a successful transmission and if no new data was written to the message object (NewDat = 0) since the start of the transmission, the TxRqst bit will be reset. If TxIE is set, IntPnd will be set after a successful transmission. If the DCAN has lost the arbitration or if an error occurred during the transmission, the message will be retransmitted as soon as the CAN bus is free again. If meanwhile the transmission of a message with higher priority has been requested, the messages will be transmitted in the order of their priority.

If Automatic Retransmission mode is disabled by setting the DAR bit in the CAN Control Register, the behavior of bits TxRqst and NewDat in the Message Control Register of the Interface Register set is as follows:

- When a transmission starts, the TxRqst bit of the respective Interface Register set is reset, while bit NewDat remains set.
- When the transmission has been successfully completed, the NewDat bit is reset.

When a transmission failed (lost arbitration or error) bit NewDat remains set. To restart the transmission, the application has to set TxRqst again.

Received Remote Frames do not require a Receive Object. They will automatically trigger the transmission of a Data Frame, if in the matching Transmit Object the RmtEn bit is set.

26.8.4 Updating a Transmit Object

The CPU may update the data bytes of a Transmit Object any time via the IF1/IF2 Interface Registers, neither d nor TxRqst have to be reset before the update.

Even if only a part of the data bytes are to be updated, all four bytes in the corresponding IF1/IF2 Data A Register or IF1/IF2 Data B Register have to be valid before the content of that register is transferred to the message object. Either the CPU has to write all four bytes into the IF1/IF2 Data Register or the message object is transferred to the IF1/IF2 Data Register before the CPU writes the new data bytes.

When only the data bytes are updated, first 0x87 can be written to bits [23:16] of the Command Register and then the number of the message object is written to bits [7:0] of the Command Register, concurrently updating the data bytes and setting TxRqst with NewDat.

To prevent the reset of TxRqst at the end of a transmission that may already be in progress while the data is updated, NewDat has to be set together with TxRqst in event driven CAN communication. For details, see Section 26.8.3.

When NewDat is set together with TxRqst, NewDat will be reset as soon as the new transmission has started.

26.8.5 Changing a Transmit Object

If the number of implemented message objects is not sufficient to be used as permanent message objects only, the Transmit Objects may be managed dynamically. The CPU can write the whole message (Arbitration, Control, and Data) into the Interface Register. The bits [23:16] of the Command Register can be set to 0xB7 for the transfer of the whole message object content into the message object. Before changing the configuration of a message object, MsgVal has to be reset (see Section 26.7.7).

If a previously requested transmission of this message object is not completed but already in progress, it will be continued; however it will not be repeated if it is disturbed.

To only update the data bytes of a message to be transmitted, bits [23:16] of the Command Register should be set to 0x87.

NOTE: After the update of the Transmit Object, the Interface Register set will contain a copy of the actual contents of the object, including the part that had not been updated.

26.8.6 Acceptance Filtering of Received Messages

When the arbitration and control bits (Identifier + IDE + RTR + DLC) of an incoming message is completely shifted into the shift register of the CAN Core, the Message Handler starts to scan of the Message RAM for a matching valid message object:

- The Acceptance Filtering unit is loaded with the arbitration bits from the CAN Core shift register.
- Then the arbitration and mask bits (including MsgVal, UMask, NewDat, and EoB) of Message Object 1 are loaded into the Acceptance Filtering unit and are compared with the arbitration bits from the shift register. This is repeated for all following message objects until a matching message object is found, or until the end of the Message RAM is reached.
- If a match occurs, the scanning is stopped and the Message Handler proceeds depending on the type of the frame (Data Frame or Remote Frame) received.

26.8.7 Reception of Data Frames

The Message Handler stores the message from the CAN Core shift register into the respective message object in the Message RAM. Not only the data bytes, but all arbitration bits and the Data Length Code are stored into the corresponding message object. This ensures that the data bytes stay associated to the identifier even if arbitration mask registers are used.

The NewDat bit is set to indicate that new data (not yet seen by the CPU) has been received. The CPU should reset the NewDat bit when it reads the message object. If at the time of the reception the NewDat bit was already set, MsgLst is set to indicate that the previous data (supposedly not seen by the CPU) is lost. If the RxIE bit is set, the IntPnd bit is set, causing the Interrupt Register to point to this message object.

The TxRqst bit of this message object is reset to prevent the transmission of a Remote Frame, while the requested Data Frame has just been received.

26.8.8 Reception of Remote Frames

When a Remote Frame is received, three different configurations of the matching message object have to be considered:

1. Dir = 1 (direction = transmit), RmtEn = 1, UMask = 1 or 0. The TxRqst bit of this message object is set at the reception of a matching Remote Frame. The rest of the message object remains unchanged.
2. Dir = 1 (direction = transmit), RmtEn = 0, UMask = 0. The Remote Frame is ignored, this message object remains unchanged.
3. Dir = 1 (direction = transmit), RmtEn = 0, UMask = 1. The Remote Frame is treated similar to a received Data Frame. At the reception of a matching Remote Frame, the TxRqst bit of this message object is reset. The arbitration and control bits (Identifier + IDE + RTR + DLC) from the shift register are stored in the message object in the Message RAM and the NewDat bit of this message object is set. The data bytes of the message object remain unchanged.

26.8.9 Reading Received Messages

The CPU may read a received message any time via the IFx Interface Registers, the data consistency is guaranteed by the Message Handler state machine.

Typically the CPU will write first 0x7F to bits [23:16] and then the number of the message object to bits [7:0] of the Command Register. That combination will transfer the whole received message from the Message RAM into the Interface Register set. Additionally, the bits NewDat and IntPnd are cleared in the Message RAM (not in the Interface Register set). The values of these bits in the Message Control Register always reflect the status before resetting the bits.

If the message object uses masks for acceptance filtering, the arbitration bits show which of the different matching messages has been received.

The actual value of NewDat shows whether a new message has been received since last time when this message object was read. The actual value of MsgLst shows whether more than one message have been received since the last time when this message object was read. MsgLst will not be automatically reset.

26.8.10 Requesting New Data for a Receive Object

By means of a Remote Frame, the CPU may request another CAN node to provide new data for a receive object. Setting the TxRqst bit of a receive object will cause the transmission of a Remote Frame with the receive object's identifier. This Remote Frame triggers the other CAN node to start the transmission of the matching Data Frame. If the matching Data Frame is received before the Remote Frame could be transmitted, the TxRqst bit is automatically reset.

Setting the TxRqst bit without changing the contents of a message object requires the value 0x84 in bits [23:16] of the Command Register.

26.8.11 Storing Received Messages in FIFO Buffers

Several message objects may be grouped to form one or more FIFO Buffers. Each FIFO Buffer configured to store received messages with a particular (group of) Identifier(s). Arbitration and Mask Registers of the FIFO Buffer's message objects are identical. The EoB (End of Buffer) bits of all but the last of the FIFO Buffer's message objects are 0, in the last one the EoB bit is 1.

Received messages with identifiers matching to a FIFO Buffer are stored into a message object of this FIFO Buffer, starting with the message object with the lowest message number.

When a message is stored into a message object of a FIFO Buffer the NewDat bit of this message object is set. By setting NewDat while EoB is 0, the message object is locked for further write accesses by the Message Handler until the CPU has cleared the NewDat bit.

Messages are stored into a FIFO Buffer until the last message object of this FIFO Buffer is reached. If none of the preceding message objects is released by writing NewDat to 0, all further messages for this FIFO Buffer will be written into the last message object of the FIFO Buffer (EoB = 1) and therefore overwrite previous messages in this message object.

26.8.12 Reading from a FIFO Buffer

Several messages may be accumulated in a set of message objects that are concatenated to form a FIFO Buffer before the application program is required (in order to avoid the loss of data) to empty the buffer.

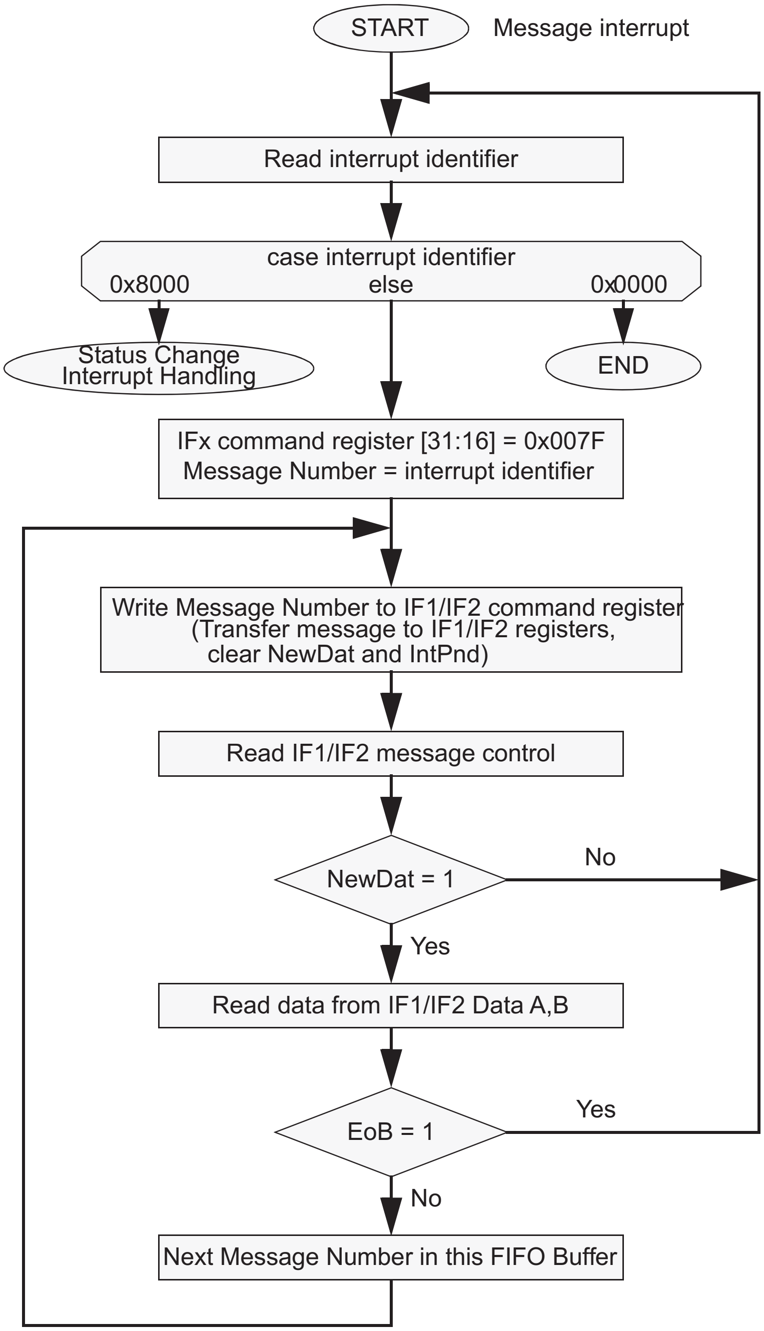
A FIFO Buffer of length N will store N-1 plus the last received message since last time it was cleared.

A FIFO Buffer is cleared by reading and resetting the NewDat bits of all its message objects, starting at the FIFO Object with the lowest message number. This should be done in a subroutine following the example shown in Figure 26-11.

NOTE: All message objects of a FIFO buffer needs to be read and cleared before the next batch of messages can be stored. Otherwise true FIFO functionality can not be guaranteed, since the message objects of a partly read buffer will be re-filled according to the normal (descending) priority.

Reading from a FIFO Buffer message object and resetting its NewDat bit is handled the same way as reading from a single message object.

Figure 26-11. CPU Handling of a FIFO Buffer (Interrupt Driven)


START
Message interrupt
Read interrupt identifier
case interrupt identifier
0x8000
else
0x0000
Status Change Interrupt Handling
END
IFx command register [31:16] = 0x007F
Message Number = interrupt identifier
Write Message Number to IF1/IF2 command register
(Transfer message to IF1/IF2 registers,
clear NewDat and IntPnd)
Read IF1/IF2 message control
NewDat = 1
No
Yes
Read data from IF1/IF2 Data A,B
EoB = 1
Yes
No
Next Message Number in this FIFO Buffer

26.9 CAN Message Transfer

Once the DCAN is initialized and the Init bit is reset to 0, the CAN Core synchronizes itself to the CAN bus and is ready for message transfer as per the configured message objects.

The CPU may enable the interrupt lines (setting IE0 and IE1 to 1) at the same time when it clears Init and CCE. The status interrupts EIE and SIE may be enabled simultaneously.

The CAN communication can be carried out in any of the following two modes:

1. Interrupt mode
2. Polling mode.

The Interrupt Register points to those message objects with IntPnd = 1. It is updated even if the interrupt lines to the CPU are disabled (IE0 and IE1 are 0).

The CPU may poll all Message Object’s NewDat and TxRqst bits in parallel from the NewData X Registers and the Transmission Request X Registers. Polling can be made easier if all Transmit Objects are grouped at the low numbers, all Receive Objects are grouped at the high numbers.

Received messages are stored into their appropriate message objects if they pass acceptance filtering.

The whole message (including all arbitration bits, DLC and up to eight data bytes) is stored into the message object. As a consequence, when the identifier mask is used, the arbitration bits that are masked to “don’t care” may change in the message object when a received message is stored.

The CPU may read or write each message at any time via the Interface Registers, as the Message Handler guarantees data consistency in case of concurrent accesses (for reconfiguration, see Section 26.7.6)

If a permanent message object (arbitration and control bits set up during configuration and leaving unchanged for multiple CAN transfers) exists for the message, it is possible to only update the data bytes.

If several transmit messages should be assigned to one message object, the whole message object has to be configured before the transmission of this message is requested.

The transmission of multiple message objects may be requested at the same time. They are subsequently transmitted, according to their internal priority.

Messages may be updated or set to not valid at any time, even if a requested transmission is still pending (for reconfiguration, see Section 26.7.7). However, the data bytes will be discarded if a message is updated before a pending transmission has started.

Depending on the configuration of the message object, a transmission may be automatically requested by the reception of a remote frame with a matching identifier.

26.9.1 Automatic Retransmission

According to the CAN Specification (ISO11898), the DCAN provides a mechanism to automatically retransmit frames that have lost arbitration or have been disturbed by errors during transmission. The frame transmission service will not be confirmed to the user before the transmission is successfully completed.

By default, this automatic retransmission is enabled. It can be disabled by setting bit DAR (Disable Automatic Retransmission) in CAN Control Register. Further details to this mode are provided in Section 26.8.3.

26.9.2 Auto-Bus-On

Per default, after the DCAN has entered Bus-Off state, the CPU can start a Bus-Off-Recovery sequence by resetting Init bit. If this is not done, the module will stay in Bus-Off state.

The DCAN provides an automatic Auto-Bus-On feature that is enabled by bit ABO in CAN Control Register. If set, the DCAN will automatically start the Bus-Off-Recovery sequence. The sequence can be delayed by a user-defined number of VCLK cycles that can be defined in Auto-Bus-On Time Register.

NOTE: If the DCAN goes Bus-Off due to massive occurrence of CAN bus errors, it stops all bus activities and automatically sets the Init bit. Once the Init bit has been reset by the CPU or due to the Auto-Bus-On feature, the device will wait for 129 occurrences of Bus Idle (equal to 129 × 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus-Off recovery sequence, the error counters will be reset.

26.10 Interrupt Functionality

Interrupts can be generated on two interrupt lines:

1. DCAN0INT line
2. DCAN1INT line

These lines can be enabled by setting the IE0 and IE1 bits, respectively, in the CAN Control Register.

The DCAN provides three groups of interrupt sources: Message Object Interrupts, Status Change Interrupts and Error Interrupts (see Figure 26-12 and Figure 26-13).

The source of an interrupt can be determined by the interrupt identifiers Int0ID / Int1ID in the Interrupt Register (see Section 26.17.5). When no interrupt is pending, the register will hold the value zero.

Each interrupt line remains active until the dedicated field in the Interrupt Register DCAN INT (Int0ID / Int1ID) again reach zero (this means the cause of the interrupt is reset), or until IE0 / IE1 are reset.

The value 0x8000 in the Int0ID field indicates that an interrupt is pending because the CAN Core has updated (not necessarily changed) the Error and Status Register (Error Interrupt or Status Interrupt). This interrupt has the highest priority. The CPU can update (reset) the status bits WakeUpPnd, RxOk, TxOk and LEC by reading the Error and Status Register DCAN ES, but a write access of the CPU will never generate or reset an interrupt.

Values between 1 and the number of the last message object indicates that the source of the interrupt is one of the message objects, Int0ID resp. Int1ID will point to the pending message interrupt with the highest priority. The Message Object 1 has the highest priority, the last message object has the lowest priority.

An interrupt service routine that reads the message that is the source of the interrupt, may read the message and reset the message object's IntPnd at the same time (ClrIntPnd bit in the IF1/IF2 Command Register). When IntPnd is cleared, the Interrupt Register will point to the next message object with a pending interrupt.

26.10.1 Message Object Interrupts

Message Object interrupts are generated by events from the message objects. They are controlled by the flags IntPND, TxIE, and RxIE, that are described in Section 26.5.1.

Message Object interrupts can be routed to either DCAN0INT or DCAN1INT line, controlled by the Interrupt Multiplexer Register (see Section 26.17.18).

26.10.2 Status Change Interrupts

The events WakeUpPnd, RxOk, TxOk and LEC in Error and Status Register (DCAN ES) belong to the Status Change Interrupts. The Status Change Interrupt group can be enabled by bit in CAN Control Register.

If SIE is set, a Status Change Interrupt will be generated at each CAN frame, independent of bus errors or valid CAN communication, and also independent of the Message RAM configuration.

Status Change interrupts can only be routed to interrupt line DCAN0INT that has to be enabled by setting IE0 in the CAN Control Register.

NOTE: Reading the Error and Status Register will clear the WakeUpPnd flag. If in global power-down mode, the WakeUpPnd flag is cleared by such a read access before the DCAN module has been waken up by the system, the DCAN may re-assert the WakeUpPnd flag, and a second interrupt may occur (additional information can be found in Section 26.11.2).

26.10.3 Error Interrupts

The events PER, BOff and EWarn (monitored in Error and Status Register, DCAN ES) belong to the Error Interrupts. The Error Interrupt group can be enabled by setting bit EIE in CAN Control Register.

Error interrupts can only be routed to interrupt line DCAN0INT that has to be enabled by setting IE0 in the CAN Control Register.

Figure 26-12. CAN Interrupt Topology 1

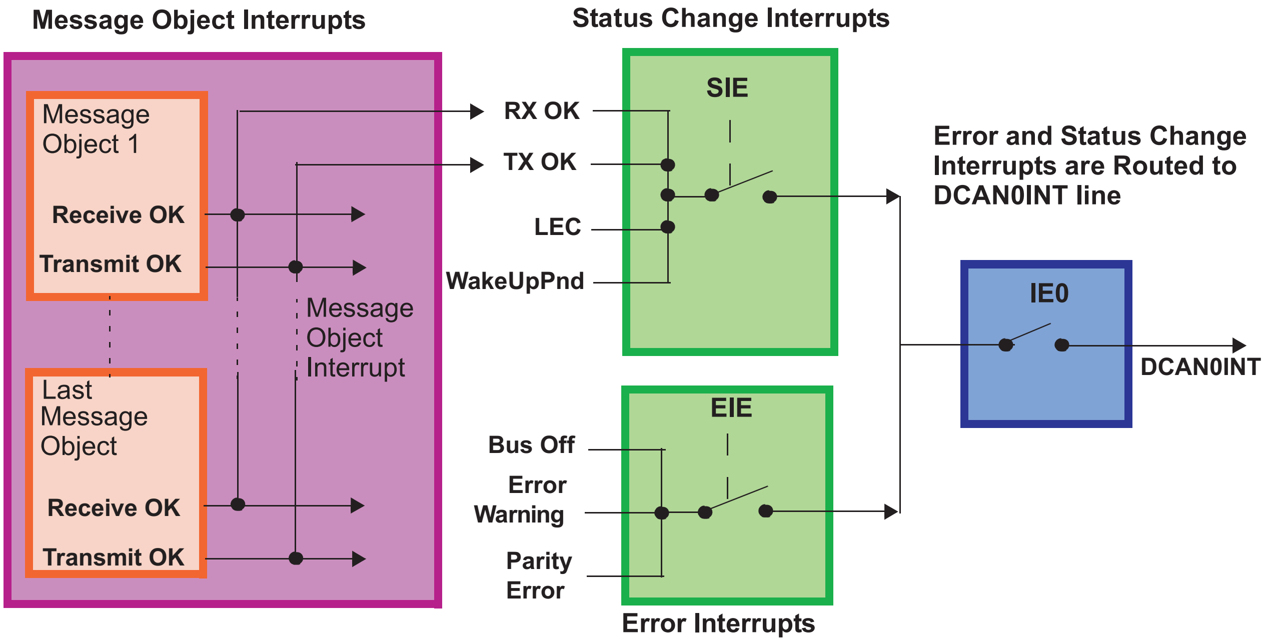
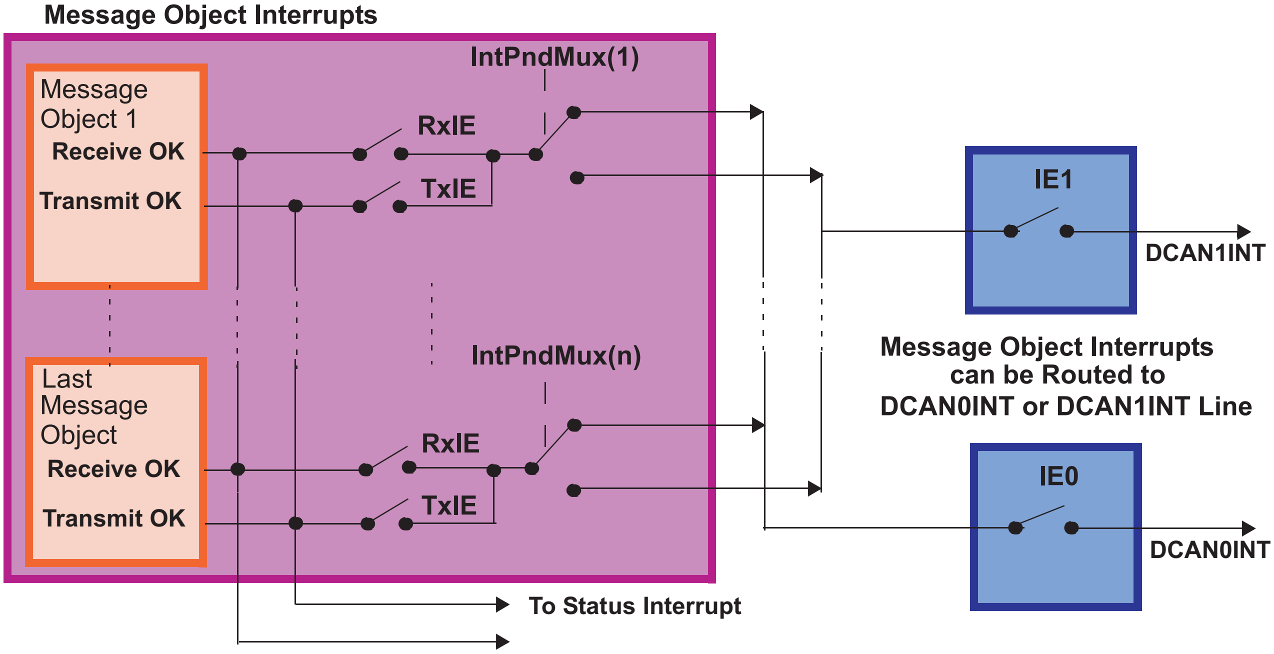


Figure 26-13. CAN Interrupt Topology 2



26.11 Global Power-Down Mode

The device architecture supports a centralized global power down control over the peripheral modules through the Peripheral Central Resource (PCR) module.

26.11.1 Entering Global Power-Down Mode

The global power-down mode for the DCAN is requested by setting the appropriate peripheral power down set bit (PSPWRDWNSETx) in the PCR module.

The DCAN then finishes all transmit requests of the message objects. When all requests are done, the DCAN waits until a bus idle state is recognized. Then it will automatically set the Init bit to indicate that the global power-down mode has been entered.

26.11.2 Wakeup From Global Power-Down Mode

When the DCAN module is in global power-down mode, a CAN bus activity detection circuit exists, which can be active, if enabled. If this circuit is active,on occurrence of a dominant CAN bus level, the DCAN will set the WakeUpPnd bit in Error and Status Register (DCAN ES).

If Status Interrupts are enabled, also an interrupt will be generated. This interrupt could be used by the application to wakeup the DCAN. For this, the application needs to set the appropriate peripheral power down clear bit (PSPWRDWNCLRx) in the PCR module, and to clear the Init bit in CAN Control Register.

After the Init bit has been cleared, the DCAN module waits until it detects 11 consecutive recessive bits on the CAN_RX pin and then goes Bus-Active again.

NOTE: The CAN transceiver circuit has to stay active during CAN bus activity detection. The first CAN message, which initiates the bus activity, cannot be received. This means that the first message received in power-down mode is lost.

26.12 Local Power-Down Mode

Besides the centralized power down mechanism controlled by the PCR module (global power down, see Section 26.15), the DCAN supports a local power-down mode that can be controlled within the DCAN control registers.

26.12.1 Entering Local Power-Down Mode

The local power-down mode is requested by setting the PDR bit in CAN Control Register.

The DCAN then finishes all transmit requests of the message objects. When all requests are done, DCAN waits until a bus idle state is recognized. Then it will automatically set the Init bit in CAN Control Register to prevent any further CAN transfers, and it will also set the PDA bit in CAN Error and Status Register. With setting the PDA bits, the DCAN module indicates that the local power-down mode has been entered.

During local power-down mode, the internal clocks of the DCAN module are turned off, but there is a wake up logic (see Section 26.12.2) that can be active, if enabled. Also the actual contents of the control registers can be read back.

NOTE: In local low power mode, the application should not clear the Init bit while PDR is set. If there are any messages in the Message RAM configured as to be transmitted and the application resets the init bit, these messages may be sent.

26.12.2 Wakeup From Local Power-Down Mode

There are two ways to wake up the DCAN from local power-down mode:

1. The application could wake up the DCAN module manually by clearing the PDR bit and then clearing the Init bit in CAN Control Register.
2. Alternatively, a CAN bus activity detection circuit can be activated by setting the wake up on bus activity bit (WUBA) in CAN Control Register. If this circuit is active, on occurrence of a dominant CAN bus level, the DCAN will automatically start the wake up sequence. It will clear the PDR bit in CAN Control Register and also clear the PDA bit in Error and Status Register. The WakeUpPnd bit in CAN Error and Status Register will be set. If Status Interrupts are enabled, also an interrupt will be generated. Finally the Init bit in CAN control register will be cleared.

After the Init bit has been cleared, the module waits until it detects 11 consecutive recessive bits on the CAN_RX pin and then goes Bus-Active again.

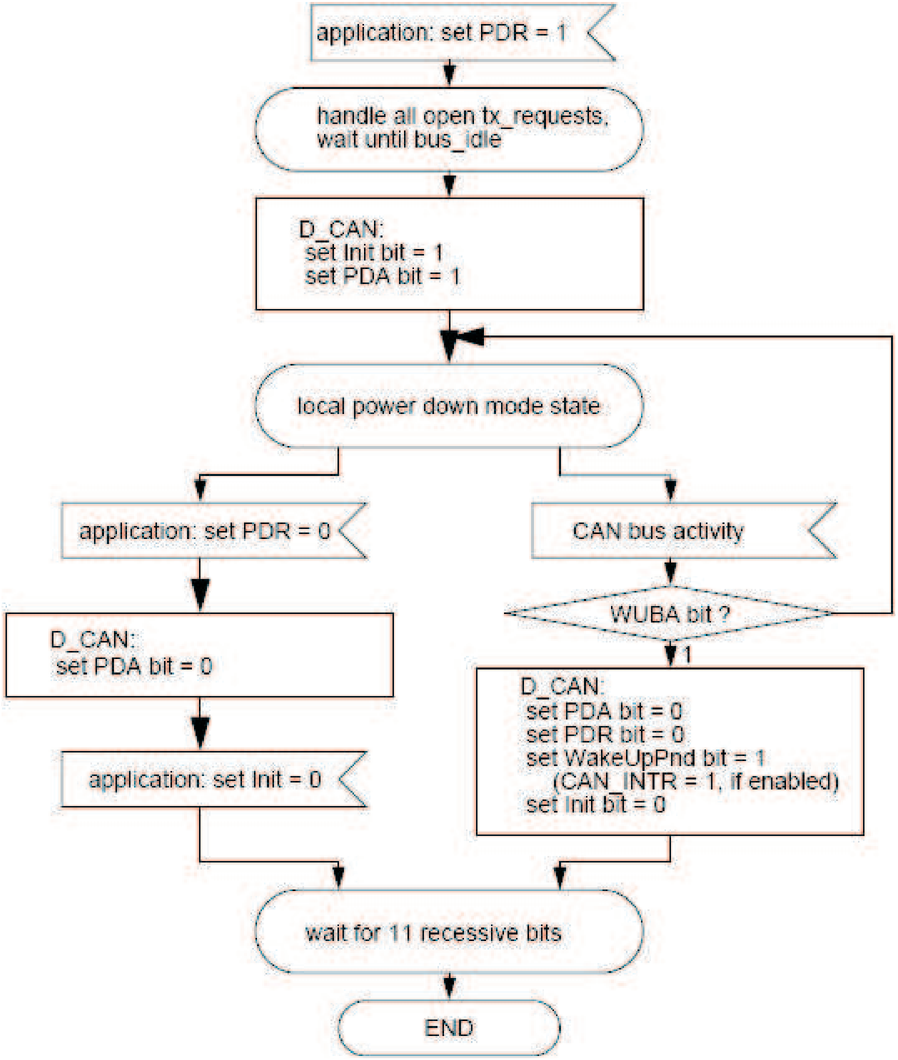
NOTE: The CAN transceiver circuit has to stay active while CAN bus observation. The first CAN message, which initiates the bus activity, cannot be received. This means that the first message received in power-down and automatic wake-up mode, is lost.

Figure 26-14 shows a flow diagram about entering and leaving local power-down mode.

26.13 GIO Support

The CAN_RX and CAN_TX pins of each DCAN module can be used as general purpose IO pins, if CAN functionality is not needed. This function is controlled by the CAN TX IO Control register (see Section 26.17.30) and the CAN RX IO Control register (see Section 26.17.31).

Figure 26-14. Local Power-Down Mode Flow Diagram


application: set PDR = 1
handle all open tx_requests, wait until bus_idle
D_CAN: set Init bit = 1 set PDA bit = 1
local power down mode state
application: set PDR = 0
CAN bus activity
WUBA bit ?
1
D_CAN: set PDA bit = 0
D_CAN: set PDA bit = 0 set PDR bit = 0 set WakeUpPnd bit = 1 (CAN_INTR = 1, if enabled) set Init bit = 0
application: set Init = 0
wait for 11 recessive bits
END

26.14 Test Modes

The DCAN provides several test modes that are mainly intended for production tests or self test.

For all test modes, the Test bit in the CAN Control Register needs to be set to 1. This enables write access to the Test Register.

NOTE: When using any of the Loop Back modes, it must be ensured by software that all message transfers are finished before setting the Init bit to 1.

26.14.1 Silent Mode

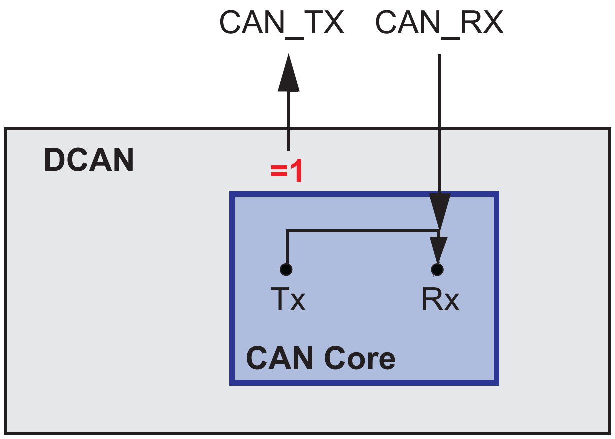
The Silent Mode may be used to analyze the traffic on the CAN bus without affecting it by sending dominant bits (for example, acknowledge bit, overload flag, active error flag). The DCAN is still able to receive valid data frames and valid remote frames, but it will not send any dominant bits. However, these are internally routed to the CAN Core.

Figure 26-15 shows the connection of signals CAN_TX and CAN_RX to the CAN Core in Silent Mode.

Silent Mode can be activated by setting the Silent bit in Test Register to 1.

In ISO 11898-1, the Silent Mode is called the Bus Monitoring Mode.

Figure 26-15. CAN Core in Silent Mode



26.14.2 Loop Back Mode

The Loop Back Mode is mainly intended for hardware self-test functions. In this mode, the CAN Core uses internal feedback from Tx output to Rx input. Transmitted messages are treated as received messages, and can be stored into message objects if they pass acceptance filtering. The actual value of the CAN_RX input pin is disregarded by the CAN Core. Transmitted messages still can be monitored at the CAN_TX pin.

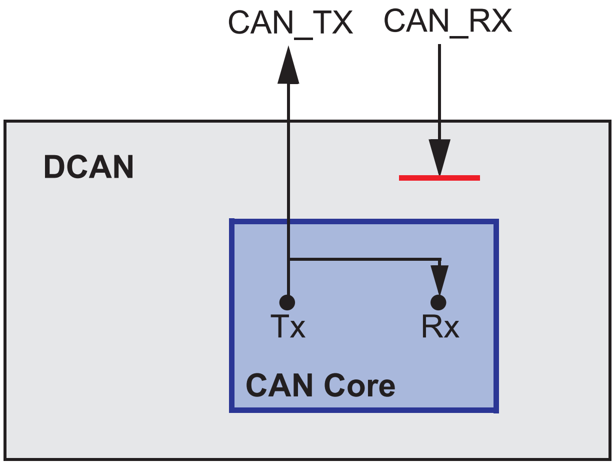
In order to be independent from external stimulation, the CAN Core ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in Loop Back Mode.

Figure 26-16 shows the connection of signals CAN_TX and CAN_RX to the CAN Core in Loop Back Mode.

Loop Back Mode can be activated by setting the LBack bit in Test Register to 1.

NOTE: In Loop Back mode, the signal path from CAN Core to Tx pin, the Tx pin itself, and the signal path from Tx pin back to CAN Core are disregarded. For including these into the testing, see External Loop Back mode (Section 26.14.3).

Figure 26-16. CAN Core in Loop Back Mode



26.14.3 External Loop Back Mode

The External Loop Back Mode is similar to the Loop Back Mode, however it includes the signal path from CAN Core to Tx pin, the Tx pin itself, and the signal path from Tx pin back to CAN Core. When External Loop Back Mode is selected, the input of the CAN core is connected to the input buffer of the Tx pin.

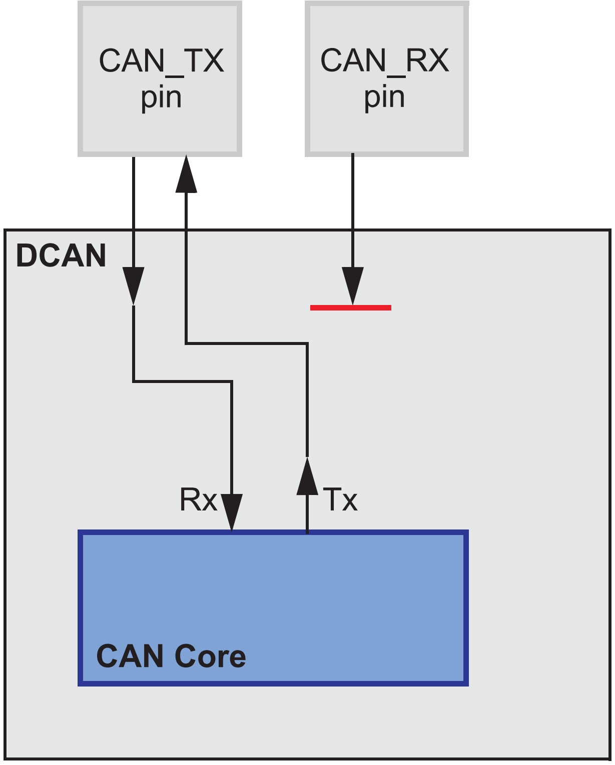
With this configuration, the Tx pin IO circuit can be tested.

External Loop Back Mode can be activated by setting the EXL bit in Test Register to 1.

Figure 26-17 shows the connection of signals CAN_TX and CAN_RX to the CAN Core in External Loop Back Mode.

NOTE: When Loop Back Mode is active (LBack bit is set), the EXL bit will be ignored.

Figure 26-17. CAN Core in External Loop Back Mode

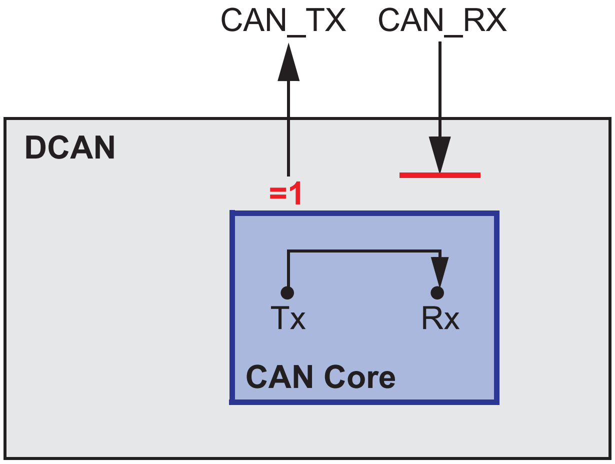


26.14.4 Loop Back Combined with Silent Mode

It is also possible to combine Loop Back Mode and Silent Mode by setting the LBack and Silent bits at the same time. This mode can be used for a “Hot Self-test”, that is, the DCAN hardware can be tested without affecting the CAN network. In this mode, the CAN_RX pin is disconnected from the CAN Core and no dominant bits will be sent on the CAN_TX pin.

Figure 26-18 shows the connection of the signals CAN_TX and CAN_RX to the CAN Core in case of the combination of Loop Back Mode with Silent Mode.

Figure 26-18. CAN Core in Loop Back Combined with Silent Mode



26.14.5 Software Control of CAN_TX Pin

Four output functions are available for the CAN transmit pin CAN_TX. Additionally to its default function (serial data output), the CAN_TX pin can drive constant dominant or recessive values, or it can drive the CAN Sample Point signal to monitor the CAN Core’s bit timing.

Combined with the readable value of the CAN_RX pin, this can be used to check the physical layer of the CAN bus.

The output mode of pin CAN_TX is selected by programming the Test Register Tx bits as described in Section 26.17.6.

NOTE: The software control for pin CAN_TX interferes with CAN protocol functions. For CAN message transfer or any of the test modes, Loop Back Mode, External Loop Back Mode, or Silent Mode, the CAN_TX pin should operate in its default functionality.

26.15 Parity Check Mechanism

The DCAN provides a parity check mechanism to ensure data integrity of Message RAM data. For each word (32 bits) in Message RAM, one parity bit will be calculated. The formation of the different words is according to the Message RAM representation in RDA mode (see Section 26.5.4).

Parity information is stored in the Message RAM on write accesses and will be checked against the stored parity bit from Message RAM on read accesses.

The parity check functionality can be enabled or disabled by PMD bit field in CAN Control Register.

In case of disabled parity check, the parity bits in message RAM will be left unchanged on write access to data area and no check will be done on read access.

If parity checking is enabled, parity bits will be automatically generated and checked by the DCAN. The parity bits could be read in Debug/Suspend mode (see Section 26.5.3) or in RDA mode (see Section 26.5.4). However, direct write access to the parity bits is only possible in this two modes with parity check disabled.

A parity bit will be set, if the modulo-2-sum of the data bits is 1. This definition is equivalent to: The parity bit will be set, if the number of 1 bits in the data is odd.

NOTE: Parity scheme can be changed via the System module DEV Parity Control Register1 on device basis for all peri RAMs.

26.15.1 Behavior on Parity Error

On any read access to Message RAM (for example, during start of a CAN frame transmission), the parity of the message object will be checked. If a parity error is detected, the PER bit in Error and Status Register will be set. If error interrupts are enabled, also an interrupt would be generated. In order to avoid the transmission of invalid data over the CAN bus, the d bit of the message object will be reset.

The message object data can be read by the host CPU, independently of parity errors. Thus, the application has to ensure that the read data is valid, for example, by immediately checking the Parity Error Code register on parity error interrupt.

NOTE: During RAM initialization, no parity check will be done.

26.15.2 Parity Testing

Testing the parity mechanism can be done by enabling the bit RDA (RamDirectAccess) and manually writing the parity bits directly to the dedicated RAM locations. With this, data and parity bits could be checked when reading directly from RAM.

NOTE: If parity check was disabled, the application has to ensure correct parity bit handling in order to prevent parity errors later on when parity check is enabled.

26.16 Debug/Suspend Mode

When the CPU is halted during debug, all DCAN registers are visible and can be inspected and modified by the CPU.

In addition, the Message RAM is directly memory-mapped as described in Table 26-3.

The CAN controller provides two options for entering the debug/suspend state. The options are controlled by the IDS bit in the CAN Control Register (DCAN CTL). By default, when IDS is 0, the DCAN controller completes any active transfers on the CAN bus and waits until the bus is idle before halting. When IDS is 1, the DCAN halts immediately as soon as the CPU is halted.

The InitDbg bit in DCAN CTL register indicates when the DCAN controller has actually entered the debug/suspend state.

NOTE: During Debug/Suspend Mode, the Message RAM cannot be accessed via the IFx register sets.

Writing to control registers in debug/suspend mode may influence the CAN state machine and further message handling.

For debug support, the auto clear functionality of the following DCAN registers is disabled:

- Error and Status Register (clear of status flags by read)
- IF1/IF2 Command Registers (clear of DMA Active flag by read-write)

26.17 DCAN Control Registers

Table 26-6 lists the control registers of the DCAN. After hardware reset, the registers of the DCAN hold the values shown in the register descriptions. The base address for the control registers is FFF7 DC00h for DCAN1, FFF7 DE00h for DCAN2, and FFF7 E000h for DCAN3.

Additionally, the Bus-Off state is reset and the CAN_TX pin is set to recessive (HIGH). The Init bit in the CAN Control Register is set to enable the software initialization. The DCAN will not influence the CAN bus until the CPU resets Init to 0.

Table 26-6. DCAN Control Registers

Offset	Acronym	Register Description	Section
00h	DCAN CTL	CAN Control Register	Section 26.17.1
04h	DCAN ES	Error and Status Register	Section 26.17.2
08h	DCAN ERRC	Error Counter Register	Section 26.17.3
0Ch	DCAN BTR	Bit Timing Register	Section 26.17.4
10h	DCAN INT	Interrupt Register	Section 26.17.5
14h	DCAN TEST	Test Register	Section 26.17.6
1Ch	DCAN PERR	Parity Error Code Register	Section 26.17.7
20h	DCAN REL	Core Release Register	Section 26.17.8
80h	DCAN ABOTR	Auto-Bus-On Time Register	Section 26.17.9
84h	DCAN TXRQX	Transmission Request X Register	Section 26.17.10
88h	DCAN TXRQ12	Transmission Request 12 Register	Section 26.17.11
8Ch	DCAN TXRQ34	Transmission Request 34 Register	Section 26.17.11
90h	DCAN TXRQ56	Transmission Request 56 Register	Section 26.17.11
94h	DCAN TXRQ78	Transmission Request 78 Register	Section 26.17.11
98h	DCAN NWDATX	New Data X Register	Section 26.17.12
9Ch	DCAN NWDAT12	New Data 12 Register	Section 26.17.13
A0h	DCAN NWDAT34	New Data 34 Register	Section 26.17.13
A4h	DCAN NWDAT56	New Data 56 Register	Section 26.17.13
A8h	DCAN NWDAT78	New Data 78 Register	Section 26.17.13

Table 26-6. DCAN Control Registers (continued)

Offset	Acronym	Register Description	Section
ACh	DCAN INTPNDX	Interrupt Pending X Register	Section 26.17.14
B0h	DCAN INTPND12	Interrupt Pending 12 Register	Section 26.17.15
B4h	DCAN INTPND34	Interrupt Pending 34 Register	Section 26.17.15
B8h	DCAN INTPND56	Interrupt Pending 56 Register	Section 26.17.15
BCh	DCAN INTPND78	Interrupt Pending 78 Register	Section 26.17.15
C0h	DCAN MSGVALX	Message Valid X Register	Section 26.17.16
C4h	DCAN MSGVAL12	Message Valid 12 Register	Section 26.17.17
C8h	DCAN MSGVAL34	Message Valid 34 Register	Section 26.17.17
CCh	DCAN MSGVAL56	Message Valid 56 Register	Section 26.17.17
D0h	DCAN MSGVAL78	Message Valid 78 Register	Section 26.17.17
D8h	DCAN INTMUX12	Interrupt Multiplexer 12 Register	Section 26.17.18
DCh	DCAN INTMUX34	Interrupt Multiplexer 34 Register	Section 26.17.18
E0h	DCAN INTMUX56	Interrupt Multiplexer 56 Register	Section 26.17.18
E4h	DCAN INTMUX78	Interrupt Multiplexer 78 Register	Section 26.17.18
100h	DCAN IF1CMD	IF1 Command Register	Section 26.17.19
104h	DCAN IF1MSK	IF1 Mask Register	Section 26.17.20
108h	DCAN IF1ARB	IF1 Arbitration Register	Section 26.17.21
10Ch	DCAN IF1MCTL	IF1 Message Control Register	Section 26.17.22
110h	DCAN IF1DATA	IF1 Data A Register	Section 26.17.23
114h	DCAN IF1DATB	IF1 Data B Register	Section 26.17.23
120h	DCAN IF2CMD	IF2 Command Register	Section 26.17.19
124h	DCAN IF2MSK	IF2 Mask Register	Section 26.17.20
128h	DCAN IF2ARB	IF2 Arbitration Register	Section 26.17.21
12Ch	DCAN IF2MCTL	IF2 Message Control Register	Section 26.17.22
130h	DCAN IF2DATA	IF2 Data A Register	Section 26.17.23
134h	DCAN IF2DATB	IF2 Data B Register	Section 26.17.23
140h	DCAN IF3OBS	IF3 Observation Register	Section 26.17.24
144h	DCAN IF3MSK	IF3 Mask Register	Section 26.17.25
148h	DCAN IF3ARB	IF3 Arbitration Register	Section 26.17.26
14Ch	DCAN IF3MCTL	IF3 Message Control Register	Section 26.17.27
150h	DCAN IF3DATA	IF3 Data A Register	Section 26.17.28
154h	DCAN IF3DATB	IF3 Data B Register	Section 26.17.28
160h	DCAN IF3UPD12	IF3 Update Enable 12 Register	Section 26.17.29
164h	DCAN IF3UPD34	IF3 Update Enable 34 Register	Section 26.17.29
168h	DCAN IF3UPD56	IF3 Update Enable 56 Register	Section 26.17.29
16Ch	DCAN IF3UPD78	IF3 Update Enable 78 Register	Section 26.17.29
1E0h	DCAN TIOC	CAN TX IO Control Register	Section 26.17.30
1E4h	DCAN RIOC	CAN RX IO Control Register	Section 26.17.31

26.17.1 CAN Control Register (DCAN CTL)

Figure 26-19. CAN Control Register (DCAN CTL) [offset = 00]

31 26	25	24
Reserved	WUBA	PDR
R-0	R/W-0	R/W-0

23 21	20	19	18	17	16
Reserved	DE3	DE2	DE1	IE1	InitDbg
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

15	14	13 10	9	8
SWR	Reserved	PMD	ABO	IDS
R/WP-0	R-0	R/W-5h	R/W-0	R/W-0

7	6	5	4	3	2	1	0
Test	CCE	DAR	Reserved	EIE	SIE	IE0	Init
R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-1

LEGEND: R/W = Read/Write; R = Read only; WP = Write protected by init bit; -*n* = value after reset

Table 26-7. CAN Control Register Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	These bits are always read as 0. Writes have no effect.
25	WUBA		Automatic wake up on bus activity when in local power-down mode.
		0	No detection of a dominant CAN bus level while in local power-down mode.
		1	Detection of a dominant CAN bus level while in local power-down mode is enabled. On occurrence of a dominant CAN bus level, the wake up sequence is started. (Additional information can be found in Section 26.12.) **Note:** The CAN message, which Initiates the bus activity, cannot be received. This means that the first message received in power-down and automatic wake-up mode, will be lost.
24	PDR		Request for local low power-down mode.
		0	No application request for local low power-down mode. If the application has cleared this bit while DCAN in local power-down mode, also the Init bit has to be cleared.
		1	Local power-down mode has been requested by application. The DCAN will acknowledge the local power-down mode by setting bit PDA in Error and Status Register. The local clocks will be turned off by DCAN internal logic (Additional information can be found in Section 26.12).
23-21	Reserved	0	These bits are always read as 0. Writes have no effect.
20	DE3		Enable DMA request line for IF3.
		0	Disabled
		1	Enabled **Note:** A pending DMA request for IF3 remains active until first access to one of the IF3 registers.
19	DE2		Enable DMA request line for IF2.
		0	Disabled
		1	Enabled **Note:** A pending DMA request for IF2 remains active until first access to one of the IF2 registers.
18	DE1		Enable DMA request line for IF1.
		0	Disabled
		1	Enabled **Note:** A pending DMA request for IF1 remains active until first access to one of the IF1 registers.
17	IE1		Interrupt line 1 enable.
		0	Disabled. Module Interrupt DCAN1INT is always low.
		1	Enabled. Interrupts will assert line DCAN1INT to 1; line remains active until pending interrupts are processed.

Table 26-7. CAN Control Register Field Descriptions (continued)

Bit	Field	Value	Description
16	InitDbg		Internal Init state while debug access.
		0	Not in debug mode, or debug mode requested but is not entered.
		1	Debug mode requested and is internally entered; the DCAN is ready for debug accesses.
15	SWR		SW Reset enable.
		0	Normal operation.
		1	Module is forced to reset state. This bit will automatically get cleared after execution of SW reset after one VBUSP clock cycle.
			Note: To execute SW reset the following procedure is necessary:
			1. Set Init bit to shut down CAN communication.
			2. Set SWR bit additionally to Init bit.
14	Reserved	0	This bit is always read as 0. Writes have no effect.
13-10	PMD		Parity enable.
		5h	Parity function is disabled.
		Others	Parity function is enabled.
9	ABO		Auto-Bus-On enable.
		0	The Auto-Bus-On feature is disabled.
		1	The Auto-Bus-On feature is enabled.
8	IDS		Interruption Debug Support enable.
		0	When Debug/Suspend mode is requested, DCAN will wait for a started transmission or reception to be completed before entering Debug/Suspend mode.
		1	When Debug/Suspend mode is requested, DCAN will interrupt any transmission or reception, and enter Debug/Suspend mode immediately.
7	Test		Test Mode enable.
		0	Normal operation
		1	Test mode
6	CCE		Configuration Change Enable.
		0	The CPU has no write access to the BTR config register.
		1	The CPU has write access to the BTR config register (when Init bit is set).
5	DAR		Disable Automatic Retransmission.
		0	Automatic Retransmission of not successful messages is enabled.
		1	Automatic Retransmission is disabled.
4	Reserved	0	This bit is always read as 0. Writes have no effect.
3	EIE		Error Interrupt Enable.
		0	Disabled. PER, BOff, and EWarn bits cannot generate an interrupt.
		1	Enabled. PER, BOff, and EWarn bits can generate an interrupt at DCAN0INT line and affect the Interrupt Register.
2	SIE		Status Change Interrupt Enable.
		0	Disabled. WakeUpPnd, RxOk, TxOk, and LEC bits cannot generate an interrupt.
		1	Enabled. WakeUpPnd, RxOk, TxOk, and LEC can generate an interrupt at DCAN0INT line and affect the Interrupt Register.
1	IE0		Interrupt line 0 enable.
		0	Disabled. Module Interrupt DCAN0INT is always low.
		1	Enabled. Interrupts will assert line DCAN0INT to 1; line remains active until pending interrupts are processed.
0	Init		Initialization.
		0	Normal operation.
		1	Initialization mode is entered.

NOTE: The Bus-Off recovery sequence (see CAN specification) cannot be shortened by setting or resetting Init bit. If the module goes Bus-Off, it will automatically set the Init bit and stop all bus activities.

When the Init bit is cleared by the application again, the module will then wait for 129 occurrences of Bus Idle (129 × 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus-Off recovery sequence, the error counters will be reset.

After the Init bit is reset, each time when a sequence of 11 recessive bits is monitored, a Bit0 error code is written to the Error and Status Register, enabling the CPU to check whether the CAN bus is stuck at dominant or continuously disturbed, and to monitor the proceeding of the Bus-Off recovery sequence.

26.17.2 Error and Status Register (DCAN ES)

Figure 26-20. Error and Status Register (DCAN ES) [offset = 04h]

31-16
Reserved
R-0

15-11	10	9	8
Reserved	PDA	WakeUpPnd	PER
R-0	R-0	R/C-0	R/C-0

7	6	5	4	3	2-0
BOff	EWarn	EPass	RxOK	TxOK	LEC
R-0	R-0	R-0	R/C-0	R/C-0	R/S-7h

LEGEND: R = Read only; C = Clear; S = Set; -*n* = value after reset

Table 26-8. Error and Status Register Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	0	These bits are always read as 0. Writes have no effect.
10	PDA		Local power-down mode acknowledge
		0	DCAN is not in local power-down mode.
		1	Application request for setting DCAN to local power-down mode was successful. DCAN is in local power-down mode.
9	WakeUpPnd		Wake Up Pending
			This bit can be used by the CPU to identify the DCAN as the source to wake up the system.
		0	No Wake Up is requested by DCAN.
		1	DCAN has initiated a wake up of the system due to dominant CAN bus while module power down. This bit will be reset if Error and Status Register is read.
8	PER		Parity Error Detected
		0	No parity error has been detected since last read access.
		1	The parity check mechanism has detected a parity error in the Message RAM. This bit will be reset if Error and Status Register is read.
7	BOff		Bus-Off State
		0	The CAN module is not Bus-Off state.
		1	The CAN module is in Bus-Off state.
6	EWarn		Warning State
		0	Both error counters are below the error warning limit of 96.
		1	At least one of the error counters has reached the error warning limit of 96.

Table 26-8. Error and Status Register Field Descriptions (continued)

Bit	Field	Value	Description
5	EPass		Error Passive State
		0	On CAN Bus error, the DCAN could send active error frames.
		1	The CAN Core is in the error passive state as defined in the CAN Specification.
4	RxOK		Received a message successfully
		0	No message has been successfully received since the last time when this bit was read by the CPU. This bit is never reset by DCAN internal events.
		1	A message has been successfully received since the last time when this bit was reset by a read access of the CPU (independent of the result of acceptance filtering).This bit will be reset if Error and Status Register is read.
3	TxOK		Transmitted a message successfully
		0	No message has been successfully transmitted since the last time when this bit was read by the CPU. This bit is never reset by DCAN internal events.
		1	A message has been successfully transmitted (error free and acknowledged by at least one other node) since the last time when this bit was reset by a read access of the CPU. This bit will be reset if Error and Status Register is read.
2-0	LEC		Last Error Code
			The LEC field indicates the type of the last error on the CAN bus. This field will be cleared to 0 when a message has been transferred (reception or transmission) without error.
		0	No Error
		1h	Stuff Error: More than five equal bits in a row have been detected in a part of a received message where this is not allowed.
		2h	Form Error: A fixed format part of a received frame has the wrong format.
		3h	Ack Error: The message this CAN Core transmitted was not acknowledged by another node.
		4h	Bit1 Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was dominant.
		5h	Bit0 Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (logical value 0), but the monitored bus level was recessive. During Bus-Off recovery, this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus-Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).
		6h	CRC Error: In a received message, the CRC check sum was incorrect. (CRC received for an incoming message does not match the calculated CRC for the received data).
		7h	No CAN bus event was detected since the last time when CPU has read the Error and Status Register. Any read access to the Error and Status Register re-Initializes the LEC to value 7h.

Interrupts are generated by bits PER, BOff, and EWarn (if EIE bit in CAN Control Register is set) and by bits WakeUpPnd, RxOk, TxOk, and LEC (if SIE bit in CAN Control Register is set). A change of bit EPass will not generate an Interrupt.

NOTE: Reading the Error and Status Register clears the WakeUpPnd, PER, RxOk, and TxOk bits and sets the LEC to value 7h. Additionally, the Status Interrupt value (8000h) in the Interrupt Register will be replaced by the next lower priority interrupt value.

For debug support, the auto clear functionality of Error and Status Register (clear of status flags by read) is disabled when in Debug/Suspend mode.

26.17.3 Error Counter Register (DCAN ERRC)

Figure 26-21. Error Counter Register (DCAN ERRC) [offset = 08h]

31 16
Reserved
R-0

15	14 8	7 0
RP	REC	TEC
R-0	R-0	R-0

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

Table 26-9. Error Counter Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	These bits are always read as 0. Writes have no effect.
15	RP		Receive Error Passive
		0	The Receive Error Counter is below the error passive level.
		1	The Receive Error Counter has reached the error passive level as defined in the CAN Specification.
14-8	REC	0-7Fh	Receive Error Counter. Actual state of the Receive Error Counter. (Values from 0 to 127.)
7-0	TEC	0-FFh	Transmit Error Counter. Actual state of the Transmit Error Counter. (Values from 0 to 255.)

26.17.4 Bit Timing Register (DCAN BTR)

Figure 26-22. Bit Timing Register (DCAN BTR) [offset = 0Ch]

31 – 20	19 – 16
Reserved	BRPE
R-0	R/WP-0

15	14 – 12	11 – 8	7 – 6	5 – 0
Rsvd	TSeg2	TSeg1	SJW	BRP
R-0	R/WP-2h	R/WP-3h	R/WP-0	R/WP-1h

LEGEND: R/W = Read/Write; R = Read only; WP = Write Protected by CCE bit; -*n* = value after reset

Table 26-10. Bit Timing Register Field Descriptions

Bit	Field	Value	Description
31-20	Reserved	0	These bits are always read as 0. Writes have no effect.
	BRPE	0-Fh	Baud Rate Prescaler Extension. Valid programmed values are 0 to 15. By programming BRPE the Baud Rate Prescaler can be extended to values up to 1024.
15	Reserved	0	This bit is always read as 0. Writes have no effect.
14-12	TSeg2	0-7h	Time segment after the sample point Valid programmed values are 0 to 7. The actual TSeg2 value that is interpreted for the Bit Timing will be the programmed TSeg2 value + 1.
11-8	TSeg1	1h-Fh	Time segment before the sample point Valid programmed values are 1 to 15. The actual TSeg1 value interpreted for the Bit Timing will be the programmed TSeg1 value + 1.
7-6	SJW	0-3h	Synchronization Jump Width Valid programmed values are 0 to 3.The actual SJW value interpreted for the Synchronization will be the programmed SJW value + 1.
5-0	BRP	0-3Fh	Baud Rate Prescaler Value by which the CAN_CLK frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid programmed values are 0 to 63.The actual BRP value interpreted for the Bit Timing will be the programmed BRP value + 1.

NOTE: This register is only writable if CCE and Init bits in the CAN Control Register are set.

The CAN bit time may be programmed in the range of 8 to 25 time quanta.

The CAN time quantum may be programmed in the range of 1 to 1024 CAN_CLK periods.

With a CAN_CLK of 8 MHz and BRPE = 00, the reset value of 2301h configures the DCAN for a bit rate of 500kBit/s.

For details, see Section 26.3.2.1.

26.17.5 Interrupt Register (DCAN INT)

Figure 26-23. Interrupt Register (DCAN INT) [offset = 10h]

31 24	23 16
Reserved	Int1ID
R-0	R-0

15 0
Int0ID
R-0

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

Table 26-11. Interrupt Register Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	These bits are always read as 0. Writes have no effect.
23-16	Int1ID		Interrupt 1 Identifier (indicates the message object with the highest pending interrupt).
		0	No interrupt is pending.
		1h-40h	Number of message object that caused the interrupt.
		41h-FFh	Reserved
			If several interrupts are pending, the CAN Interrupt Register will point to the pending interrupt with the highest priority. The DCAN1INT interrupt line remains active until Int1ID reaches value 0 (the cause of the interrupt is reset) or until IE1 is cleared.
			A message interrupt is cleared by clearing the message object's IntPnd bit.
			Among the message interrupts, the message object's interrupt priority decreases with increasing message number.
15-0	Int0ID		Interrupt 0 Identifier (the number here indicates the source of the interrupt).
		0	No interrupt is pending.
		1h-40h	Number of message object that caused the interrupt.
		41h-7FFFh	Reserved
		8000h	Error and Status Register value is not 0x07.
		8001h-FFFFh	Reserved
			If several interrupts are pending, the CAN Interrupt Register will point to the pending interrupt with the highest priority. The DCAN0INT interrupt line remains active until Int0ID reaches value 0 (the cause of the interrupt is reset) or until IE0 is cleared.
			The Status Interrupt has the highest priority. Among the message interrupts, the message object's interrupt priority decreases with increasing message number.

26.17.6 Test Register (DCAN TEST)

Figure 26-24. Test Register (DCAN TEST) [offset = 14h]

31–16
Reserved
R-0

15–10	9	8
Reserved	RDA	EXL
R-0	R/WP-0	R/WP-0

7	6–5	4	3	2–0
Rx	Tx	LBack	Silent	Reserved
R-U	R/WP-0	R/WP-0	R/WP-0	R-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write Protected by Test bit; -*n* = value after reset; U = Undefined

Table 26-12. Test Register Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	These bits are always read as 0. Writes have no effect.
9	RDA		RAM Direct Access Enable
		0	Normal Operation
		1	Direct access to the RAM is enabled while in Test Mode.
8	EXL		External Loop Back Mode
		0	Disabled
		1	Enabled
7	Rx		Receive Pin. Monitors the actual value of the CAN_RX pin.
		0	The CAN bus is dominant.
		1	The CAN bus is recessive.
6-5	Tx		Control of CAN_TX pin
		0	Normal operation, CAN_TX is controlled by the CAN Core.
		1h	Sample Point can be monitored at CAN_TX pin.
		2h	CAN_TX pin drives a dominant value.
		3h	CAN_TX pin drives a recessive value.
4	LBack		Loop Back Mode
		0	Disabled
		1	Enabled
3	Silent		Silent Mode
		0	Disabled
		1	Enabled
2-0	Reserved	0	These bits are always read as 0. Writes have no effect.

For all test modes, the Test bit in CAN Control Register needs to be set to 1. If Test bit is set, the RDA, EXL, Tx1, Tx0, LBack, and Silent bits are writable. Bit Rx monitors the state of pin CAN_RX and therefore is only readable. All Test Register functions are disabled when the Test bit is cleared to 0.

NOTE: The Test Register is only writable if Test bit in CAN Control Register is set.

Setting Tx to other than 00 will disturb message transfer.

When the internal loop back mode is active (LBack bit is set), EXL bit will be ignored.

26.17.7 Parity Error Code Register (DCAN PERR)

Figure 26-25. Parity Error Code Register (DCAN PERR) [offset = 1Ch]

31–16
Reserved
R-0

15–11	10–8	7–0
Reserved	Word Number	Message Number
R-0	R-U	R-U

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

Table 26-13. Parity Error Code Register Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	0	These bits are always read as 0. Writes have no effect.
10-8	Word Number		Word number where parity error has been detected.
		1h-5h	RDA word number (1 to 5) of the message object (according to the Message RAM representation in RDA mode, see Section 26.5.4).
7-0	Message Number	1h-FFh	Message object number where parity error has been detected. Only values 1h-40h are valid. Values 41h-FFh are invalid.

If a parity error is detected, the PER flag will be set in the Error and Status Register. This bit is not reset by the parity check mechanism; it must be reset by reading the Error and Status Register.

In addition to the PER flag, the Parity Error Code Register will indicate the memory area where the parity error has been detected (message number and word number).

If more than one word with a parity error was detected, the highest word number with a parity error will be displayed.

After a parity error has been detected, the register will hold the last error code until power is removed.

26.17.8 Core Release Register (DCAN REL)

Figure 26-26. Core Release Register (DCAN REL) [offset = 20h]

31–28	27–24	23–20	19–16
REL	STEP	SUBSTEP	YEAR
R-Ah	R-3h	R-1h	R-7h

15–8	7–0
MON	DAY
R-5h	R-4h

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

Table 26-14. Core Release Register (DCAN REL) Field Descriptions

Bit	Field	Value	Description
31-28	REL	0-9h	Core Release. One digit, BCD-coded.
27-24	STEP	0-9h	Step of Core Release. One digit, BCD-coded.
23-20	SUBSTEP	0-9h	Substep of Core Release. One digit, BCD-coded.
19-16	YEAR	0-9h	Design Time Stamp, Year. One digit, BCD-coded. This field is set by constant parameter on DCAN synthesis.
15-8	MON	0-12h	Design Time Stamp, Month. Two digits, BCD-coded. This field is set by constant parameter on DCAN synthesis.
7-0	DAY	0-31h	Design Time Stamp, Day. Two digits, BCD-coded. This field is set by constant parameter on DCAN synthesis.

26.17.9 Auto-Bus-On Time Register (DCAN ABOTR)

Figure 26-27. Auto-Bus-On Time Register (DCAN ABOTR) [offset = 80h]

31 — 16
ABO_TIME
R/W-0

15 — 0
ABO_TIME
R/W-0

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

Table 26-15. Auto-Bus-On Time Register Field Descriptions

Bit	Field	Value	Description
31-0	ABO_TIME	0-FFFF FFFFh	Number of VBUS clock cycles before a Bus-Off recovery sequence is started by clearing the Init bit. This function has to be enabled by setting the ABO bit in the CAN Control Register. The Auto-Bus-On timer is realized by a 32-bit counter that starts to count down to 0 when the module goes Bus-Off. The counter will be reloaded with the preload value of the ABO_TIME register after this phase.

NOTE: On write access to the CAN Control register while Auto-Bus-On timer is running, the Auto-Bus-On procedure will be aborted.

During Debug/Suspend mode, running Auto-Bus-On timer will be paused.

26.17.10 Transmission Request X Register (DCAN TXRQ X)

With the Transmission Request X Register, the CPU can detect if one or more bits in the different Transmission Request Registers are set. Each register bit represents a group of eight message objects. If at least one of the TxRqst bits of these message objects are set, the corresponding bit in the Transmission Request X Register will be set.

Figure 26-28. Transmission Request X Register (DCAN TXRQ X) [offset = 84h]

31 — 16
Reserved
R-0

15-14	13-12	11-10	9-8	7-6	5-4	3-2	1-0
TxRqstReg8	TxRqstReg7	TxRqstReg6	TxRqstReg5	TxRqstReg4	TxRqstReg3	TxRqstReg2	TxRqstReg1
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

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

Example

Bit 0 of the Transmission Request X Register represents byte 0 of the Transmission Request 1 Register (DCAN TXRQ12). If one or more bits in this byte are set, bit 0 of the Transmission Request X Register will be set.

26.17.11 Transmission Request Registers (DCAN TXRQ12 to DCAN TXRQ78)

These registers hold the TxRqst bits of the implemented message objects. By reading out these bits, the CPU can check for pending transmission requests. The TxRqst bit in a specific message object can be set/reset by the CPU via the IF1/IF2 Message Interface Registers, or by the Message Handler after reception of a remote frame or after a successful transmission.

Figure 26-29. Transmission Request 12 Register [offset = 88h]

31	0
TxRqst[32:1]	
R-0	

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

Figure 26-30. Transmission Request 34 Register [offset = 8Ch]

31	0
TxRqst[64:33]	
R-0	

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

Figure 26-31. Transmission Request 56 Register [offset = 90h]

31	0
TxRqst[96:65]	
R-0	

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

Figure 26-32. Transmission Request 78 Register [offset = 94h]

31	0
TxRqst[128:97]	
R-0	

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

Table 26-16. Transmission Request Registers Field Descriptions

Bit	Name	Value	Description
31-0	TxRqst[*n*]		Transmission Request Bits (for all message objects)
		0	No transmission has been requested for this message object.
		1	The transmission of this message object is requested and is not yet done.

26.17.12 New Data X Register (DCAN NWDAT X)

With the New Data X Register, the CPU can detect if one or more bits in the different New Data Registers are set. Each register bit represents a group of eight message objects. If at least on of the NewDat bits of these message objects are set, the corresponding bit in the New Data X Register will be set.

Figure 26-33. New Data X Register (DCAN NWDAT X) [offset = 98h]

31–16
Reserved
R-0

15–14	13–12	11–10	9–8	7–6	5–4	3–2	1–0
NewDatReg8	NewDatReg7	NewDatReg6	NewDatReg5	NewDatReg4	NewDatReg3	NewDatReg2	NewDatReg1
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

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

Example

Bit 0 of the New Data X Register represents byte 0 of the New Data 1 Register (DCAN NWDAT12). If one or more bits in this byte are set, bit 0 of the New Data X Register will be set.

26.17.13 New Data Registers (DCAN NWDAT12 to DCAN NWDAT78)

These registers hold the NewDat bits of the implemented message objects. By reading out these bits, the CPU can check for new data in the message objects. The NewDat bit of a specific message object can be set/reset by the CPU via the IF1/IF2 Interface Register sets, or by the Message Handler after reception of a data frame or after a successful transmission.

Figure 26-34. New Data 12 Register [offset = 9Ch]

31	0
NewDat[32:1]	
R-0	

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

Figure 26-35. New Data 34 Register [offset = A0h]

31	0
NewDat[64:33]	
R-0	

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

Figure 26-36. New Data 56 Register [offset = A4h]

31	0
NewDat[96:65]	
R-0	

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

Figure 26-37. New Data 78 Register [offset = A8h]

31	0
NewDat[128:97]	
R-0	

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

Table 26-17. New Data Registers Field Descriptions

Bit	Name	Value	Description
31-0	NewDat[*n*]		New Data Bits (for all message objects)
		0	No new data has been written into the data portion of this message object by the Message Handler since the last time when this flag was cleared by the CPU.
		1	The Message Handler or the CPU has written new data into the data portion of this message object.

26.17.14 Interrupt Pending X Register (DCAN INTPND X)

With the Interrupt Pending X Register, the CPU can detect if one or more bits in the different Interrupt Pending Registers are set. Each bit of this register represents a group of eight message objects. If at least one of the IntPnd bits of these message objects are set, the corresponding bit in the Interrupt Pending X Register will be set.

Figure 26-38. Interrupt Pending X Register (DCAN INTPND X) [offset = ACh]

31	16
Reserved	
R-0	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IntPndReg8		IntPndReg7		IntPndReg6		IntPndReg5		IntPndReg4		IntPndReg3		IntPndReg2		IntPndReg1	
R-0		R-0		R-0		R-0		R-0		R-0		R-0		R-0	

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

Example 2

Bit 0 of the Interrupt Pending X Register represents byte 0 of the Interrupt Pending 1 Register. If one or more bits in this byte are set, bit 0 of the Interrupt Pending X Register will be set.

26.17.15 Interrupt Pending Registers (DCAN INTPND12 to DCAN INTPND78)

These registers hold the IntPnd bits of the implemented message objects. By reading out these bits, the CPU can check for pending interrupts in the message objects. The IntPnd bit of a specific message object can be set/reset by the CPU via the IF1/IF2 Interface Register sets, or by the Message Handler after a reception or a successful transmission.

Figure 26-39. Interrupt Pending 12 Register [offset = B0h]

31 … 0
IntPnd[32:1]
R-0

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

Figure 26-40. Interrupt Pending 34 Register [offset = B4h]

31 … 0
IntPnd[64:33]
R-0

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

Figure 26-41. Interrupt Pending 56 Register [offset = B8h]

31 … 0
IntPnd[96:65]
R-0

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

Figure 26-42. Interrupt Pending 78 Register [offset = BCh]

31 … 0
IntPnd[128:97]
R-0

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

Table 26-18. Interrupt Pending Registers Field Descriptions

Bit	Name	Value	Description
31-0	IntPnd[*n*]		Interrupt Pending Bits (for all message objects)
		0	This message object is not the source of an interrupt.
		1	This message object is the source of an interrupt.

26.17.16 Message Valid X Register (DCAN MSGVAL X)

With the Message Valid X Register, the CPU can detect if one or more bits in the different Message Valid Registers are set. Each bit of this register represents a group of eight message objects. If at least one of the MsgVal bits of these message objects are set, the corresponding bit in the Message Valid X Register will be set.

Figure 26-43. Message Valid X Register (DCAN MSGVAL X) [offset = C0h]

31	16
Reserved	
R-0	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MsgValReg8		MsgValReg7		MsgValReg6		MsgValReg5		MsgValReg4		MsgValReg3		MsgValReg2		MsgValReg1	
R-0		R-0		R-0		R-0		R-0		R-0		R-0		R-0	

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

Example 3

Bit 0 of the Message Valid X Register represents byte 0 of the Message Valid 1 Register. If one or more bits in this byte are set, bit 0 of the Message Valid X Register will be set.

26.17.17 Message Valid Registers (DCAN MSGVAL12 to DCAN MSGVAL78)

These registers hold the MsgVal bits of the implemented message objects. By reading out these bits, the CPU can check which message objects are valid. The MsgVal bit of a specific message object can be set/reset by the CPU via the IF1/IF2 Interface Register sets, or by the Message Handler after a reception or a successful transmission.

Figure 26-44. Message Valid 12 Register [offset = C4h]

<table>
<tr><td>31</td><td align="right">0</td></tr>
<tr><td colspan="2" align="center">MsgVal[32:1]</td></tr>
<tr><td colspan="2" align="center">R-0</td></tr>
</table>

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

Figure 26-45. Message Valid 34 Register [offset = C8h]

<table>
<tr><td>31</td><td align="right">0</td></tr>
<tr><td colspan="2" align="center">MsgVal[64:33]</td></tr>
<tr><td colspan="2" align="center">R-0</td></tr>
</table>

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

Figure 26-46. Message Valid 56 Register [offset = CCh]

<table>
<tr><td>31</td><td align="right">0</td></tr>
<tr><td colspan="2" align="center">MsgVal[96:65]</td></tr>
<tr><td colspan="2" align="center">R-0</td></tr>
</table>

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

Figure 26-47. Message Valid 78 Register [offset = D0h]

<table>
<tr><td>31</td><td align="right">0</td></tr>
<tr><td colspan="2" align="center">MsgVal[128:97]</td></tr>
<tr><td colspan="2" align="center">R-0</td></tr>
</table>

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

Table 26-19. Message Valid Registers Field Descriptions

Bit	Name	Value	Description
31-0	MsgVal[*n*]		Message Valid Bits (for all message objects)
		0	This message object is ignored by the Message Handler.
		1	This message object is configured and will be considered by the Message Handler.

26.17.18 Interrupt Multiplexer Registers (DCAN INTMUX12 to DCAN INTMUX78)

The IntMux flag determines for each message object which of the two interrupt lines (DCAN0INT or DCAN1INT) will be asserted when the IntPnd of this message object is set. Both interrupt lines can be globally enabled or disabled by setting or clearing IE0 and IE1 bits in CAN Control Register.

The IntPnd bit of a specific message object can be set or reset by the CPU via the IF1/IF2 Interface Register sets, or by Message Handler after reception or successful transmission of a frame. This will also affect the Int0ID resp Int1ID flags in the Interrupt Register.

Figure 26-48. Interrupt Multiplexer 12 Register [offset = D8h]

31	0
IntMux[32:1]	
R/W-0	

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

Figure 26-49. Interrupt Multiplexer 34 Register [offset = DCh]

31	0
IntMux[64:33]	
R/W-0	

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

Figure 26-50. Interrupt Multiplexer 56 Register [offset = E0h]

31	0
IntMux[96:65]	
R/W-0	

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

Figure 26-51. Interrupt Multiplexer 78 Register [offset = E4h]

31	0
IntMux[128:97]	
R/W-0	

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

Table 26-20. Interrupt Multiplexer Registers Field Descriptions

Bit	Name	Value	Description
31-0	IntMux[*n*]		Multiplexes IntPnd value to either DCAN0INT or DCAN1INT interrupt lines. The mapping from the bits to the message objects is as follows: Bit 0 -> last implemented message object. Bit 1 -> message object number 1. Bit 2 -> message object number 2.
		0	DCAN0INT line is active if corresponding IntPnd flag is 1.
		1	DCAN1INT line is active if corresponding IntPnd flag is 1.

26.17.19 IF1/IF2 Command Registers (DCAN IF1CMD, DCAN IF2CMD)

The IF1/IF2 Command Register configure and Initiate the transfer between the IF1/IF2 Register sets and the Message RAM. It is configurable which portions of the message object should be transferred.

A transfer is started when the CPU writes the message number to bits [7:0] of the IF1/IF2 Command Register. With this write operation, the Busy bit is automatically set to 1 to indicate that a transfer is in progress.

After 4 to 14 VBUS clock cycles, the transfer between the Interface Register and the Message RAM will be completed and the Busy bit is cleared. The maximum number of cycles is needed when the message transfer concurs with a CAN message transmission, acceptance filtering, or message storage.

If the CPU writes to both IF1/IF2 Command Registers consecutively (request of a second transfer while first transfer is still in progress), the second transfer will start after the first one has been completed.

NOTE: While the Busy bit is 1, IF1/IF2 Register sets are write protected.

For debug support, the auto clear functionality of the IF1/IF2 Command Registers (clear of DMA Active flag by r/w) is disabled during Debug/Suspend mode.

If an invalid Message Number is written to bits [7:0] of the IF1/IF2 Command Register, the Message Handler may access an implemented (valid) message object instead.

Figure 26-52. IF1 Command Registers (DCAN IF1CMD) [offset = 100h]

31							24
Reserved							
R-0							
23	22	21	20	19	18	17	16
WR_RD	Mask	Arb	Control	ClrIntPnd	TxRqst/NewDat	Data A	Data B
R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0
15	14	13					8
Busy	DMA Active	Reserved					
R-0	R/WP/C-0	R-0					
7							0
Message Number							
R/WP-1h							

LEGEND: R/W = Read/Write; R = Read only; WP = Protected Write (protected by Busy bit); C = Clear by IF1 Access; -*n* = value after reset

Figure 26-53. IF2 Command Registers (DCAN IF2CMD) [offset = 120h]

31							24
Reserved							
R-0							
23	22	21	20	19	18	17	16
WR_RD	Mask	Arb	Control	ClrIntPnd	TxRqst/NewDat	Data A	Data B
R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0
15	14	13					8
Busy	DMA Active	Reserved					
R-0	R/WP/C-0	R-0					
7							0
Message Number							
R/WP-1h							

LEGEND: R/W = Read/Write; R = Read only; WP = Protected Write (protected by Busy bit); C = Clear by IF1 Access; -*n* = value after reset

Table 26-21. IF1/IF2 Command Register Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	These bits are always read as 0. Writes have no effect.
23	WR_RD		Write/Read
		0	Direction = Read: Transfer direction is from the message object addressed by Message Number (Bits [7:0]) to the IF1/IF2 register set.
		1	Direction = Write: Transfer direction is from the IF1/IF2 register set to the message object addressed by Message Number (Bits [7:0]).
22	Mask		Access Mask bits
		0	Mask bits will not be changed.
		1	Direction = Read: The Mask bits (Identifier Mask + MDir + MXtd) will be transferred from the message object addressed by Message Number (Bits [7:0]) to the IF1/IF2 Register set. Direction = Write: The Mask bits (Identifier Mask + MDir + MXtd) will be transferred from the IF1/IF2 Register set to the message object addressed by Message Number (Bits [7:0]).
21	Arb		Access Arbitration bits
		0	Arbitration bits will not be changed.
		1	Direction = Read: The Arbitration bits (Identifier + Dir + Xtd + MsgVal) will be transferred from the message object addressed by Message Number (Bits [7:0]) to the corresponding IF1/IF2 Register set. Direction = Write: The Arbitration bits (Identifier + Dir + Xtd + MsgVal) will be transferred from the IF1/IF2 Register set to the message object addressed by Message Number (Bits [7:0]).
20	Control		Access Control bits
		0	Control bits will not be changed
		1	Direction = Read: The Message Control bits will be transferred from the message object addressed by Message Number (Bits [7:0]) to the corresponding IF1/IF2 Register set. Direction = Write: The Message Control bits will be transferred from the IF1/IF2 Register set to the message object addressed by Message Number (Bits [7:0]). If the TxRqst/NewDat bit in this register (Bit [18]) is set, the TxRqst/NewDat bit in the IF1/IF2 Message Control Register will be ignored.
19	ClrIntPnd		Clear Interrupt Pending bit
		0	IntPnd bit will not be changed.
		1	Direction = Read: Clears IntPnd bit in the message object. Direction = Write: This bit is ignored. Copying of IntPnd flag from IF1/IF2 Registers to Message RAM can be controlled by only the Control flag (Bit [20]).
18	TxRqst/NewDat		Access Transmission Request bit
		0	Direction = Read: NewDat bit will not be changed. Direction = Write: TxRqst/NewDat bit will be handled according to the Control bit.
		1	Direction = Read: Clears NewDat bit in the message object. Direction = Write: Sets TxRqst/NewDat in the message object. **Note:** If a CAN transmission is requested by setting TxRqst/NewDat in this register, the TxRqst/NewDat bits in the message object will be set to 1 and independent of the values in IF1/IF2 Message Control Register. A read access to a message object can be combined with the reset of the control bits IntPnd and NewDat. The values of these bits transferred to the IF1/IF2 Message Control Register always reflect the status before resetting them.
17	Data A		Access Data Bytes 0-3
		0	Data Bytes 0-3 will not be changed.
		1	Direction = Read: The Data Bytes 0-3 will be transferred from the message object addressed by the Message Number (Bits [7:0]) to the corresponding IF1/IF2 Register set Direction = Write: The Data Bytes 0-3 will be transferred from the IF1/IF2 Register set to the message object addressed by the Message Number (Bits [7:0]) **Note:** The duration of the message transfer is independent of the number of bytes to be transferred.

Table 26-21. IF1/IF2 Command Register Field Descriptions (continued)

Bit	Field	Value	Description
16	Data B		Access Data Bytes 4-7
		0	Data Bytes 4-7 will not be changed
		1	Direction = Read: The Data Bytes 4-7 will be transferred from the message object addressed by the Message Number (Bits [7:0]) to the corresponding IF1/IF2 Register set. Direction = Write: The Data Bytes 4-7 will be transferred from the IF1/IF2 Register set to the message object addressed by the Message Number (Bits [7:0]). **Note:** The duration of the message transfer is independent of the number of bytes to be transferred.
15	Busy		Busy flag
		0	No transfer between IF1/IF2 Register set and Message RAM is in progress.
		1	Transfer between IF1/IF2 Register set and Message RAM is in progress. This bit is set to 1 after the message number has been written to bits [7:0]. IF1/IF2 Register set will be write-protected. The bit is cleared after read/write action has finished.
14	DMA Active		Activation of DMA feature for subsequent internal IF1/IF2 update
		0	DMA request line is independent of IF1/IF2 activities.
		1	DMA is requested after completed transfer between IF1/IF2 Register set and Message RAM. The DMA request remains active until the first read or write to one of the IF1/IF2 registers. An exception is a write to Message Number (Bits [7:0]) when DMA Active is 1. **Note:** Due to the auto reset feature of the DMA Active bit, this bit has to be separately set for each subsequent DMA cycle.
13-8	Reserved	0	These bits are always read as 0. Writes have no effect.
7-0	Message Number		Number of message object in Message RAM that is used for data transfer
		0	Invalid message number.
		1h-40h	Valid message numbers.
		41h-FFh	Invalid message numbers. **Note:** When an invalid message number is written to the IF1/IF2 Command Register that is higher than the last implemented message object number, a modulo addressing will occur. For example, when accessing message object 33 in a DCAN module with 32 message objects only, the message object 1 will be accessed instead.

26.17.20 IF1/IF2 Mask Registers (DCAN IF1MSK, DCAN IF2MSK)

The bits of the IF1/IF2 Mask Registers mirror the mask bits of a message object. The function of the relevant message objects bits is described in Section 26.5.1.

NOTE: While the Busy bit in the IF1/IF2 Command Register is 1, IF1/IF2 Register Set is write protected.

Figure 26-54. IF1 Mask Register (DCAN IF1MSK) [offset = 104h]

31	30	29	28	16
MXtd	MDir	Rsvd	Msk[28:16]	
R/WP-1	R/WP-1	R-1	R/WP-1FFFh	

15	0
Msk[15:0]	
R/WP-FFFFh	

LEGEND: R/W = Read/Write; R = Read only; WP = Protected Write (protected by Busy bit); -*n* = value after reset

Figure 26-55. IF2 Mask Register (DCAN IF2MSK) [offset = 124h]

31	30	29	28	16
MXtd	MDir	Rsvd	Msk[28:16]	
R/WP-1	R/WP-1	R-1	R/WP-1FFFh	

15	0
Msk[15:0]	
R/WP-FFFFh	

LEGEND: R/W = Read/Write; R = Read only; WP = Protected Write (protected by Busy bit); -*n* = value after reset

Table 26-22. IF1/IF2 Mask Register Field Descriptions

Bit	Field	Value	Description
31	MXtd		Mask Extended Identifier
		0	The extended identifier bit (IDE) has no effect on the acceptance filtering.
		1	The extended identifier bit (IDE) is used for acceptance filtering.
			When 11-bit ("standard") identifiers are used for a message object, the identifiers of received Data Frames are written into bits ID[28:18]. For acceptance filtering, only these bits with mask bits Msk[28:18] are considered.
30	MDir		Mask Message Direction
		0	The message direction bit (Dir) has no effect on the acceptance filtering.
		1	The message direction bit (Dir) is used for acceptance filtering.
29	Reserved	1	This bit is always read as 1. Writes have no effect.
28-0	Msk[*n*]		Identifier Mask
		0	The corresponding bit in the identifier of the message object is not used for acceptance filtering (don't care).
		1	The corresponding bit in the identifier of the message object is used for acceptance filtering.

26.17.21 IF1/IF2 Arbitration Registers (DCAN IF1ARB, DCAN IF2ARB)

The bits of the IF1/IF2 Arbitration Registers mirror the arbitration bits of a message object. The function of the relevant message objects bits is described in Section 26.5.1.

NOTE: While the Busy bit in the IF1/IF2 Command Register is 1, IF1/IF2 Register Set is write protected.

Figure 26-56. IF1 Arbitration Register (DCAN IF1ARB) [offset = 108h]

31	30	29	28	16
MsgVal	Xtd	Dir	ID[28:16]	
R/WP-0	R/WP-0	R/WP-0	R/WP-0	

15	0
ID[15:0]	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Protected Write (protected by Busy bit); -*n* = value after reset

Figure 26-57. IF2 Arbitration Register (DCAN IF2ARB) [offset = 128h]

31	30	29	28	16
MsgVal	Xtd	Dir	ID[28:16]	
R/WP-0	R/WP-0	R/WP-0	R/WP-0	

15	0
ID[15:0]	
R/WP-0	

LEGEND: R/W = Read/Write; WP = Protected Write (protected by Busy bit); -*n* = value after reset

Table 26-23. IF1/IF2 Arbitration Register Field Descriptions

Bit	Field	Value	Description
31	MsgVal		Message Valid
		0	The message object is ignored by the Message Handler.
		1	The message object is used by the Message Handler.
			Note: The CPU should reset the MsgVal bit of all unused Messages Objects during the initialization before it resets bit Init in the CAN Control Register. MsgVal must also be reset if the messages object is no longer used in operation. For reconfiguration of message objects during normal operation, see Section 26.7.6 and Section 26.7.7.
30	Xtd		Extended Identifier
		0	The 11-bit ("standard") identifier is used for this message object.
		1	The 29-bit ("extended") identifier is used for this message object.
29	Dir		Message direction
		0	Direction = Receive: On TxRqst, a Remote Frame with the identifier of this message object is transmitted. On receiving a Data Frame with a matching identifier, this message is stored in this message object.
		1	Direction = Transmit: On TxRqst, the respective message object is transmitted as a Data Frame. On receiving a Remote Frame with a matching identifier, the TxRqst bit of this message object is set (if RmtEn = 1).
28-0	ID[28:0]		Message Identifier
		ID[28:0]	29-bit Identifier ("Extended Frame")
		ID[28:18]	11-bit Identifier ("Standard Frame")

The Arbitration bits ID[28:0], Xtd, and Dir are used to define the identifier and type of outgoing messages and (together with the Mask bits Msk[28:0], MXtd, and MDir) for acceptance filtering of incoming messages.

A received message is stored into the valid message object with matching identifier and Direction = receive (Data Frame) or Direction = transmit (Remote Frame).

Extended frames can be stored only in message objects with Xtd = 1, standard frames in message objects with Xtd = 0.

If a received message (Data Frame or Remote Frame) matches more than one valid message objects, it is stored into the one with the lowest message number.

26.17.22 IF1/IF2 Message Control Registers (DCAN IF1MCTL, DCAN IF2MCTL)

The bits of the IF1/IF2 Message Control Registers mirror the message control bits of a message object. The function of the relevant message objects bits is described in Section 26.5.1.

NOTE: While the Busy bit in the IF1/IF2 Command Register is 1, IF1/IF2 Register Set is write protected.

Figure 26-58. IF1 Message Control Register (DCAN IF1MCTL) [offset = 10Ch]

31 16
Reserved
R-0

15	14	13	12	11	10	9	8
NewDat	MsgLst	IntPnd	UMask	TxIE	RxIE	RmtEn	TxRqst
R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

7	6 4	3 0
EoB	Reserved	DLC
R/WP-0	R-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Protected Write (protected by Busy bit); -*n* = value after reset

Figure 26-59. IF2 Message Control Register (DCAN IF2MCTL) [offset = 12Ch]

31 16
Reserved
R-0

15	14	13	12	11	10	9	8
NewDat	MsgLst	IntPnd	UMask	TxIE	RxIE	RmtEn	TxRqst
R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

7	6 4	3 0
EoB	Reserved	DLC
R/WP-0	R-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WP = Protected Write (protected by Busy bit); -*n* = value after reset

Table 26-24. IF1/IF2 Message Control Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		These bits are always read as 0. Writes have no effect.
15	NewDat		New Data
		0	No new data has been written into the data portion of this message object by the Message Handler since the last time this flag was cleared by the CPU.
		1	The Message Handler or the CPU has written new data into the data portion of this message object.
14	MsgLst		Message Lost (only valid for message objects with direction = receive)
		0	No message lost since the last time when this bit was reset by the CPU.
		1	The Message Handler stored a new message into this object when NewDat was still set, so the previous message has been overwritten.
13	IntPnd		Interrupt Pending
		0	This message object is not the source of an interrupt.
		1	This message object is the source of an interrupt. The interrupt identifier in the interrupt register will point to this message object if there is no other interrupt source with higher priority.
12	UMask		Use Acceptance Mask
		0	Mask ignored
		1	Use Mask (Msk[28:0], MXtd, and MDir) for acceptance filtering.
			If the UMask bit is set to 1, the message object's mask bits have to be programmed during initialization of the message object before MsgVal is set to 1.
11	TxIE		Transmit Interrupt Enable
		0	IntPnd will not be triggered after the successful transmission of a frame.
		1	IntPnd will be triggered after the successful transmission of a frame.
10	RxIE		Receive Interrupt Enable
		0	IntPnd will not be triggered after the successful reception of a frame.
		1	IntPnd will be triggered after the successful reception of a frame.
9	RmtEn		Remote Enable
		0	At the reception of a Remote Frame, TxRqst is not changed.
		1	At the reception of a Remote Frame, TxRqst is set.
8	TxRqst		Transmit Request
		0	This message object is not waiting for a transmission.
		1	The transmission of this message object is requested and not yet done.
7	EoB		End of Block
		0	The message object is part of a FIFO Buffer block and is not the last message object of this FIFO Buffer block.
		1	The message object is a single message object or the last message object in a FIFO Buffer block.
			Note: This bit is used to concatenate multiple message objects to build a FIFO Buffer. For single message objects (not belonging to a FIFO Buffer), this bit must always be set to 1.
6-4	Reserved		These bits are always read as 0. Writes have no effect.
3-0	DLC		Data Length Code
		0-8h	Data Frame has 0-8 data bytes.
		9h-Fh	Data Frame has 8 data bytes.
			Note: The Data Length Code of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it will write the DLC to the value given by the received message.

26.17.23 IF1/IF2 Data A and Data B Registers (DCAN IF1DATA/DATB, DCAN IF2DATA/DATB)

The data bytes of CAN messages are stored in the IF1/IF2 registers in the following order.

In a CAN Data Frame, Data 0 is the first, and Data 7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first

Figure 26-60. IF1 Data A Register (DCAN IF1DATA) [offset = 110h]

31	24	23	16
Data 3		Data 2	
R/WP-0		R/WP-0	

15	8	7	0
Data 1		Data 0	
R/WP-0		R/WP-0	

LEGEND: R/W = Read/Write; WP = Protected Write (protected by Busy bit); -*n* = value after reset

Figure 26-61. IF1 Data B Register (DCAN IF1DATB) [offset = 114h]

31	24	23	16
Data 7		Data 6	
R/WP-0		R/WP-0	

15	8	7	0
Data 5		Data 4	
R/WP-0		R/WP-0	

LEGEND: R/W = Read/Write; WP = Protected Write (protected by Busy bit); -*n* = value after reset

Figure 26-62. IF2 Data A Register (DCAN IF2DATA) [offset = 130h]

31	24	23	16
Data 3		Data 2	
R/WP-0		R/WP-0	

15	8	7	0
Data 1		Data 0	
R/WP-0		R/WP-0	

LEGEND: R/W = Read/Write; WP = Protected Write (protected by Busy bit); -*n* = value after reset

Figure 26-63. IF2 Data B Register (DCAN IF2DATB) [offset = 134h]

31	24	23	16
Data 7		Data 6	
R/WP-0		R/WP-0	

15	8	7	0
Data 5		Data 4	
R/WP-0		R/WP-0	

LEGEND: R/W = Read/Write; WP = Protected Write (protected by Busy bit); -*n* = value after reset

26.17.24 IF3 Observation Register (DCAN IF3OBS)

The IF3 register set can automatically be updated with received message objects without the need to Initiate the transfer from Message RAM by CPU (Additional information can be found in Section 26.5.1).

The observation flags (bits [4:0]) in the IF3 Observation register are used to determine which data sections of the IF3 Interface Register set have to be read in order to complete a DMA read cycle. After all marked data sections are read, the DCAN is enabled to update the IF3 Interface Register set with new data.

Any access order of single bytes or half-words is supported. When using byte or half-word accesses, a data section is marked as completed, if all bytes are read.

NOTE: If IF3 Update Enable is used and no Observation flag is set, the corresponding message objects will be copied to IF3 without activating the DMA request line and without waiting for DMA read accesses.

A write access to this register aborts a pending DMA cycle by resetting the DMA line and enables updating of IF3 Interface Register set with new data. To avoid data inconsistency, the DMA controller should be disabled before reconfiguring IF3 observation register.

The status of the current read-cycle can be observed via status flags (Bits [12:8]).

An interrupt request may be generated by the IF3Upd flag if the DE3 bit of DCAN CTL register is set. See the device data sheet to find out if this interrupt source is available.

With this, the observation status bits and the IF3Upd bit could be used by the application to realize the notification about new IF3 content in polling or interrupt mode.

Figure 26-64. IF3 Observation Register (DCAN IF3OBS) [offset = 140h]

31-16
Reserved
R-0

15	14-13	12	11	10	9	8
IF3Upd	Reserved	IF3SDB	IF3SDA	IF3SC	IF3SA	IF3SM
R-0	R-0	R-0	R-0	R-0	R-0	R-0

7-5	4	3	2	1	0
Reserved	DataB	DataA	Ctrl	Arb	Mask
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

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

Table 26-25. IF3 Observation Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	These bits are always read as 0. Writes have no effect.
15	IF3Upd		IF3 Update Data
		0	No new data has been loaded since IF3 was last read.
		1	New data has been loaded since IF3 was last read.
14-13	Reserved	0	These bits are always read as 0. Writes have no effect
12	IF3SDB		IF3 Status of Data B read access
		0	All Data B bytes are already read or are not marked to be read.
		1	Data B section still has data to read.
11	IF3SDA		IF3 Status of Data A read access
		0	All Data A bytes are already read or are not marked to be read.
		1	Data A section still has data to read.
10	IF3SC		IF3 Status of Control bits read access
		0	All Control section bytes are already read or are not marked to be read.
		1	Control section still has data to read.

Table 26-25. IF3 Observation Register Field Descriptions (continued)

Bit	Field	Value	Description
9	IF3SA		IF3 Status of Arbitration data read access
		0	All Arbitration data bytes are already read or are not marked to be read.
		1	Arbitration section still has data to read.
8	IF3SM		IF3 Status of Mask data read access
		0	All Mask data bytes are already read or are not marked to be read.
		1	Mask section still has data to read.
7-5	Reserved	0	These bits are always read as 0. Writes have no effect
4	DataB		Data B read observation
		0	Data B section does not need to be read.
		1	Data B section has to be read to enable next IF3 update.
3	DataA		Data A read observation
		0	Data A section does not need to be read.
		1	Data A section has to be read to enable next IF3 update.
2	Ctrl		Ctrl read observation
		0	Ctrl section does not need to be read.
		1	Ctrl section has to be read to enable next IF3 update.
1	Arb		Arbitration data read observation
		0	Arbitration data does not need to be read.
		1	Arbitration data has to be read to enable next IF3 update.
0	Mask		Mask data read observation
		0	Mask data does not need to be read.
		1	Mask data has to be read to enable next IF3 update.

26.17.25 IF3 Mask Register (DCAN IF3MSK)

Figure 26-65. IF3 Mask Register (DCAN IF3MSK) [offset = 144h]

31	30	29	28	16
MXtd	MDir	Rsvd	Msk[28:16]	
R-1	R-1	R-1	R-1FFFh	

15	0
Msk[15:0]	
R-FFFFh	

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

Table 26-26. IF3 Mask Register Field Descriptions

Bit	Field	Value	Description
31	MXtd		Mask Extended Identifier
		0	The extended identifier bit (IDE) has no effect on acceptance filtering.
		1	The extended identifier bit (IDE) is used for acceptance filtering.
			Note: When 11-bit ("standard") identifiers are used for a message object, the identifiers of received Data Frames are written into bits ID[28:18]. For acceptance filtering, only these bits, together with mask bits Msk[28:18], are considered.
30	MDir		Mask Message Direction
		0	The message direction bit (Dir) has no effect on acceptance filtering.
		1	The message direction bit (Dir) is used for acceptance filtering.
29	Reserved	1	This bit is always read as 1. Writes have no effect.
28-0	Msk[*n*]		Identifier Mask
		0	The corresponding bit in the identifier of the message object is not used for acceptance filtering (don't care).
		1	The corresponding bit in the identifier of the message object is used for acceptance filtering.

26.17.26 IF3 Arbitration Register (DCAN IF3ARB)

Figure 26-66. IF3 Arbitration Register (DCAN IF3ARB) [offset = 148h]

31	30	29	28	16
MsgVal	Xtd	R-0	ID[28:16]	
R-0	R-0	R-0	R-0	

15	0
ID[15:0]	
R-0	

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

Table 26-27. IF3 Arbitration Register Field Descriptions

Bit	Field	Value	Description
31	MsgVal		Message Valid
		0	The message object is ignored by the Message Handler.
		1	The message object is to be used by the Message Handler.
			Note: The CPU should reset the MsgVal bit of all unused Messages Objects during the initialization before it resets bit Init in the CAN Control Register. MsgVal must also be reset if the messages object is no longer used in operation. For reconfiguration of message objects during normal operation, see Section 26.7.6 and Section 26.7.7.
30	Xtd		Extended Identifier
		0	The 11-bit ("standard") identifier is used for this message object.
		1	The 29-bit ("extended") identifier is used for this message object.
29	Dir		Message direction
		0	Direction = Receive: On TxRqst, a remote frame with the identifier of this message object is transmitted. On receiving a data frame with a matching identifier, the message is stored in this message object.
		1	Direction = Transmit: On TxRqst, the respective message object is transmitted as a data frame. On receiving a remote frame with a matching identifier, the TxRqst bit of this message object is set (if RmtEn = 1).
28-0	ID[28:0]		Message Identifier
		ID[28:0]	29-bit Identifier ("Extended Frame")
		ID[28:18]	11-bit Identifier ("Standard Frame")

26.17.27 IF3 Message Control Register (DCAN IF3MCTL)

Figure 26-67. IF3 Message Control Register (DCAN IF3MCTL) [offset = 14Ch]

31–16
Reserved
R-0

15	14	13	12	11	10	9	8
NewDat	MsgLst	IntPnd	UMask	TxIE	RxIE	RmtEn	TxRqst
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

7	6–4	3–0
EoB	Reserved	DLC
R-0	R-0	R-0

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

Table 26-28. IF3 Message Control Register Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	These bits are always read as 0. Writes have no effect.
15	NewDat		New Data
		0	No new data has been written into the data portion of this message object by the Message Handler since the last time this flag was cleared by the CPU.
		1	The Message Handler or the CPU has written new data into the data portion of this message object.
14	MsgLst		Message Lost (only valid for message objects with direction = receive)
		0	No message lost since the last time when this bit was reset by the CPU.
		1	The Message Handler stored a new message into this object when NewDat was still set, so the previous message has been overwritten.
13	IntPnd		Interrupt Pending
		0	This message object is not the source of an interrupt.
		1	This message object is the source of an interrupt. The interrupt identifier in the interrupt register will point to this message object if there is no other interrupt source with higher priority.
12	UMask		Use Acceptance Mask
		0	Mask ignored
		1	Use Mask (Msk[28:0], MXtd, and MDir) for acceptance filtering.
			If the UMask bit is set to 1, the message object's mask bits have to be programmed during initialization of the message object before MsgVal is set to 1.
11	TxIE		Transmit Interrupt Enable
		0	IntPnd will not be triggered after the successful transmission of a frame.
		1	IntPnd will be triggered after the successful transmission of a frame.
10	RxIE		Receive Interrupt Enable
		0	IntPnd will not be triggered after the successful transmission of a frame.
		1	IntPnd will be triggered after the successful transmission of a frame.
9	RmtEn		Remote Enable
		0	At the reception of a Remote Frame, TxRqst is not changed.
		1	At the reception of a Remote Frame, TxRqst is set.
8	TxRqst		Transmit Request
		0	This message object is not waiting for a transmission.
		1	The transmission of this message object is requested and not yet done.

Table 26-28. IF3 Message Control Register Field Descriptions (continued)

Bit	Field	Value	Description
7	EoB		End of Block
		0	The message object is part of a FIFO Buffer block and is not the last message object of the FIFO Buffer block.
		1	The message object is a single message object or the last message object in a FIFO Buffer block.
			Note: This bit is used to concatenate multiple message objects to build a FIFO Buffer. For single message objects (not belonging to a FIFO Buffer), this bit must always be set to 1.
6-4	Reserved	0	These bits are always read as 0. Writes have no effect.
3-0	DLC		Data Length Code
		0-8h	Data Frame has 0-8 data bytes.
		9h-Fh	Data Frame has 8 data bytes.
			Note: The Data Length Code of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it will write the DLC to the value given by the received message.

26.17.28 IF3 Data A and Data B Registers (DCAN IF3DATA/DATB)

The data bytes of CAN messages are stored in the IF3 registers in the following order.

In a CAN Data Frame, Data 0 is the first, and Data 7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first.

Figure 26-68. IF3 Data A Register (DCAN IF3DATA) [offset = 150h]

31–24	23–16
Data 3	Data 2
R-0	R-0

15–8	7–0
Data 1	Data 0
R-0	R-0

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

Figure 26-69. IF3 Data B Register (DCAN IF3DATB) [offset = 154h]

31–24	23–16
Data 7	Data 6
R/WP-0	R/WP-0

15–8	7–0
Data 5	Data 4
R-0	R-0

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

26.17.29 IF3 Update Enable Registers (DCAN IF3UPD12 to IF3UPD78)

The automatic update functionality of the IF3 register set can be configured for each message object. A message object is enabled for automatic IF3 update, if the dedicated IF3UpdEn flag is set. This means that an active NewDat flag of this message object (for example, due to reception of a CAN frame) will trigger an automatic copy of the whole message object to IF3 register set.

NOTE: IF3 Update enable should not be set for transmit objects.

Figure 26-70. IF3 Update Enable 12 Register [offset = 160h]

31 … 0
IF3UpdEn[32:1]
R/W-0

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

Figure 26-71. IF3 Update Enable 34 Register [offset = 164h]

31 … 0
IF3UpdEn[64:33]
R/W-0

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

Figure 26-72. IF3 Update Enable 56 Register [offset = 168h]

31 … 0
IF3UpdEn[96:65]
R/W-0

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

Figure 26-73. IF3 Update Enable 78 Register [offset = 16Ch]

31 … 0
IF3UpdEn[128:97]
R/W-0

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

Table 26-29. IF3 Update Control Register Field Descriptions

Bit	Name	Value	Description
31-0	IF3UpdEn[*n*]		IF3 Update Enabled (for all message objects)
		0	Automatic IF3 update is disabled for this message object.
		1	Automatic IF3 update is enabled for this message object. A message object is scheduled to be copied to IF3 register set, if NewDat flag of the message object is active.

26.17.30 CAN TX IO Control Register (DCAN TIOC)

The CAN_TX pin of the DCAN module can be used as general-purpose IO pin if CAN function is not needed.

NOTE: The values of the IO Control registers are only writable if Init bit of CAN Control Register is set.

The OD, Func, Dir, and Out bits of the CAN TX IO Control register are forced to certain values when Init bit of CAN Control Register is reset (see bit descriptions).

Figure 26-74. CAN TX IO Control Register (DCAN TIOC) [offset = 1E0h]

31-19	18	17	16
Reserved	PU	PD	OD
R-0	R/W-D	R/W-D	R/WP-0

15-4	3	2	1	0
Reserved	Func	Dir	Out	In
R-0	R/WP-0	R/WP-0	R/WP-0	R-U

LEGEND: R/W = Read/Write; R = Read only; WP = Protected Write (protected by Init bit); D = Device-dependent; -*n* = value after reset; U = Undefined

Table 26-30. CAN TX IO Control Register Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	These bits are always read as 0. Writes have no effect.
18	PU		CAN_TX Pullup/Pulldown select. This bit is only active when CAN_TX is configured to be an input.
		0	CAN_TX Pulldown is selected, when pull logic is active (PD = 0).
		1	CAN_TX Pullup is selected, when pull logic is active (PD = 0).
17	PD		CAN_TX pull disable. This bit is only active when CAN_TX is configured to be an input.
		0	CAN_TX pull is active.
		1	CAN_TX pull is disabled.
16	OD		CAN_TX open drain enable. This bit is only active when CAN_TX is configured to be in GIO mode (TIOC.Func = 0).
		0	The CAN_TX pin is configured in push/pull mode.
		1	The CAN_TX pin is configured in open drain mode.
			Forced to 0 if Init bit of CAN control register is reset.
15-4	Reserved	0	These bits are always read as 0. Writes have no effect.
3	Func		CAN_TX function. This bit changes the function of the CAN_TX pin.
		0	CAN_TX pin is in GIO mode.
		1	CAN_TX pin is in functional mode (as an output to transmit CAN data).
			Forced to 1 if Init bit of CAN control register is reset.
2	Dir		CAN_TX data direction. This bit controls the direction of the CAN_TX pin when it is configured to be in GIO mode only (TIOC.Func = 0).
		0	The CAN_TX pin is an input.
		1	The CAN_TX pin is an output.
			Forced to 1 if Init bit of CAN control register is reset.
1	Out		CAN_TX data out write. This bit is only active when CAN_TX pin is configured to be in GIO mode (TIOC.Func = 0) and configured to be an output pin (TIOC.Dir = 1). The value of this bit indicates the value to be output to the CAN_TX pin.
		0	The CAN_TX pin is driven to logic low (0).
		1	The CAN_TX pin is at logic high (1).
			Forced to Tx output of the CAN Core, if Init bit of CAN Control register is reset.

Table 26-30. CAN TX IO Control Register Field Descriptions (continued)

Bit	Field	Value	Description
0	In		CAN_TX data in.
		0	The CAN_TX pin is at logic low (0).
		1	The CAN_TX pin is at logic high (1).
			Note: When CAN_TX pin is connected to a CAN transceiver, an external pullup resistor has to be used to ensure that the CAN bus will not be disturbed (for example, while the DCAN module is reset).

26.17.31 CAN RX IO Control Register (DCAN RIOC)

The CAN_RX pin of the DCAN module can be used as general-purpose IO pin if CAN function is not needed.

NOTE: The values of the IO Control registers are writable only if Init bit of CAN Control Register is set.

The OD, Func, and Dir bits of the CAN RX IO Control register are forced to certain values when Init bit of CAN Control Register is reset, see bit description.

Figure 26-75. CAN RX IO Control Register (DCAN RIOC) [offset = 1E4h]

31-19	18	17	16
Reserved	PU	PD	OD
R-0	R/W-D	R/W-D	R/WP-0

15-4	3	2	1	0
Reserved	Func	Dir	Out	In
R-0	R/WP-0	R/WP-0	R/WP-0	R-U

LEGEND: R/W = Read/Write; R = Read only; WP = Protected Write (protected by Init bit); D = Device-dependent; -*n* = value after reset; U = Undefined

Table 26-31. CAN RX IO Control Register Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	These bits are always read as 0. Writes have no effect.
18	PU		CAN_RX Pullup/Pulldown select. This bit is only active when CAN_RX is configured to be an input.
		0	CAN_RX Pulldown is selected, when pull logic is active (PD = 0).
		1	CAN_RX Pullup is selected, when pull logic is active (PD = 0).
17	PD		CAN_RX pull disable. This bit is only active when CAN_RX is configured to be an input.
		0	CAN_RX pull is active.
		1	CAN_RX pull is disabled.
16	OD		CAN_RX open drain enable. This bit is only active when CAN_RX is configured to be in GIO mode (RIOC.Func = 0).
		0	The CAN_RX pin is configured in push/pull mode.
		1	The CAN_RX pin is configured in open drain mode.
			Forced to 0 if Init bit of CAN control register is reset.
15-4	Reserved	0	These bits are always read as 0. Writes have no effect.
3	Func		CAN_RX function. This bit changes the function of the CAN_RX pin.
		0	CAN_RX pin is in GIO mode.
		1	CAN_RX pin is in functional mode (as an input to receive CAN data).
			Forced to 1 if Init bit of CAN control register is reset.

Table 26-31. CAN RX IO Control Register Field Descriptions (continued)

Bit	Field	Value	Description
2	Dir		CAN_RX data direction. This bit controls the direction of the CAN_RX pin when it is configured to be in GIO mode only (RIOC.Func = 0).
		0	The CAN_RX pin is an input.
		1	The CAN_RX pin is an output.
			Forced to 0 if Init bit of CAN control register is reset.
1	Out		CAN_RX data out write. This bit is only active when CAN_RX pin is configured to be in GIO mode (RIOC.Func = 0) and configured to be an output pin (RIOC.Dir = 1). The value of this bit indicates the value to be output to the CAN_RX pin.
		0	The CAN_RX pin is driven to logic low (0).
		1	The CAN_RX pin is at logic high (1).
0	In		CAN_RX data in.
		0	The CAN_RX pin is at logic low (0).
		1	The CAN_RX pin is at logic high (1).
			Note: When CAN_RX pin is connected to a CAN transceiver, an external pullup resistor has to be used to ensure that the CAN bus will not be disturbed (for example, while the DCAN module is reset).

Multi-Buffered Serial Peripheral Interface Module (MibSPI) with Parallel Pin Option (MibSPIP)

This chapter provides the specifications for a 16-bit configurable synchronous multi-buffered multi-pin serial peripheral interface (MibSPI). This chapter also provides the specifications for MibSPI with Parallel Pin Option (MibSPIP). The MibSPI is a programmable-length shift register used for high-speed communication between external peripherals or other microcontrollers.

Throughout this chapter, all references to SPI also apply to MibSPI/MibSPIP, unless otherwise noted.

NOTE: This chapter describes a superset implementation of the MibSPI/SPI modules that includes features and functionality that may not be available on some devices. Device-specific content that should be determined by referencing the datasheet includes DMA functionality, MibSPI RAM size, number of transfer groups, number of chip selects, parallel mode support, and availability of 5-pin operation ($\overline{\text{SPIENA}}$).

Topic **Page**

27.1 Overview 1402
27.2 Operating Modes 1403
27.3 Test Features 1424
27.4 General-Purpose I/O 1426
27.5 Low-Power Mode 1426
27.6 Interrupts 1427
27.7 DMA Interface 1429
27.8 Module Configuration 1430
27.9 Control Registers 1432
27.10 Multi-Buffer RAM 1498
27.11 Parity Memory 1505
27.12 MibSPI Pin Timing Parameters 1508

27.1 Overview

The MibSPI/MibSPIP 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 in and out of the device at a programmed bit-transfer rate. Typical applications for the SPI include interfacing to external peripherals, such as I/Os, memories, display drivers, and analog-to-digital converters.

The SPI has the following attributes:

- 16-bit shift register
- Receive buffer register
- 11-bit baud clock generator
- Serial clock (SPICLK) pin
- 1 SPISOMI/SPISIMO pin for data transfer, with programmable pin direction
- SPI enable ($\overline{\text{SPIENA}}$) pin
- Up to 6 slave chip select ($\overline{\text{SPICS}}$) pins
- SPICLK can be internally-generated (and driven) or received from an external clock source
- Each word transferred can have a unique format
- SPI pins can be used as functional or digital Input/Output pins (GIOs)

NOTE: SIMO - Slave In Master Out Pin
SOMI - Slave Out Master In Pin
SPICS - SPI Chip Select Pin
SPIENA - SPI Enable Pin

27.1.1 Word Format Options

Each word transferred can have a unique format. Several format characteristics are programmable for each word transferred:

- SPICLK frequency
- Character length (2 to 16 bits)
- Phase (with and without delay)
- Polarity (high or low)
- Parity enabled/disabled
- Chip Select(CS) timers for setup and hold
- Shift direction (Most-Significant Bit (MSB) first or Least-Significant Bit (LSB) first)
- Multi-pin parallel modes

27.1.2 Multi-buffering (Mib) Support

The MibSPI has a programmable buffer memory that enables programmed transmission to be completed without CPU intervention. The buffers are combined in different Transfer Groups (TGs) that can be triggered by external events (timers, Input/Output activity, and so on) or by the internal tick counter. The internal tick counter supports periodic trigger events. Each buffer of the MibSPI can be associated with different DMA channels in different TGs, allowing the user to move data between internal memory and an external slave with minimal CPU interaction.

27.1.2.1 Multi-buffer Mode

Multi-buffer Mode is an extension to the SPI. In multi-buffer mode, many extended features are configurable:

- Number of buffers for each peripheral (or data source/destination, up to 128 buffers supported) or group (up to 8 groupings)
- Triggers for each groups, trigger types, trigger sources for individual groups (14 external trigger sources and 1 internal trigger source supported)
- Memory fault detection via an internal parity circuit
- Number of DMA-controlled buffers and number of DMA request channels (up to 8 for each of transmit and receive)
- Number of DMA transfers for each buffer (up to 65536 words for up to 8 buffers)
- Uninterrupted DMA buffer transfer (NOBREAK buffer)

27.1.2.2 Compatibility Mode

Compatibility Mode of the MibSPI makes it behave exactly like a standard platform SPI module and ensures full compatibility with other SPIs. All features in compatibility mode of the MibSPI are directly applicable to a SPI. Multi-buffer Mode features are not available in Compatibility Mode.

NOTE: The SPIDAT0 register is not accessible in the multi-buffer mode of MibSPI. It is only accessible in compatibility mode.

27.1.3 Transmission Lock (Multi-Buffer Mode Master Only)

Some slave devices require transmission of a command followed by data. In this case the SPI transaction should not be interrupted by another group transfer. The LOCK bit within each buffer allows a consecutive transfer to happen without being interrupted by another higher-priority group transfer.

27.2 Operating Modes

The SPI can be configured via software to operate as either a master or a slave. The MASTER bit (SPIGCR1[0]) selects the configuration of the SPISIMO and SPISOMI pins. CLKMOD bit (SPIGCR1[1]) determines whether an internal or external clock source will be used.

The slave chip select ($\overline{\text{SPICS}}$) pins are used when communicating with multiple slave devices or, with a single slave, to delimit messages containing a leading register address. When a write occurs to SPIDAT1 in master mode, the $\overline{\text{SPICS}}$ pins are automatically driven to select the specified slave.

Handshaking mechanism, provided by the $\overline{\text{SPIENA}}$ pin, enables a slave SPI to delay the generation of the clock signal supplied by the master if it is not prepared for the next exchange of data.

NOTE: If in the slave mode of operation and configured in either 3-pin or 4-pin (without $\overline{\text{SPIENA}}$) modes, there must be a minimum of 8 VCLK cycles of delay between the last SPICLK and the start of the SPICLK for the next buffer transmit. In general, this equates to a VCLK/SPICLK ratio of ≤16 requiring a minimum of 1 SPICLK delay between transmissions.

27.2.1 Pin Configurations

The SPI supports data connections as shown in Table 27-1.

NOTE:

1. When the SPICS signals are disabled, the chip select field in the transmit data is not used.
2. When the $\overline{\text{SPIENA}}$ signal is disabled, the $\overline{\text{SPIENA}}$ pin is ignored in master mode, and not driven as part of the SPI transaction in slave mode.

Table 27-1. Pin Configurations

Pin	Master Mode		Slave Mode	
SPICLK	Drives the clock to external devices		Receives the clock from the external master	
SPISOMI	Receives data from the external slave		Sends data to the external master	
SPISIMO	Transmits data to the external slave		Receives data from the external master	
$\overline{\text{SPIENA}}$	**$\overline{\text{SPIENA}}$ disabled:** GIO	**$\overline{\text{SPIENA}}$ enabled:** Receives ENA signal from the external slave	**$\overline{\text{SPIENA}}$ disabled:** GIO	**$\overline{\text{SPIENA}}$ enabled:** Drives ENA signal from the external master
$\overline{\text{SPICS}}$	**$\overline{\text{SPICS}}$ disabled:** GIO	**$\overline{\text{SPICS}}$ enabled:** Selects one or more slave devices	**$\overline{\text{SPICS}}$ disabled:** GIO	**$\overline{\text{SPICS}}$ enabled:** Receives the CS signal from the external master

27.2.2 Data Handling

Figure 27-1 shows the SPI transaction hardware. TXBUF and RXBUF are internal buffers that are intended to improve the overall throughput of data transfer. TXBUF is a transmit buffer, while RXBUF is a receive buffer.

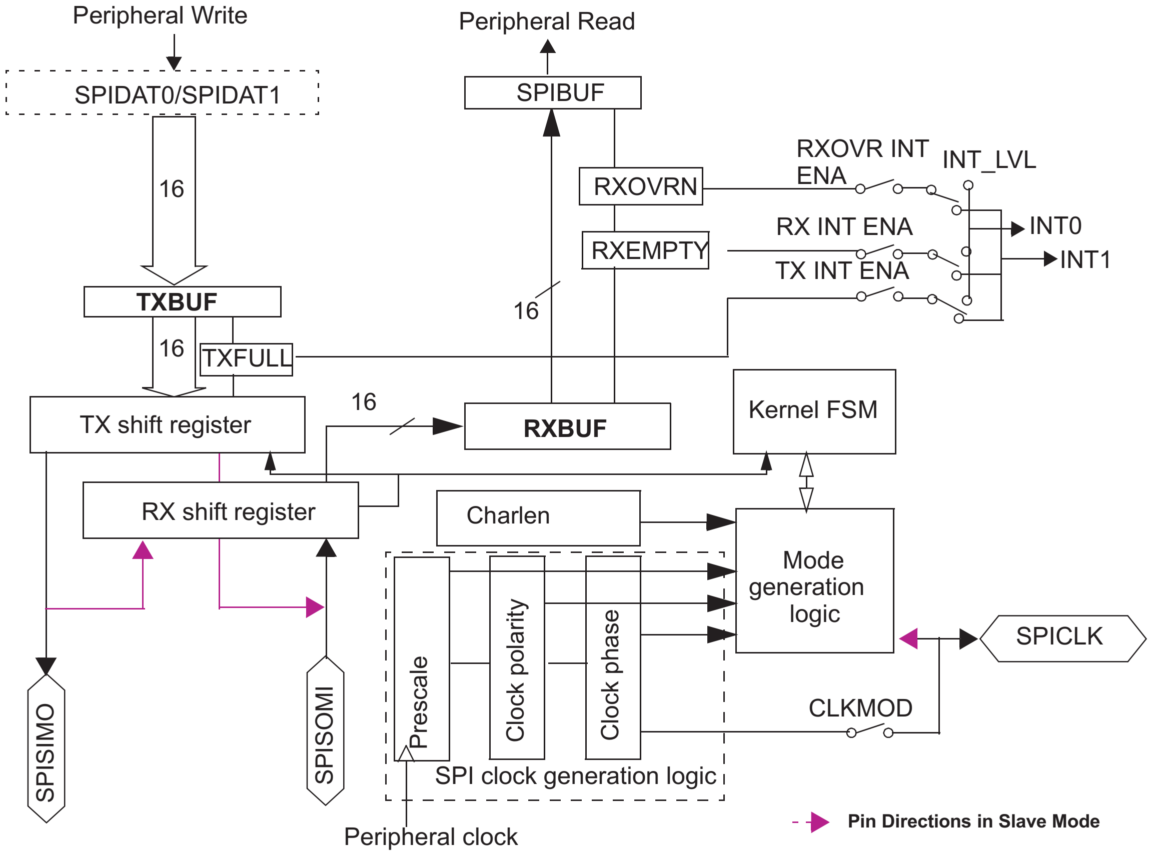
27.2.2.1 Data Sequencing when SPIDAT0 or SPIDAT1 is Written

- If both the TX shift register and TXBUF are empty, then the data is directly copied to the TX shift register. For devices with DMA, if DMA is enabled, a transmit DMA request (TX_DMA_REQ) is generated to cause the next word to be fetched. If transmit interrupts are enabled, a transmitter-empty interrupt is generated.
- If the TX shift register is already full or is in the process of shifting and if TXBUF is expty then the data written to SPIDAT0 / SPIDAT1 is copied to TXBUF and TXFULL flag is set to 1 at the same time.
- When a shift operation is complete, data from the TXBUF (if it is full) is copied into TX shift register and the TXFULL flag is cleared to 0 to indicate that next data can be fetched. A transmit DMA request (if enabled) or a transmitter-empty interrupt (if enabled) is generated at the same time.

27.2.2.2 Data Sequencing when All Bits Shifted into RXSHIFT Register

- If both SPIBUF and RXBUF are empty, the received data in RX shift register is directly copied into SPIBUF and the receive DMA request (if enabled) is generated and the receive-interrupt (if enabled) is generated. The RXEMPTY flag in SPIBUF is cleared at the same time.
- If SPIBUF is already full at the end of receive completion, the RX shift register contents is copied to RXBUF. A receive DMA request is generated, if enabled. The receive complete interrupt line remains high.
- If SPIBUF is read by the CPU or DMA and if RXBUF is full, then the contents of RXBUF are copied to SPIBUF as soon as SPIBUF is read. RXEMPTY flag remains cleared, indicating that SPIBUF is still full.
- If both SPIBUF and RXBUF are full, then RXBUF will be overwritten and the RXOVR interrupt flag is set and an interrupt is generated, if enabled.

Figure 27-1. SPI Functional Logic Diagram



1 This is a representative diagram, which shows three-pin mode hardware.
2 TXBUF, RXBUF, and SHIFT_REGISTER are user-invisible registers.
3 SPIDAT0 and SPIDAT1 are user-visible, and are physically mapped to the contents of TXBUF.
4 SPISIMO, SPISOMI, SPICLK pin directions depend on the Master or Slave Mode.

27.2.2.3 Three-Pin Mode

In master mode configuration (MASTER = 1 and CLKMOD = 1), the SPI provides the serial clock on the SPICLK pin. Data is transmitted on the SPISIMO pin and received on the SPISOMI pin (see Figure 27-2).

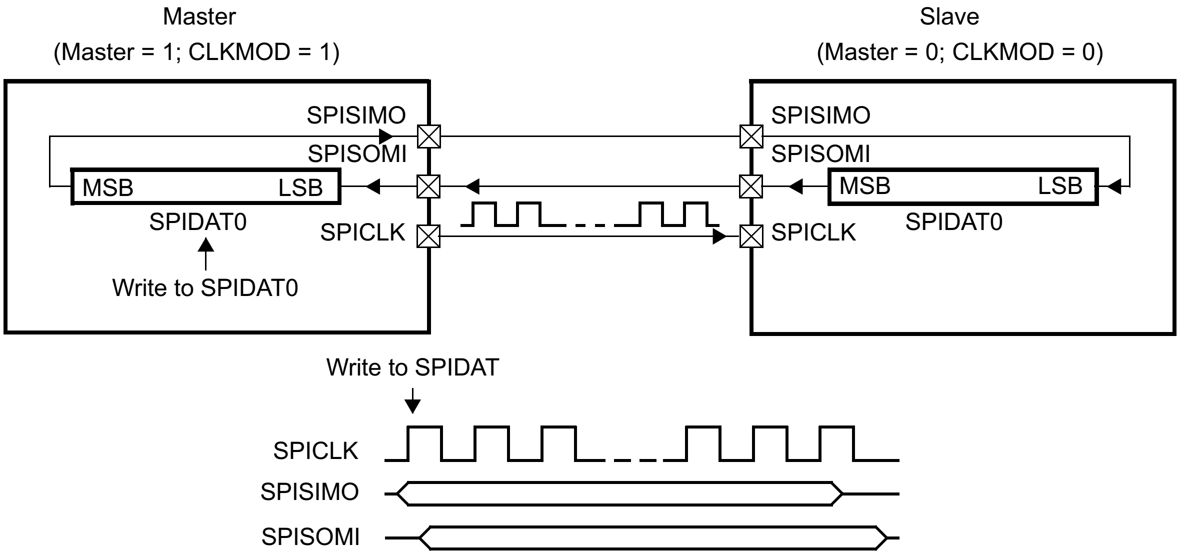
Data written to the shift register (SPIDAT0 / SPIDAT1) initiates data transmission on the SPISIMO pin, MSB first. Simultaneously, received data is shifted through the SPISOMI pin into the LSB of the SPIDAT0 register. When the selected number of bits have been transmitted, the received data in the shift register is transferred to the SPIBUF register for the CPU to read. Data is stored right-justified in SPIBUF.

See Section 27.2.2.1 and Section 27.2.2.2 for details about the data handling for transmit and receive operations.

In slave mode configuration (MASTER = 0 and CLKMOD = 0), data shifts out on the SPISOMI pin and in on the SPISIMO pin. The SPICLK pin is used as the input for the serial shift clock, which is supplied from the external network master. The transfer rate is defined by this clock.

Data written to the SPIDAT0 or SPIDAT1 register is transmitted to the network when the SPICLK signal is received from the network master. To receive data, the SPI waits for the network master to send the SPICLK signal and then shifts data on the SPISIMO pin into the RX shift register. If data is to be transmitted by the slave simultaneously, it must be written to the SPIDAT0 or SPIDAT1 register before the beginning of the SPICLK signal.

Figure 27-2. SPI Three-Pin Operation



27.2.3 Operation with $\overline{\text{SPICS}}$

In master mode, each chip select signal is used to select a specific slave. In slave mode, the chip select signal is used to enable and disable the transfer. Chip-select functionality is enabled by setting one of the $\overline{\text{SPICS}}$ pins as a chip select. It is disabled by setting all $\overline{\text{SPICS}}$ pins as GIOs in SPIPC0.

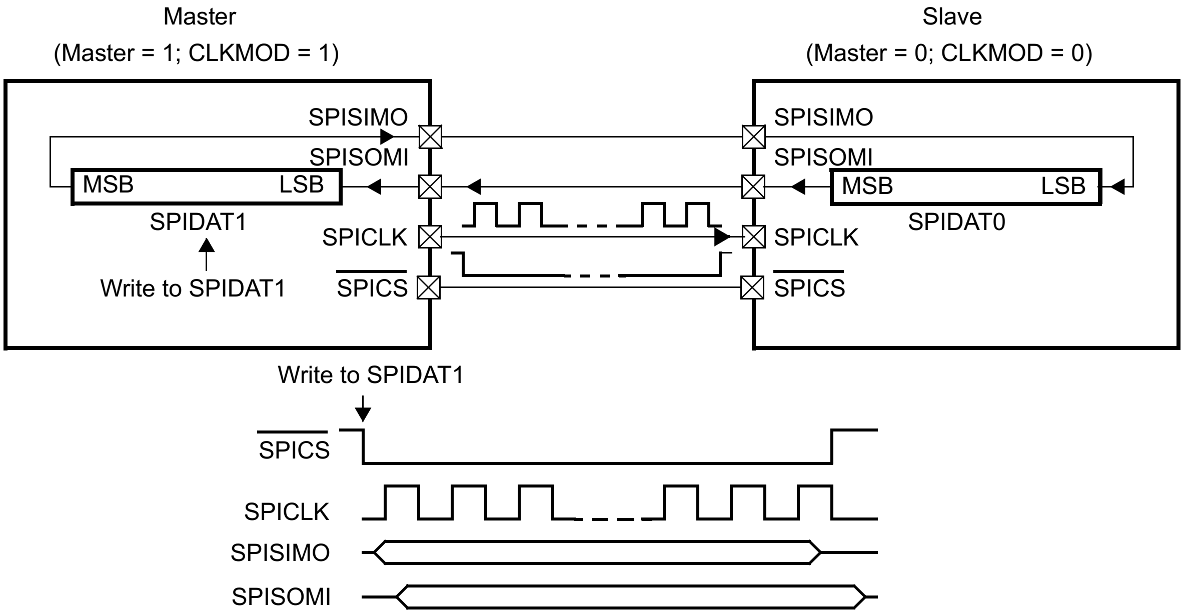
27.2.3.1 Multiple Chip Selects

The $\overline{\text{SPICS}}$ pins that are used must be configured as functional pins in the SPIPC0 register. The default pattern to be put on the $\overline{\text{SPICS}}$ when all the slaves are deactivated is set in the SPIDEF register. This pattern allows different slaves with different chip-select polarity to be activated by the SPI.

The master-mode SPI is capable of driving either 0 or 1 as the active value for any $\overline{\text{SPICS}}$ output pin. The drive state for the $\overline{\text{SPICS}}$ pins is controlled by the CSNR field of SPIDAT1. The pattern that is driven will select the slave to which the transmission is dedicated.

In slave mode, the SPI can only be selected by an active value of 0 on any of its selected $\overline{\text{SPICS}}$ input pins.

Figure 27-3. Operation with $\overline{\text{SPICS}}$



27.2.4 Operation with SPIENA

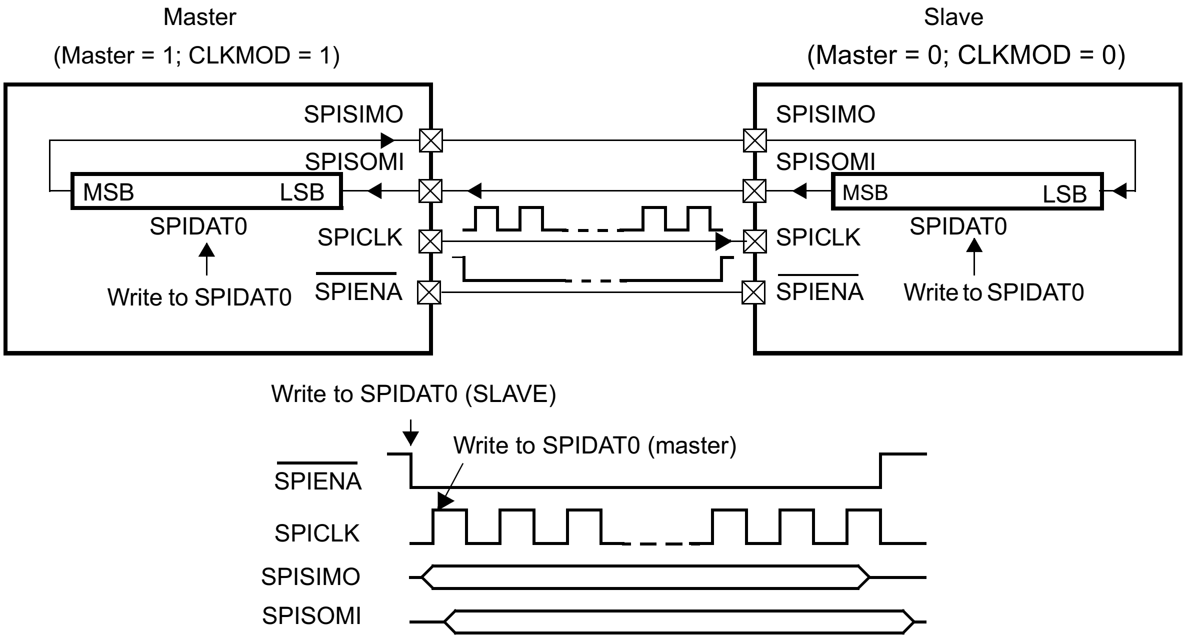
The SPIENA operates as a WAIT signal pin. For both the slave and the master, the SPIENA pin must be configured to be functional (SPIPC0[8] = 1). In this mode, an active-low signal from the slave on the SPIENA pin allows the master SPI to drive the clock pulse stream. A high signal tells the master to hold the clock signal (and delay SPI activity).

If the SPIENA pin is in high-impedance mode (ENABLE_HIGHZ = 1), the slave will put SPIENA into the high-impedance once it completes receiving a new character. If the SPIENA pin is in push-pull mode (ENABLE_HIGHZ = 0), the slave will drive SPIENA to 1 once it completes receiving a new character. The slave will drive SPIENA low again for the next word to transfer, after new data is written to the slave TX shift register.

In master mode (CLKMOD = 1), if the SPIENA pin is configured as functional, then the pin acts as an input pin. If configured as a slave SPI and as functional, the SPIENA pin acts as an output pin.

NOTE: During a transfer, if a slave-mode SPI detects a deassertion of its chip select before its internal character length counter overflows, then it places SPISOMI and SPIENA (if ENABLE_HIGHZ bit is set to 1) in high-impedance mode. Once this condition has occurred, if a SPICLK edge is detected while the chip select is deasserted, then the SPI stops that transfer and sets an DLENERR error flag and generates an interrupt (if enabled).

Figure 27-4. Operation with SPIENA



27.2.5 Five-Pin Operation (Hardware Handshaking)

Five-pin operation combines the functionality of three-pin mode, plus the enable pin and one or more chip select pins. The result is full hardware handshaking. To use this mode, both the SPIENA pin and the required number of SPICS pins must be configured as functional pins.

If the SPIENA pin is in high-impedance mode (ENABLE_HIGHZ = 1), the slave SPI will put this signal into the high-impedance state by default. The slave will drive the signal SPIENA low when new data is written to the slave shift register and the slave has been selected by the master (SPICS is low).

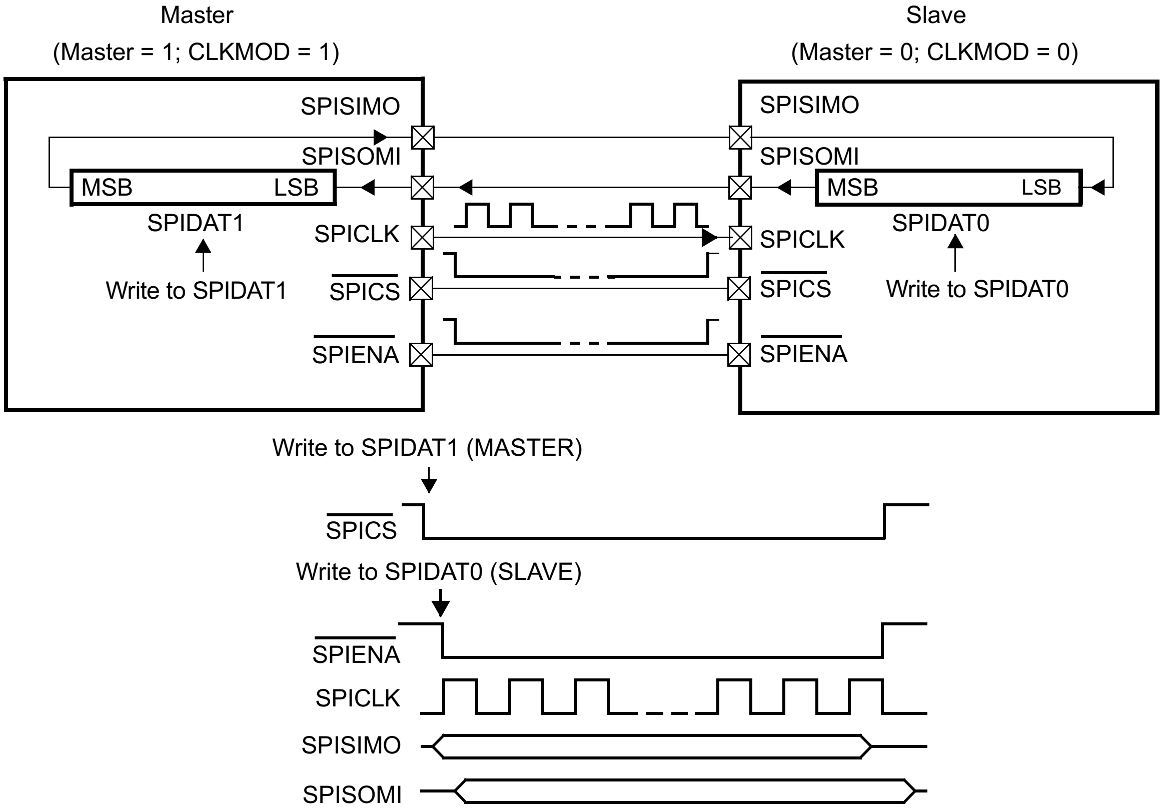
If the SPIENA pin is in push-pull mode (ENABLE_HIGHZ = 0), the slave SPI drives this pin high by default when it is in functional mode. The slave SPI will drive the SPIENA signal low when new data is written to the slave shift register (SPIDAT0/SPIDAT1) and the slave is selected by the master (SPICS is low). If the slave is deselected by the master (SPICS goes high), the slave SPIENA signal is driven high.

NOTE: Push-pull mode of the SPIENA pin can be used only when there is a single slave in the system. When multiple SPI slave devices are connected to the common SPIENA pin, all of the slaves should configure their SPIENA pins in high-impedance mode.

In master mode, if the SPICS pins are configured as functional pins, then the pins will be in output mode. A write to the master's SPIDAT1/SPIDAT0 register will automatically drive the SPICS signals low. The master will drive the SPICS signals high again after completing the transfer of the bits of the data.

In slave mode (CLKMOD = 0), the SPICS pins act as SPI functional inputs.

Figure 27-5. SPI Five-Pin Option with SPIENA and SPICS



27.2.6 Data Formats

To support multiple different types of slaves in one SPI network, four independent data word formats are implemented that allow configuration of individual data word length, polarity, phase, and bit rate. Each word transmitted can select which data format to use via the bits DFSEL[1:0] in its control field from one of the four data word formats. Same data format can be supported on multiple chip selects.

Data formats 0, 1, 2, and 3 can be configured through SPIFMTx control registers.

Each SPI data format includes the standard SPI data format with enhanced features:

- Individually-configurable shift direction can be used to select MSB first or LSB first, whereas the position of the MSB depends on the configured data word length.
- Receive data is automatically right-aligned, independent of shift direction and data word length. Transmit data has to be written right-aligned into the SPI and the internal shift register will transmit according to the selected shift direction and data word length for correct transfer.
- To increase fault detection of data transmission and reception, an odd or even parity bit can be added at the end of a data word. The parity generator can be enabled or disabled individually for each data format. If a received parity bit does not match with the locally calculated parity bit, the parity error flag (PARITYERR) is set and an interrupt is asserted (if enabled).

Since the master-mode SPI can drive two consecutive accesses to the same slave, an 8-bit delay counter is available to satisfy the delay time for data to be refreshed in the accessed slave. The delay counter can be programmed as part of the data format.

CHARLEN[4:0] specifies the number of bits (2 to 16) in the data word. The CHARLEN[4:0] value directs the state control logic to count the number of bits received or transmitted to determine when a complete word is transferred.

Data word length **must** be programmed to the same length for both the **master** and the **slave**. However, when chip selects are used, there may be multiple targets with different lengths in the system.

NOTE: Data must be right-justified when it is written to the SPI for transmission irrespective of its character length or word length.

Figure 27-6 shows how a 12-bit word (0xEC9) needs to be written to the transmit buffer to be transmitted correctly.

Figure 27-6. Format for Transmitting an 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

NOTE: The received data is always stored right-justified regardless of the character length or direction of shifting and is padded with leading 0s when the character length is less than 16 bits.

Figure 27-7 shows how a 10-bit word (0x0A2) is stored in the buffer once it is received.

Figure 27-7. Format for Receiving an 10-Bit Word

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0

27.2.7 Clocking Modes

SPICLK may operate in four different modes, depending on the choice of phase (delay/no delay) and the polarity (rising edge/falling edge) of the clock.

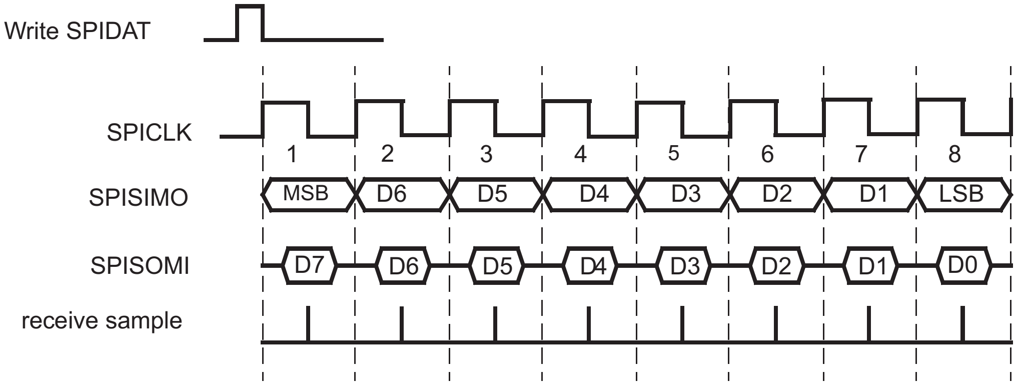
The data input and output edges depend on the values of both POLARITY and PHASE as shown in Table 27-2.

Table 27-2. Clocking Modes

POLARITY	PHASE	Action
0	0	Data is output on the rising edge of SPICLK. Input data is latched on the falling edge.
0	1	Data is output one half-cycle before the first rising edge of SPICLK and on subsequent falling edges. Input data is latched on the rising edge of SPICLK.
1	0	Data is output on the falling edge of SPICLK. Input data is latched on the rising edge.
1	1	Data is output one half-cycle before the first falling edge of SPICLK and on subsequent rising edges. Input data is latched on the falling edge of SPICLK.

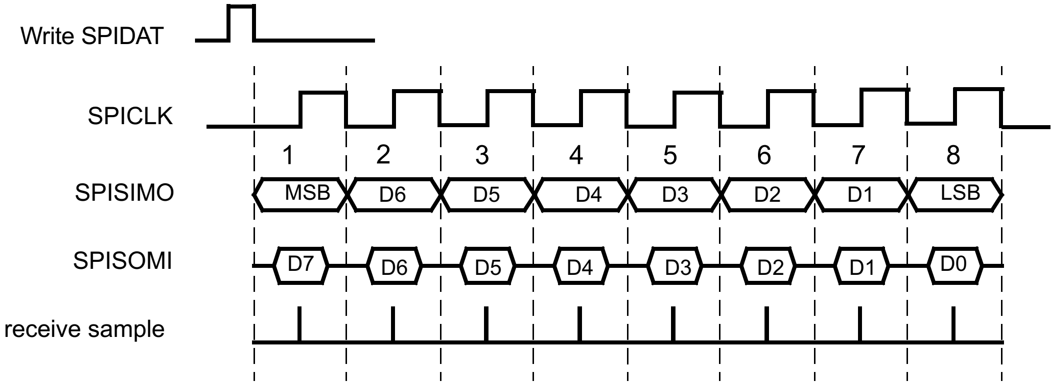
Figure 27-8 to Figure 27-11 illustrate the four possible configurations of **SPICLK** corresponding to each mode. Having four signal options allows the SPI to interface with many different types of serial devices.

Figure 27-8. Clock Mode with Polarity = 0 and Phase = 0



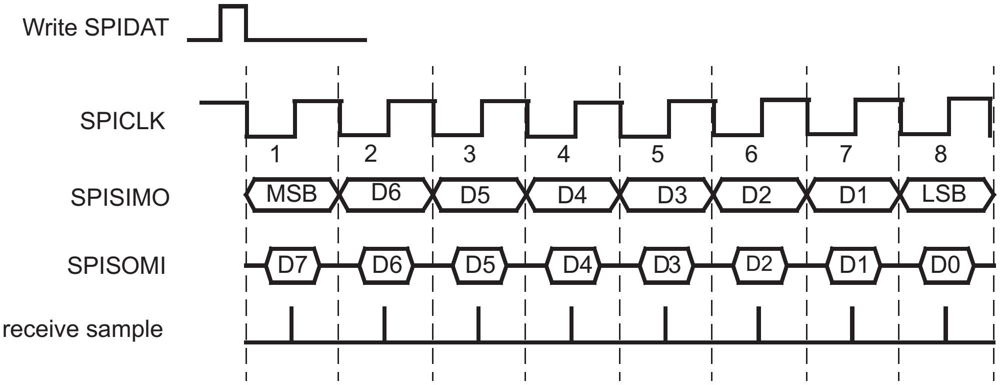
Data is output on the rising edge of SPICLK.
Input data is latched on the falling edge of SPICLK.

Figure 27-9. Clock Mode with Polarity = 0 and Phase = 1



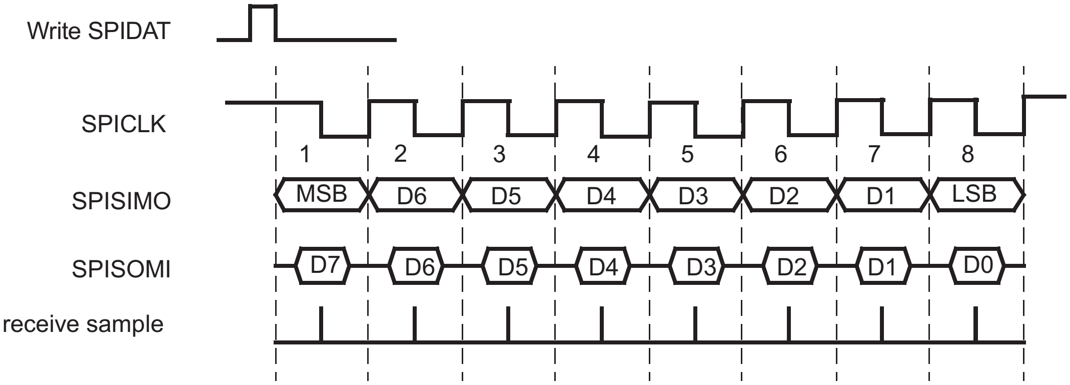
Data is output one-half cycle before the first rising edge of SPICLK and on subsequent falling edges of SPICLK
Input data is latched on the rising edge of SPICLK

Figure 27-10. Clock Mode with Polarity = 1 and Phase = 0



Data is output on the falling edge of SPICLK.
Input data is latched on the rising edge of SPICLK.

Figure 27-11. Clock Mode with Polarity = 1 and Phase = 1

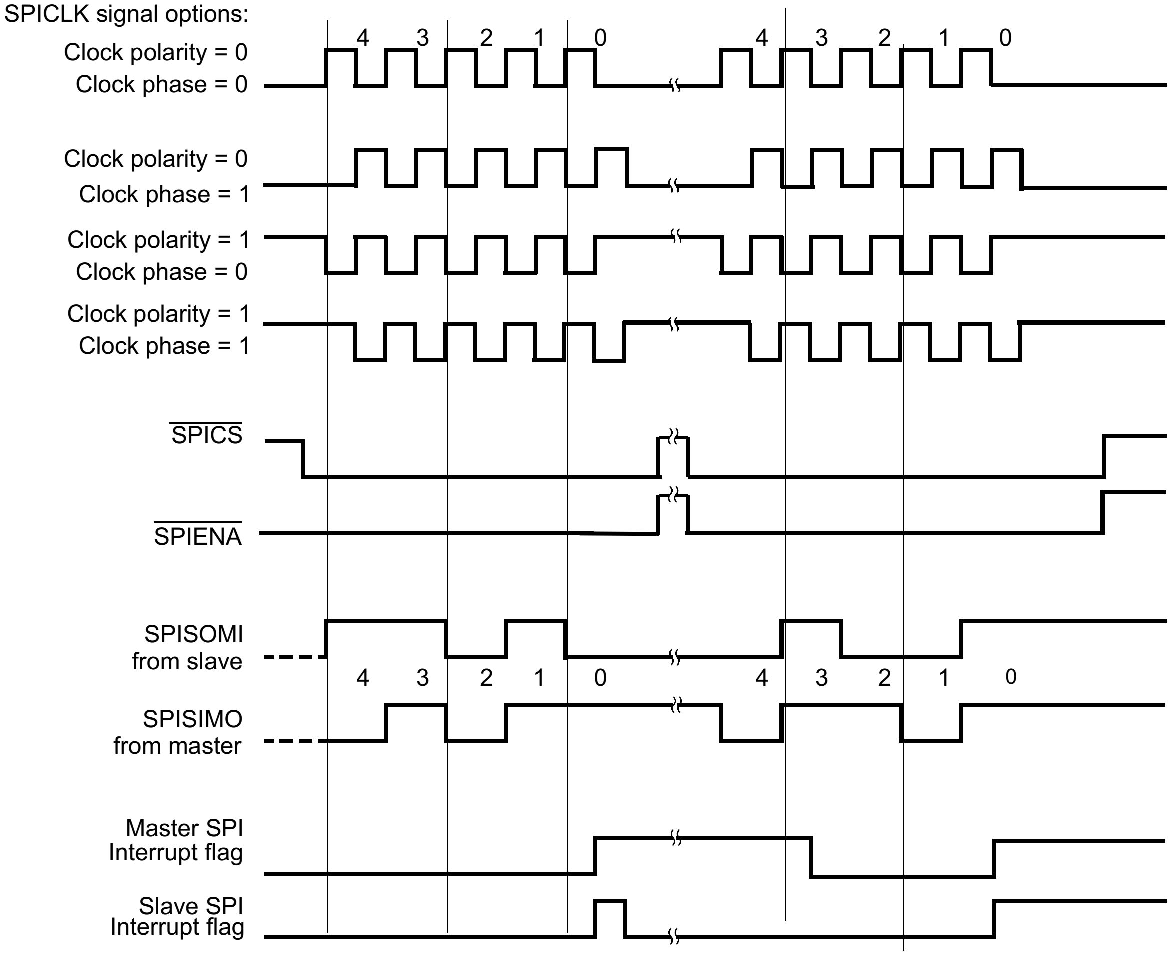


Data is output one-half cycle before the first falling edge of SPICLK and on the subsequent rising edges of SPICLK.
Input data is latched on the falling edge of SPICLK.

27.2.8 Data Transfer Example

Figure 27-12 illustrates a SPI data transfer between two devices using a character length of five bits.

Figure 27-12. Five Bits per Character (5-Pin Option)



27.2.9 Decoded and Encoded Chip Select (Master Only)

In this device, the SPI can connect to up to 6 individual slave devices using chip-selects by routing one wire to each slave. The 6 chip selects in the control field are directly connected to the 6 pins. The default value of each chip select (not active) can be configured via the register CSDEF. During a transmission, the value of the chip select control field (CSNR) of the SPIDAT1 register (SPIDAT1) is driven on the $\overline{\text{SPICS}}$ pins. When the transmission finishes, the default chip-select value (defined by the CSDEF register) is put on the $\overline{\text{SPICS}}$ pins.

The SPI can support more than 6 slaves by using encoded chip selects. To connect the SPI with encoded slaves devices, the CSNR field allows multiple active $\overline{\text{SPICS}}$ pins at the same time, which enables encoded chip selects from 0 to 16. To use encoded chip selects, all 6 chip select lines have to be connected to each slave device and each slave needs to have a unique chip-select address. The CSDEF register is used to provide the address at which slaves devices are all de-selected.

Users can combine decoded and encoded chip selects. For example, *n* $\overline{\text{SPICS}}$ pins can be used for encoding an *n*-bit address and the remaining pins can be connected to decoded-mode slaves.

27.2.10 Variable Chip Select Setup and Hold Timing (Master Only)

In order to support slow slave devices, a delay counter can be configured to delay data transmission after the chip select is activated. A second delay counter can be configured to delay the chip select deactivation after the last data bit is transferred. Both delay counters are clocked with the peripheral clock (VCLK).

If a particular data format specifically does not require these additional set-up or hold times for the chip select pins, then they can be disabled in the corresponding SPIFMTx register.

27.2.11 Hold Chip-Select Active

Some slave devices 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.

CSHOLD is programmable in both master and slave modes of the multi-buffer mode of SPI. However, the meaning of CSHOLD in master mode and slave mode are different.

NOTE: If the CSHOLD bit is set within the current data control field, the programmed hold time and the following programmed set-up time will not be applied between transactions.

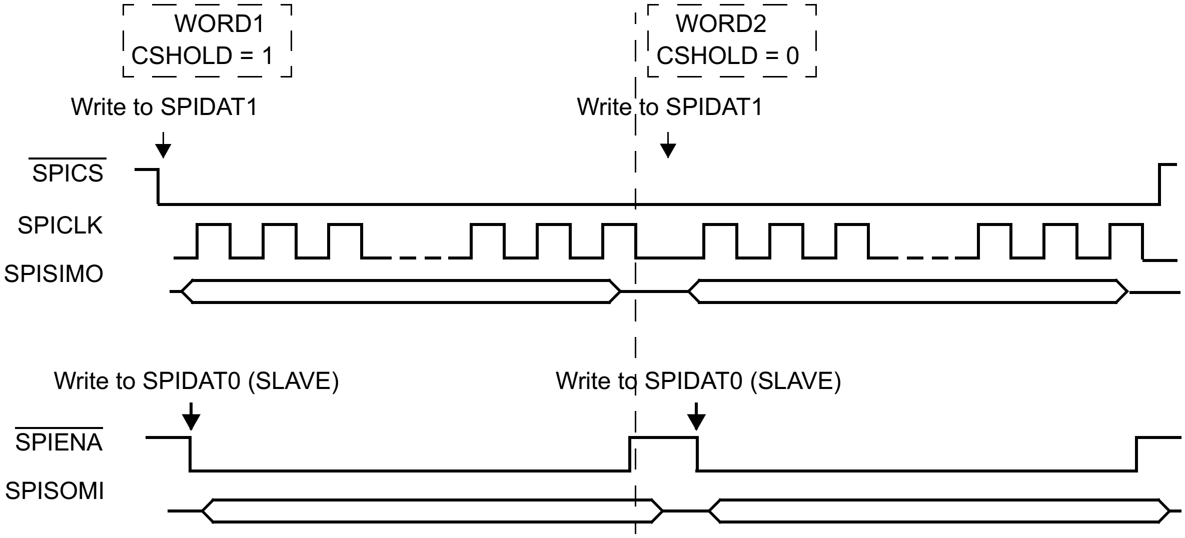
27.2.11.1 CSHOLD Bit in Master Mode

Each word in a master-mode SPI can be individually initialized for one of the two modes via the CSHOLD bit in its control field.

If the CSHOLD bit is set in the control field of a word, the chip select signal will not be deactivated until the next control field is loaded with new chip select information. Since the chip-select is maintained active between two transfers, the chip-select hold delay (T2CDELAY) is not applied at the end of the current transaction, and the chip-select set-up time delay (C2TDELAY) is not applied as well at the beginning of the following transaction. However, the wait delay (WDELAY) will be still applied between the two transactions, if the WDEL bit is set within the control field.

Figure 27-13 shows the SPI pins when a master-mode SPI transfers a word that has its CSHOLD bit set. The chip-select pins will not be deasserted after the completion of this word. If the next word to transmit has the same chip-select number (CSNR) value, the chip select pins will be maintained until the completion of the second word, regardless of whether the CSHOLD bit is set or not.

Figure 27-13. Typical Diagram when a Buffer in Master is in CSHOLD Mode (SPI-SPI)



27.2.11.2 CSHOLD Bit in Slave Mode (Multi-buffered Mode)

If the CSHOLD bit in a buffer is set to 1, then the MibSPI does not wait for the SPICS pins to be de-activated at the end of the shift operation to copy the received data to the receive RAM. With this feature, it is possible for a slave in multi-buffer mode to do multiple data transfers without requiring the SPICS pins to be deasserted between two buffer transfers.

If the CSHOLD bit in a buffer is cleared to 0 in a slave MibSPI, even after the shift operation is done, the MibSPI waits until the SPICS pin (if functional) is deasserted to copy the received data to the RXRAM.

If the CSHOLD bit is maintained as 0 across all the buffers, then the slave in multi-buffer mode requires its SPICS pins to be deasserted between any two buffer transfers; otherwise, the Slave SPI will be unable to respond to the next data transfer.

NOTE: In compatibility mode, the slave does not require the SPICS pin to be deasserted between two buffer transfers. The CSHOLD bit of the slave will be ignored in compatibility mode.

27.2.12 Detection of Slave Desynchronization (Master Only)

When a slave supports generation of an enable signal (ENA), desynchronization can be detected. With the enable signal a slave indicates to the master that it is ready to exchange data. A desynchronization can occur if one or more clock edges are missed by the slave. In this case, the slave may block the SOMI line until it detects clock edges corresponding to the next data word. This would corrupt the data word of the desynchronized slave and the consecutive data word. A configurable 8-bit time-out counter (T2EDELAY), which is clocked with SPICLK, is implemented to detect this slave malfunction. After the transmission has finished (end of last bit transferred: either last data bit or parity bit) the counter is started. If the ENA signal generated by the slave does not become inactive before the counter overflows, the DESYNC flag is set and an interrupt is asserted (if enabled).

NOTE: **Inconsistency of Desynchronization Flag in Compatibility Mode MibSPI**

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 due to the fact that 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 VBUS master reads the received data quickly when an RXINT is detected, then the status flag may not reflect the correct desync condition. This inconsistency in the desync flag is valid only in compatibility mode of MibSPI. In multi-buffer mode, the desync flag is always assured to be for the current buffer.

27.2.13 ENA Signal Time-Out (Master Only)

The SPI in master mode waits for the hardware handshake signal (ENA) coming from the addressed slave before performing a data transfer. To avoid stalling the SPI by a non-responsive slave device, a time-out value can be configured using C2EDELAY. If the time-out counter overflows before an active ENA signal is sampled, the TIMEOUT flag in the status register SPIFLG is set and the TIMEOUT flag in the status field of the corresponding buffer is set.

NOTE: When the chip select signal becomes active, no breaks in transmission are allowed. The next arbitration is performed while waiting for the time-out to occur.

27.2.14 Data-Length Error

A SPI can generate an error flag by detecting any mismatch in length of received or transmitted data and the programmed character length under certain conditions.

Data-Length Error in Master Mode: During a data transfer, if the SPI detects a de-assertion of the $\overline{\text{SPIENA}}$ pin (by the slave) while the character counter is not overflowed, then an error flag is set to indicate a data-length error. This can be caused by a slave receiving extra clocks (for example, due to noise on the SPICLK line).

NOTE: In a master mode SPI, the data length error will be generated only if the $\overline{\text{SPIENA}}$ pin is enabled as a functional pin.

Data-Length Error in Slave Mode: During a transfer, if the SPI detects a de-assertion of the $\overline{\text{SPICS}}$ pin before its character length counter overflows, then an error flag is set to indicate a data-length error. This situation can arise If the slave SPI misses one or more SPICLK pulses from the master. This error in slave mode implies that both the transmitted and received data were not complete.

NOTE: In a slave mode SPI, the data-length error flag will be generated only if at least one of the $\overline{\text{SPICS}}$ pins are configured as functional, and are being used for selecting the slave.

27.2.15 Parallel Mode (Multiple SIMO/SOMI Support, not available on all devices)

In order to increase throughput, the parallel mode of the SPI enables the module to send data over more than one data line (parallel 2, 4, or 8). When parallel mode is used, the data length must be set as 16 bits. Only module MIBSPIP5 supports Parallel Mode.

This feature increases throughput by 2 for 2 pins, by 4 for 4 pins, or by 8 for 8 pins.

Parallel mode supports the following features:

- Scalable data lines (1, 2, 4, 8) per direction. (SOMI and SIMO lines)
- All clock schemes are supported (clock phase and polarity)
- Parity is supported. The parity bit will be transmitted on bit0 of the SIMO/SOMI lines. The receive parity is expected on bit0 of the SOMI/SIMO pins.

Parallel mode can be programmed using the PMODEx bits of SPIPMCTRL register. See Section 27.9.25 for details about this register.

After reset the parallel mode selection bits are cleared (single SIMO/SOMI lines).

27.2.15.1 Parallel Mode Block Diagram

Figure 27-14 and Figure 27-15 show the parallel connections to the SPI shift register.

Figure 27-14. Block Diagram Shift Register, MSB First

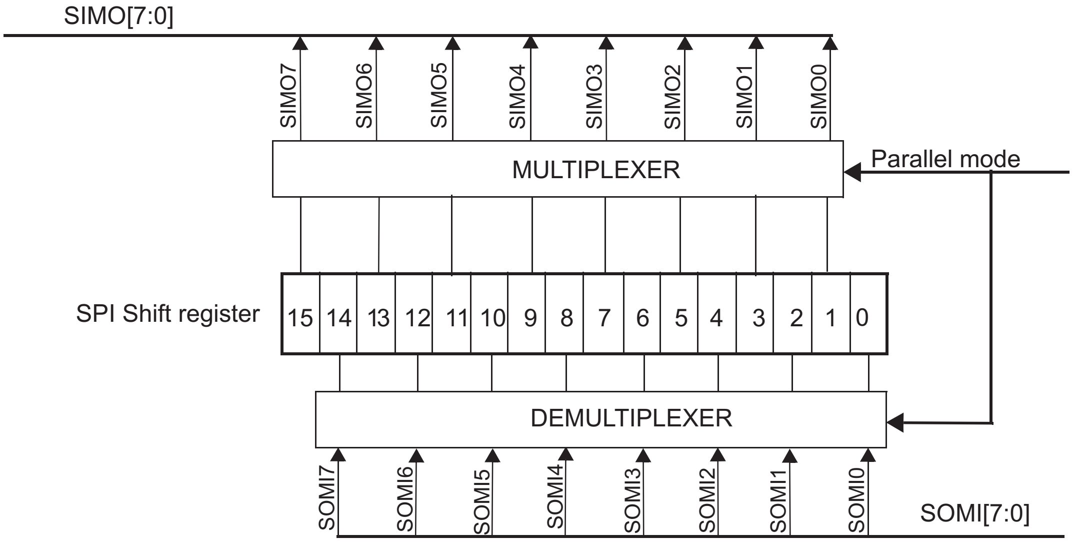
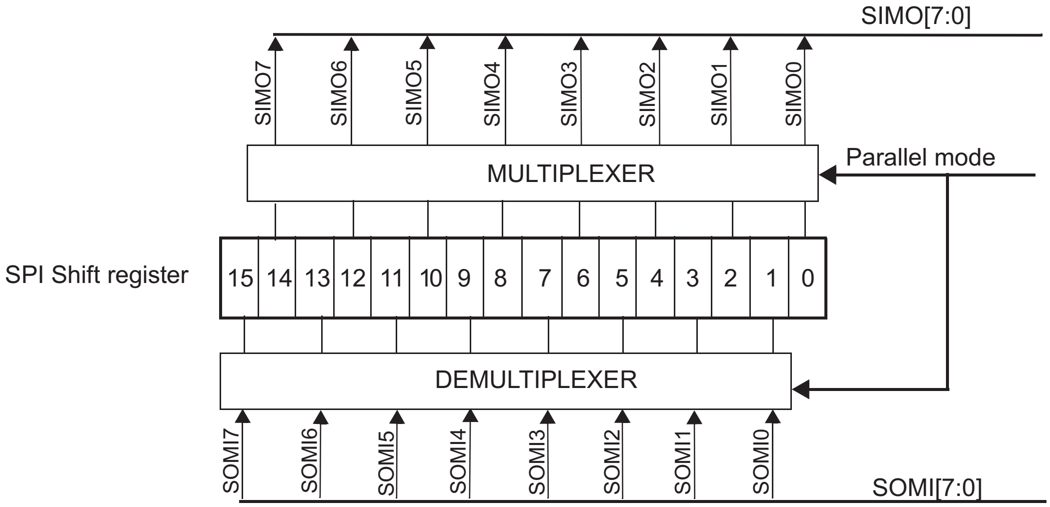


Figure 27-15. Block Diagram Shift Register, LSB First



27.2.15.2 Parallel Mode Pin Mapping, MSB First

Table 27-3 and Table 27-4 describe the SOMI and SIMO pin mapping when the SPI is used in parallel mode (1, 2, 4, 8) pin mode, MSB first.

NOTE: MSB-first or LSB-first can be configured using the SHIFTDIRx bit of the SPIFMTx registers.

Table 27-3. Pin Mapping for SIMO Pin with MSB First

Parallel Mode	Shift Register Bit	SIMO[7:0]
1	15	0
2	15	1
	7	0
4	15	3
	11	2
	7	1
	3	0
8	15	7
	13	6
	11	5
	9	4
	7	3
	5	2
	3	1
	1	0

Table 27-4. Pin Mapping for SOMI Pin with MSB First

Parallel Mode	Shift Register Bit	SOMI[7:0]
1	0	0
2	0	0
	8	1
4	0	0
	4	1
	8	2
	12	3
8	0	0
	2	1
	4	2
	6	3
	8	4
	10	5
	12	6
	14	7

27.2.15.3 Parallel Mode Pin Mapping, MSB-First, LSB-First

Table 27-5 and Table 27-6 describe the SIMO and SOMI pin mapping when SPI is used in parallel mode (1, 2, 4, 8) pin mode, LSB first.

Table 27-5. Pin Mapping for SIMO Pin with LSB First

Parallel Mode	Shift Register Bit	SIMO[7:0]
1	0	0
2	8	1
	0	0
4	12	3
	8	2
	4	1
	0	0
8	14	7
	12	6
	10	5
	8	4
	6	3
	4	2
	2	1
	0	0

Table 27-6. Pin Mapping for SOMI Pin with LSB First

Parallel Mode	Shift Register Bit	SOMI[7:0]
1	15	0
2	7	0
	15	1
4	3	0
	7	1
	11	2
	15	3
8	1	0
	3	1
	5	2
	7	3
	9	4
	11	5
	13	6
	15	7

27.2.15.4 2-Data Line Mode (MSB First, Phase 0, Polarity 0)

In 2-data line mode (master mode) the shift register bits 15 and 7 will be connected to the pins SIMO[1] and SIMO[0], and the shift register bits 8 and 0 will be connected to the pins SOMI[1] and SOMI[0] or vice versa in slave mode. After writing to the SPIDAT0/SPIDAT1 register, the bits 15 and 7 will be output on SIMO[1] and SIMO[0] on the rising edge if SPICLK. With the falling clock edge of the SPICLK, the received data on SOMI[1] and SOMI[0] will be latched to the shift register bits 8 and 0. The subsequent rising edge of SPICLK will shift the data in the shift register by 1 bit to the left. (SIMO[1] will shift the data out from bit 15 to 8, SIMO[0] will shift the data out from bit 7 to 0). After eight SPICLK cycles, when the full data word is transferred, the shift register (16 bits) is copied to the receive buffer, and the RXINT flag will be set. Figure 27-16 shows the clock /data diagram of the 2-data line mode. Figure 27-17 shows the timing of a two-pin parallel transfer.

Figure 27-16. 2-data Line Mode (Phase 0, Polarity 0)

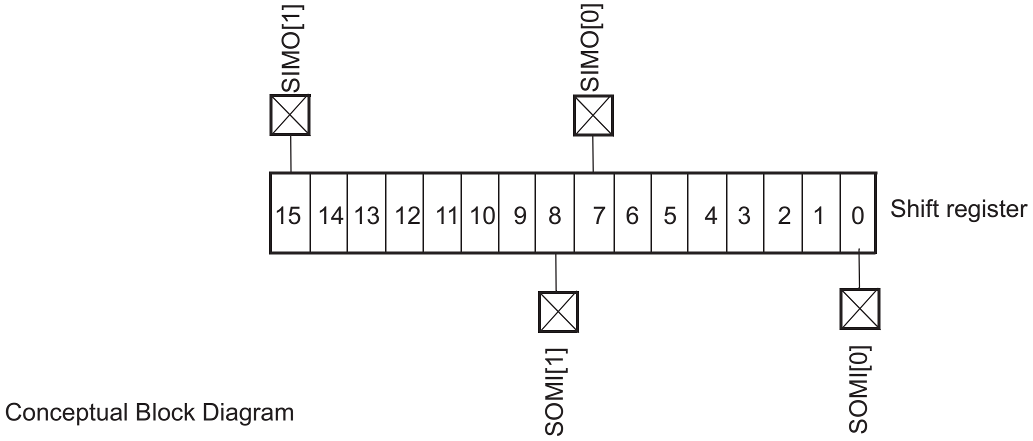
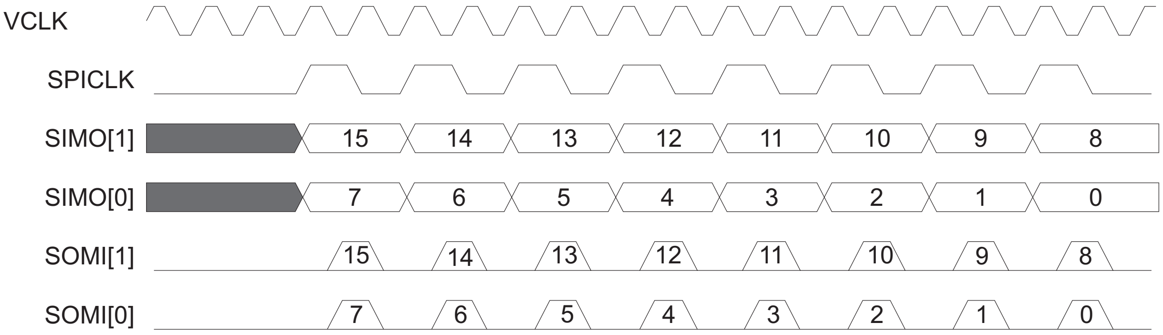


Figure 27-17. Two-Pin Parallel Mode Timing Diagram (Phase 0, Polarity 0)



27.2.15.5 4-Data Line Mode (MSB First, Phase 0, Polarity 0)

In 4-data line mode (master mode) the shift register bits 15, 11, 7, and 3 will be connected to the pins SIMO[3], SIMO[2], SIMO[1], and SIMO[0], and the shift register bits 12, 8, 4, and 0 will be connected to the pins SOMI[3], SOMI[2], SOMI[1], and SOMI[0] (or vice versa in slave mode). After writing to SPIDAT1/SPIDAT0, the bits 15, 11, 7, and 3 will be output on SIMO[3], SIMO[2], SIMO[1], and SIMO[0] on the rising edge of SPICLK. With the falling clock edge of the SPICLK, the received data on SOMI[3], SOMI[2], SOMI[1] and SOMI[0] will be latched to shift register bits 12, 8, 4, and 0. The subsequent rising edge of SPICLK will shift data in the shift register by 1 bit to the left (SIMO[3] will shift the data out from bit 15 to 12, SIMO[2] will shift the data out from bit 11 to 8, SIMO[1] will shift the data out from bit 7 to 4, SIMO[0] will shift the data out from bit 3 to 0). After four SPICLK cycles, when the full data word is transferred, the shift register (16 bits) is copied to the receive buffer, and the RXINT flag will be set.

Figure 27-18 shows the clock/data diagram of the four-data line mode. Figure 27-19, shows the timing diagram for four-data line mode.

Figure 27-18. 4-Data Line Mode (Phase 0, Polarity 0)

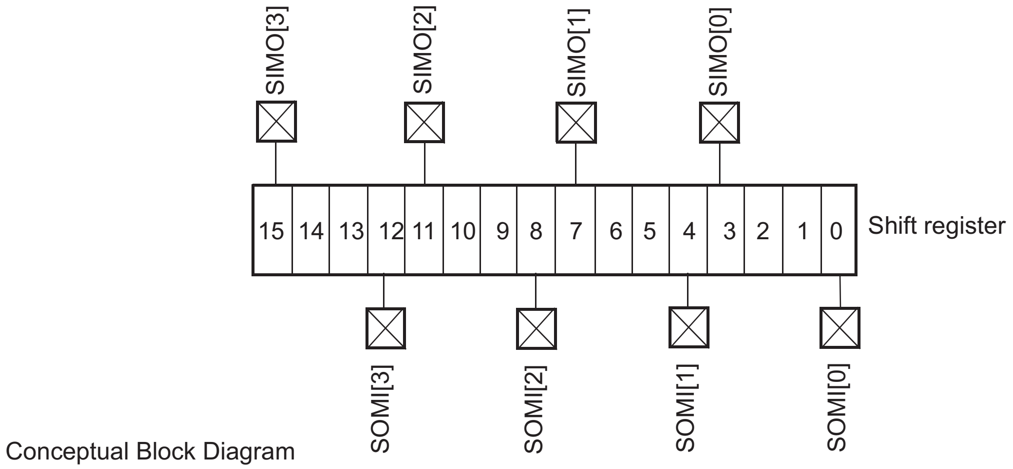
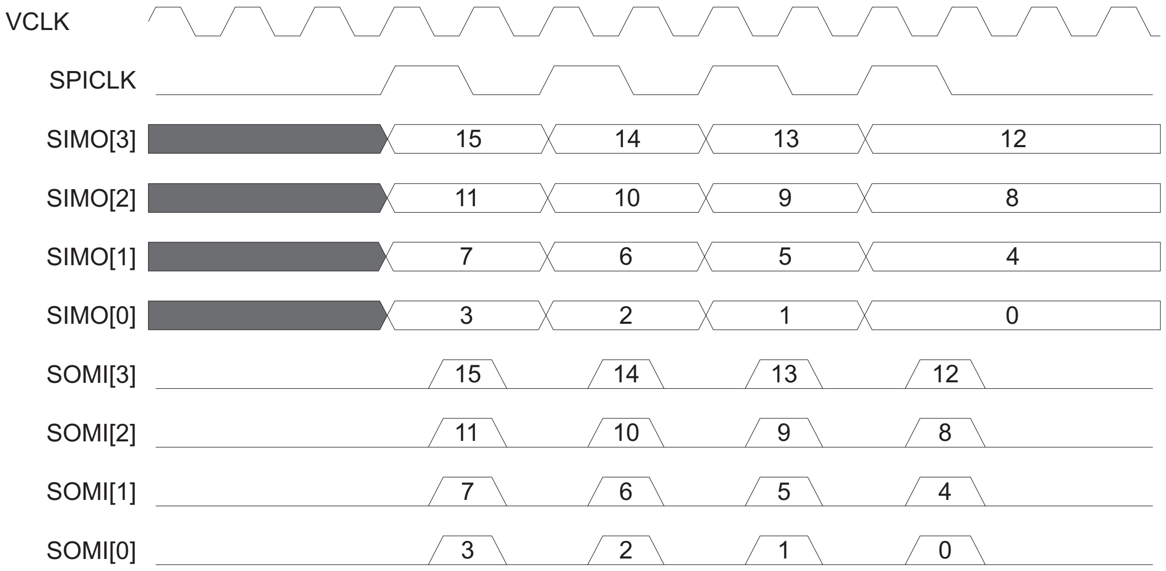


Figure 27-19. 4 Pins Parallel Mode Timing Diagram (Phase 0, Polarity 0)



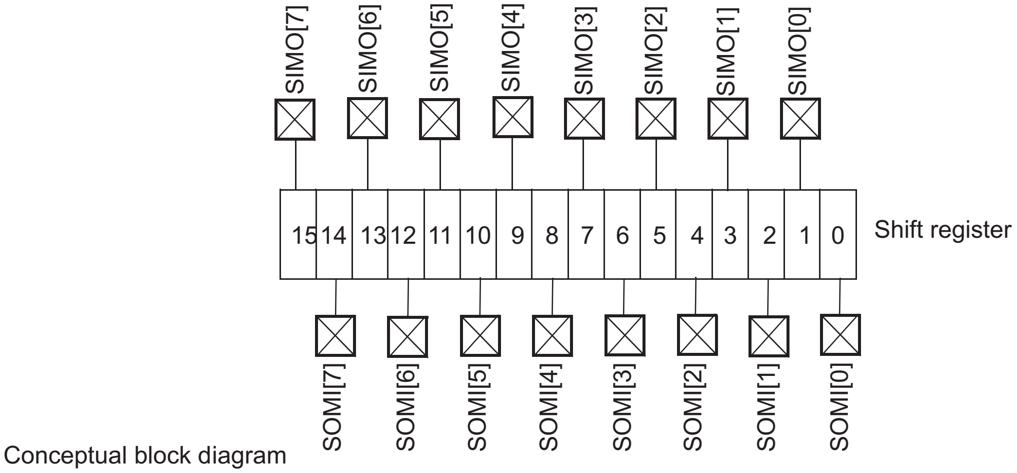
27.2.15.6 8-Data Line Mode (MSB First, Phase 0, Polarity 0)

In 8-data line mode (master mode) the shift register bits 15, 13, 11, 9, 7, 5 and 3 will be connected to the pins SIMO[7], SIMO[6], SIMO[5], SIMO[4], SIMO[3], SIMO[2], SIMO[1], and SIMO[0], and the shift-register bits 14, 12, 10, 8, 6, 4, and 0 will be connected to the pins SOMI[7], SOMI[6], SOMI[5], SOMI[4], SOMI[3], SOMI[2], SOMI[1], and SOMI[0] (or vice versa in slave mode).

After writing to SPIDAT0/SPIDAT1, the bits 15, 13, 11, 9, 7, 5, 3, and 1 will be output on SIMO[7], SIMO[6], SIMO[5], SIMO[4], SIMO[3], SIMO[2], SIMO[1], and SIMO[0], on the rising edge of SPICLK. On the falling clock edge of the SPICLK, the received data on SOMI[8], SOMI[7], SOMI[6],SOMI[5], SOMI[4], SOMI[3], SOMI[2], SOMI[1], and SOMI[0] will be latched to the shift register bits 14, 12, 10, 8, 6, 4, 2, and 0.

The subsequent rising edge of SPICLK will shift the data in the shift register by 1 bit to the left. After two SPICLK cycles, when the full data word is transferred the shift register (16 bits) is copied to the receive buffer, and the RXINT flag will be set. Figure 27-20 shows the clock/data diagram of the 8-data line mode. Figure 27-21 shows the pin timings for 8-data line mode.

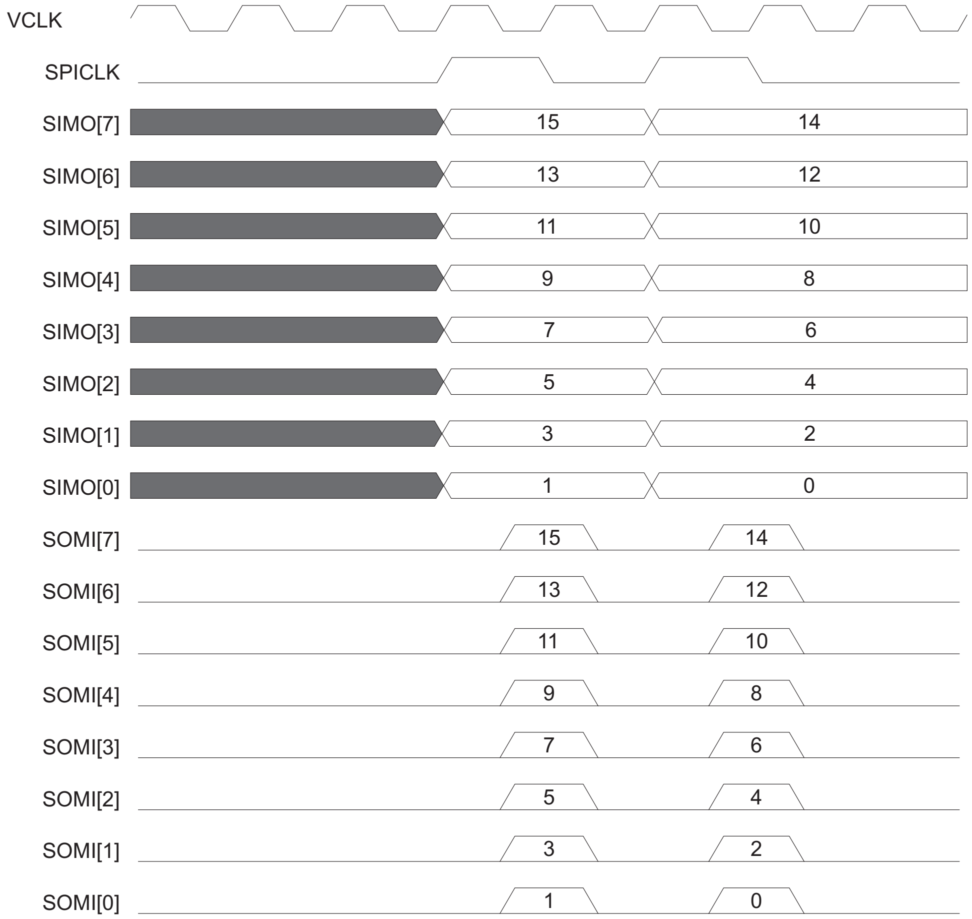
Figure 27-20. 8-data Line Mode (Phase 0, Polarity 0)



NOTE: Parity Support

Using the parity support in parallel mode may seriously affect throughput. For an eight-line mode to transfer 16 bits of data, only two SPICLK pulses are enough. If parity is enabled, one extra SPICLK pulse will be used to transfer and receive the parity bit. Parity will be transmitted and received on the 0th line regardless of 1/2/4/8-line modes. During the parity bit transfer, other data bits are not valid.

Figure 27-21. 8 Pins Parallel Mode Timing Diagram (Phase 0, Polarity 0)



NOTE: Modulo Count Parallel Mode is not supported in this device.

27.2.16 Continuous Self-Test (Master/Slave)

During data transfer, the SPI compares its own internal transmit data with its transmit data on the bus. The sample point for the compare is at one-half SPI clock after transmit point. If the data on the bus does not match the expected value, the bit-error (BITERR) flag is set and an interrupt is asserted if enabled.

NOTE: The compare is made from the output pin using its input buffer.

27.2.17 Half Duplex Mode

SPI by protocol is Full Duplex in nature, which means simultaneous TX and RX operations happen on two separate data pins, SIMO and SOMI. However, it is possible to use SPI/MibSPI to do the TX-only operation (ignoring the RX data) and the RX-only operation (using dummy TX data and ignoring the TX pin). But this requires that both SOMI and SIMO lines are bonded out in a chip to be able to support both TX-only or RX-only features.

27.2.17.1 Half Duplex Mode in Master

The Half Duplex Mode gives an additional flexibility to use the SIMO pin, which is normally used as a TX pin in Master mode, to work like an RX pin while the HDUPLEX_ENAx bit in SPIFMTx register is set to 1. In Half Duplex Master mode, the SIMO pin acts as an RX pin. Switching between Full Duplex and Half Duplex can be achieved using the SPIFMTx register being selected using the DFSEL bit of SPIDAT1 register or TXRAM locations.

27.2.17.2 Half Duplex Mode in Slave

In Half Duplex Slave mode, the SIMO pin, which is normally an RX pin, acts as a TX pin while the HDUPLEX_ENAx bit in SPIFMTx register is set to 1. In Half Duplex Slave mode, the SIMO pin acts as a TX pin. Switching between Full Duplex and Half Duplex can be achieved using the SPIFMTx register being selected using the DFSEL bit of SPIDAT1 register or TXRAM locations.

27.3 Test Features

27.3.1 Internal Loop-Back Test Mode (Master Only)

The internal loop-back self-test mode can be utilized to test the SPI transmit and receive paths, including the shift registers, the SPI buffer registers, and the parity generator. In this mode the transmit signal is internally feedback to the receiver, whereas the SIMO, SOMI, and CLK pin are disconnected; that is, the transmitted data is internally transferred to the corresponding receive buffer while external signals remain unchanged.

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.

NOTE: This mode cannot be changed during transmission.

27.3.2 Input/Output Loopback Test Mode

Input/Output Loopback Test mode supports the testing of all Input/Output pins without the aid of an external interface. Loopback can be configured as either analog-loopback (loopback through the pin-level input/output buffers) or digital loopback (internal to the SPI module). With Input/Output Loopback, all functional features of the SPI can be tested. Transmit data is fed back through the receive-data line(s). See Figure 27-22 for a diagram of the types of feedback available. The IOLPBKTSTCR register defines all of the available control fields.

In loopback mode, it is also possible to induce various error conditions. See Section 27.9.43 for details of the register field controlling these features.

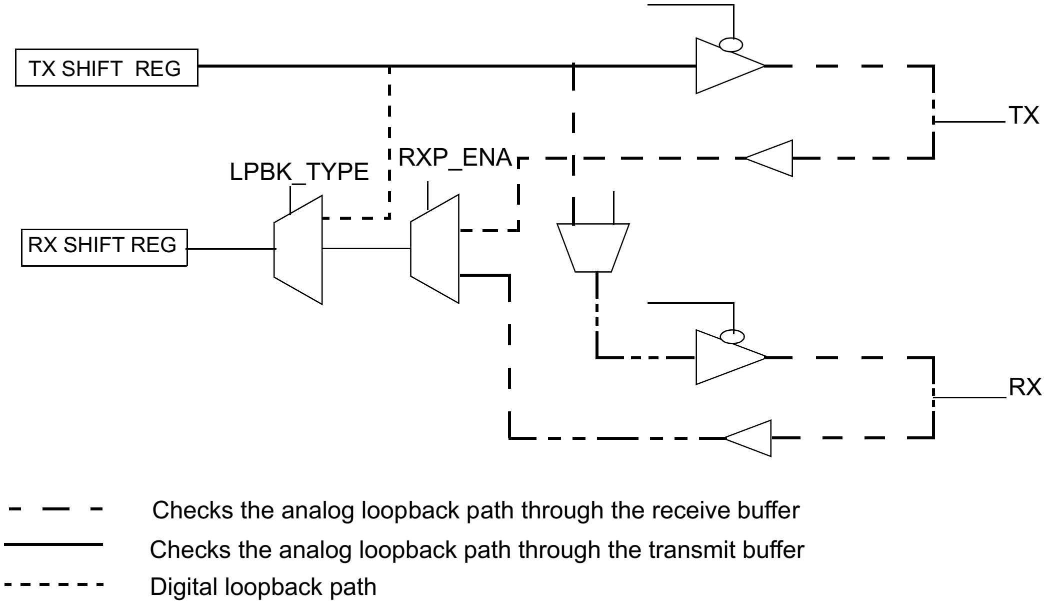
In Input/Output loopback test modes, even when the module is in slave mode, the SPICLK is generated internally. This SPICLK is used for all loopback-mode SPI transactions. Slave-mode features can be tested without the help of another master SPI, using the internally-generated SPICLK. Chip selects are also generated by the slave itself while it is in Input/Output loopback mode.

In Input/Output loopback test modes, if the module is in master mode, the $\overline{\text{ENA}}$ signal is also generated by internal logic so that an external interface is not required.

NOTE: Usage Guideline for Input/Output Loopback

Input/Output Loopback mode should be used with caution because, in some configurations, even the receive pins will be driven with transmit data. During testing, it should be ensured that none of the SPI pins are driven by any other device connected to them. Otherwise, if analog loopback is selected in I/O Loopback mode, then testing may damage the device.

Figure 27-22. I/O Paths during I/O Loopback Modes


TX SHIFT REG
TX
RXP_ENA
LPBK_TYPE
RX SHIFT REG
RX
Checks the analog loopback path through the receive buffer
Checks the analog loopback path through the transmit buffer
Digital loopback path

This diagram is intended to illustrate loopback paths and therefore may omit some normal-mode paths.

27.3.2.1 Input/Output Loopback Mode Operation in Slave Mode

In multi-buffer slave mode, there are some additional requirements for using I/O loopback mode (IOLPBK). In multi-buffer slave mode, the chip-select pins are the triggers for various TGs. Enabling the IOLPBK mode by writing 0xA to the IOLPBTSTENA bits of the IOLPBKTSTCR register triggers TG0 by driving $\overline{\text{SPICS}}$ to 0. The actual number of chip selects can be programmed to have any or all of the $\overline{\text{SPICS}}$ pins as functional. All other configurations should be completed before enabling the IOLPBK mode in multi-buffer slave mode since it triggers TG0.

After the first buffer transfer is completed, the CSNR field of the current buffer is used to trigger the next buffer. So, if multiple TGs are desired to be tested, then the CSNR field of the final buffer in each TG should hold the number of the next TG to be triggered. As long as TG boundaries are well defined and are enabled, the completion of one TG will trigger the next TG.

To stop the transfer in multi-buffer slave mode in I/O Loopback configuration, either IOLPBK mode can be disabled by writing 0x5 to the IOLPBTSTENA bits or all of the TGs can be disabled.

27.4 General-Purpose I/O

All of the SPI pins may be programmed via the SPIPCx control registers to be either functional or general-purpose I/O pins.

If the SPI function is to be used, application software must ensure that at least the SPICLK pin and the SOMI and/or SIMO pins are configured as SPI functional pins, and not as GIO pins, or else the SPI state machine will be held in reset, preventing SPI transactions.

SPI pins support:

- internal pull-up resistors
- internal pull-down resistors
- open-drain or push-pull mode

27.5 Low-Power Mode

The SPI can be put into 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, making the module completely inactive.

Local low-power mode is asserted by setting the POWERDOWN (SPIGCR1[8]) bit; setting this bit stops the clocks to the SPI internal logic and 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 registers remain accessible during local power-down mode, since the clock to the SPI registers is temporarily re-enabled for each access. RAM buffers are also accessible during low power mode.

NOTE: 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. Application software must ensure that a low power mode is not entered during a data transfer.

27.6 Interrupts

There are two levels of vectorized interrupts supported by the SPI. These interrupts can be caused under the following circumstances:

- Transmission error
- Receive overrun
- Receive complete (receive buffer full)
- Transmit buffer empty

These interrupts may be enabled or disabled via the SPIINT0 register.

During transmission, if one of the following errors occurs: BITERR, DESYNC, DLENERR, PARITYERR, or TIMEOUT, the corresponding bit in the SPIFLG register is set. If the corresponding enable bit is set, then an interrupt is generated. The level of all the above interrupts is set by the bit fields in the SPILVL register.

The error interrupts are enabled and prioritized independently from each other, but the interrupt generated will be the same if multiple errors are enabled on the same level. The SPIFLG register should be used to determine the actual cause of an error.

NOTE: Since there are two interrupt lines, one each for Level 0 and Level 1, it is possible for a programmer to separate out the interrupts for receive buffer full and transmit buffer empty. By programming one to Level 0 and the other to Level 1, it is possible to avoid a check on whether an interrupt occurred for transmit or for receive. A programmer can also choose to group all of the error interrupts into one interrupt line and both TX-empty and RX-full interrupts into another interrupt line using the LVL control register. In this way, it is possible to separate error-checking from normal data handling.

27.6.1 Interrupts in Multi-Buffer Mode

In multi-buffer mode, the SPI can generate interrupts on two levels.

In normal multi-buffer operation, the receive and transmit are not used and therefore the enable bits of SPIINT0 are not used.

The interrupts available in multi-buffer mode are:

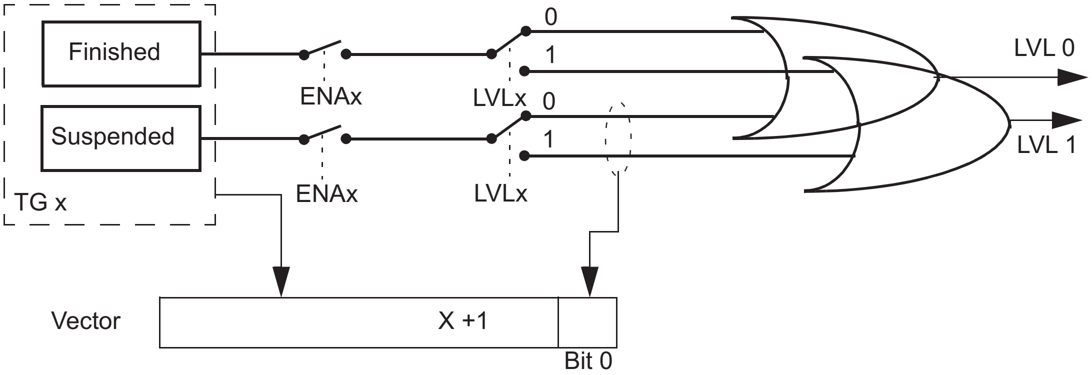
- Transmission error interrupt
- Receive overrun interrupt
- TG suspended interrupt
- TG completed interrupt

When a TG has finished and the corresponding enable bit in the TGINTENA register is set, a transfer-finished interrupt is generated. The level of priority of the interrupt is determined by the corresponding bit in the TGINTLVL register.

When a TG is suspended by a buffer that has been set as suspend to wait until TXFULL flag or/and RXEMPTY flag are set, and if the corresponding bit in the TGINTENA register is set, an transfer-suspended interrupt is generated. The level of priority of the interrupt is determined by the corresponding bit in the TGINTLVL register.

Figure 27-23 illustrates the TG interrupts.

Figure 27-23. TG Interrupt Structure

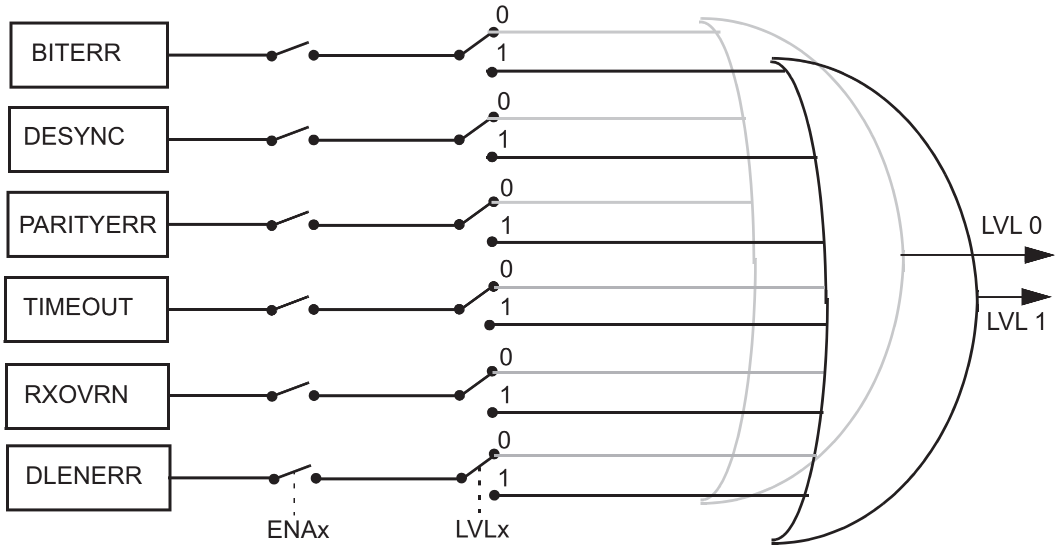


During transmission, if one of the following errors occurs, BITERR, DESYNC, PARITYERR, TIMEOUT, DLENERR, the corresponding flag in the SPIFLG register is set. If the enable bit is set, then an interrupt is generated. The level of the interrupts could be generated according to the bit field in SPILVL register.

The RXOVRN interrupt is generated when a buffer in the RXRAM is overwritten by a new received word. While writing newly received data to a RXRAM location, if the RXEMPTY bit of the corresponding location is 0, then the RXOVR bit will be set to 1 during the write operation, so that the buffer starts to indicate an overrun. This RXOVR flag is also reflected in SPIFLG register as RXOVRNINTFLG and the corresponding vector number is updated in TGINTVECT0/TGINTVECT1 register. If an overrun interrupt is enabled, then an interrupt will be generated indicating an overrun condition.

The error interrupts are enabled and prioritized independently from each other, but the vector generated by the SPI will be the same if multiple errors are enabled on the same level.

Figure 27-24. SPIFLG Interrupt Structure



Since the priority of an error interrupt is lower than a completion/suspend interrupt for a TG, the interrupts can be split into two levels. By programming all the error interrupts into Level 0 and TG-complete / TG-suspend interrupts into Level 1, it is possible to get a clear indication of the source of error interrupts. However, when a vector register shows an error interrupt, the actual buffer for which the error has occurred is not readily identifiable. Since each buffer in the multi-buffer RAM is stored along with its individual status flags, each buffer should be read until a buffer with any error flag set is found.

A separate interrupt line is provided to indicate the uncorrectable error condition in the MibSPI. This line is available (and valid) only in the multi-buffer mode of the MibSPI module and if the parity error detection feature for multi-buffer RAM is enabled.

27.7 DMA Interface

In order to reduce CPU overhead in handling SPI message traffic on a character-by-character basis, SPI can use the DMA controller to transfer the data. The DMA request enable bit (DMA REQ EN) controls the assertion of requests to the DMA controller module. When a character is being transmitted or received, the SPI will signal the DMA via the DMA request signals, TX_DMA_REQ and RX_DMA_REQ. The DMA controller will then perform the required data transfer.

For efficient behavior during DMA operations, the transmitter empty and receive-buffer full interrupts can be disabled. For specific DMA features, see the DMA controller specification.

The SPI generates a request on the TX_DMA_REQ line each time the TX data is copied to the TX shift register either from the TXBUF or from peripheral data bus (when TXBUF is empty).

The first TX_DMA_REQ pulse is generated when either of the following is true:

- DMAREQEN (SPIINT0[16]) is set to 1 while SPIEN (SPIGCR1[24]) is already 1.
- SPIEN (SPIGCR1[24]) is set to 1 while DMAREQEN (SPIINT0[16]) is already 1.

The SPI generates a request on the RX_DMA_REQ line each time the received data is copied to the SPIBUF.

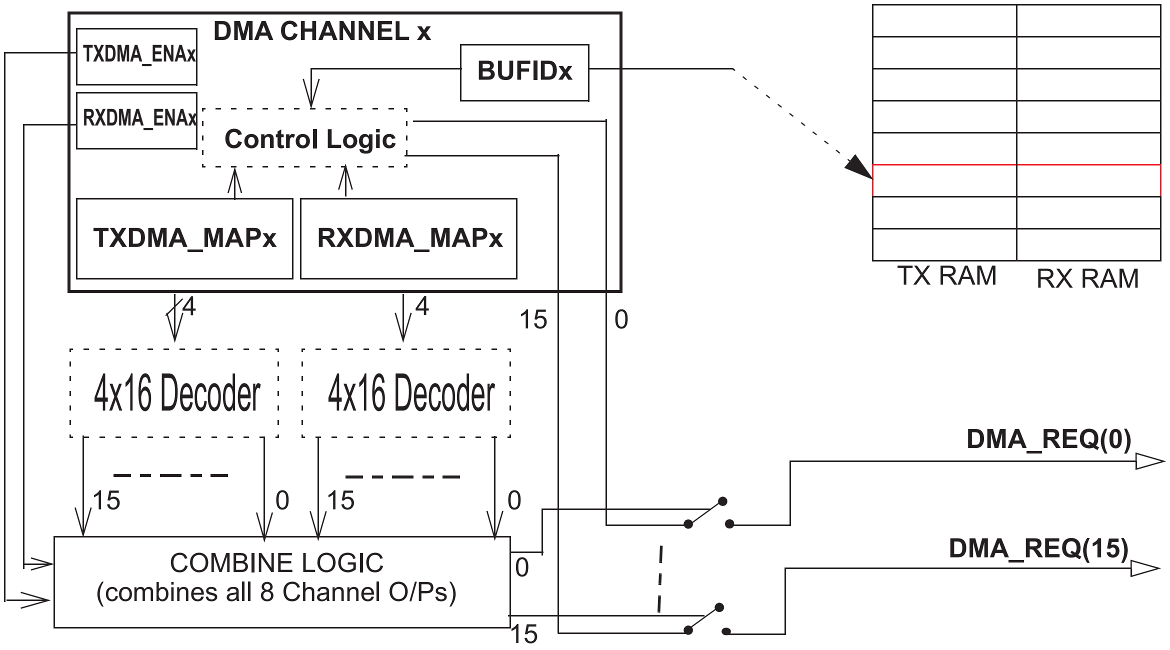
27.7.1 DMA in Multi-Buffer Mode

The MibSPI provides sophisticated programmable DMA control logic that completely eliminates the necessity of CPU intervention for data transfers, once programmed. When the multi-buffer mode is used, the DMA enable bit in the SPIINT0 register is ignored. DMA source or destination should be only the multi-buffer RAM and not SPIDAT0 / SPIDAT1 or SPIBUF register as in case of compatibility mode DMA.

The MibSPI offers up to eight DMA channels (for SEND and RECEIVE). All of the DMA channels are programmable individually and can be hooked to any buffer. The MibSPI provides up to 16 DMA request lines, and DMA requests from any channel can be programmed to be routed through any of these 16 lines. A DMA transfer can trigger both transmit and receive.

Each DMA channel has the capability to transfer a block of up to 32 data words without interruption using only one buffer of the array by configuring the DMAxCTRL register. Using the DMAxCOUNT and DMACTNTLEN register, up to 65535 (64K) words of data can be transferred without any interruption using just one buffer of the array. This enables the transfer of memory blocks from or into an external SPI memory.

Figure 27-25. DMA Channel and Request Line (Logical) Structure in Multi-buffer Mode



27.8 Module Configuration

MibSPI/MibSPIP can be configured to function as Normal SPI and Multi-buffered SPI. Upon power-up or a system-level reset, each bit in the module registers is set to a default state. The registers are writable only after the RESET bit is set to 1.

27.8.1 Compatibility (SPI) Mode Configuration

The following list details the configuration steps that software should perform prior to the transmission or reception of data. As long as the SPIEN bit in the Global Control Register 1 (SPIGCR1) is cleared to 0 the entire time that the SPI is being configured, the order in which the registers are programmed is not important.

- Enable SPI by setting RESET bit.
- Configure the SIMO, SOMI, SPICLK, and optional $\overline{\text{SPICS}}$ and $\overline{\text{SPIENA}}$ pins for SPI functionality by setting the corresponding bit in SPIPC0 register.
- Configure the module to function as Master or Slave using CLKMOD and MASTER bits.
- Configure the required SPI data format using SPIFMTx register.
- If the module is selected to function as Master, the delay parameters can be configured using SPIDELAY register.
- Enable the Interrupts using SPIINT0 register if required.
- Select the chip select to be used by setting CSNR bits in SPIDAT1 register.
- Configure CSHOLD and WDEL bits in SPIDAT1 register if required.
- Select the Data word format by setting DFSEL bits. Select the Number of the configured SPIFMTx register (0 to 3) to used for the communication.
- Set LOOPBACK bit to connect the transmitter to the receiver internally. (This feature is used to perform a self-test. Do not configure for normal communication to external devices).
- Set SPIEN bit to 1 after the SPI is configured.
- Perform Transmit and receive data, using SPIDAT1 and SPIBUF register.
- You must wait for TXFULL to reset or TXINT before writing next data to SPIDAT1 register.
- You must wait for RXEMPTY to reset or RXINT before reading the data from SPIBUF register.

27.8.2 MibSPI Mode Configuration

The following list details the configuration steps that software should perform prior to the transmission or reception of data in MIBSPI mode. As long as the SPIEN bit in the Global Control Register 1 (SPIGCR1) is cleared to 0 the entire time that the SPI is being configured, the order in which the registers are programmed is not important.

- Enable SPI by setting RESET bit.
- Set MSPIENA bit to 1 to get access to multi-buffer mode registers.
- Configure the SIMO, SOMI, SPICLK, and optional $\overline{\text{SPICS}}$ and $\overline{\text{SPIENA}}$ pins for SPI functionality by setting the corresponding bit in SPIPC0 register.
- Configure the module to function as Master or Slave using CLKMOD and MASTER bits.
- Configure the required SPI data format using SPIFMTx register.
- If the module is selected to function as Master, the delay parameters can be configured using SPIDELAY register.
- Check for BUFINITACTIVE bit to be active before configuring MIBSPI RAM. (From Device Power On it take Number of Buffers × Peripheral clock period to initialize complete RAM.)
- Enable the Transfer Group interrupts using TGITENST register if required.
- Enable error interrupts using SPIINT0 register if required.
- Set SPIEN bit to 1 after the SPI is configured.
- The Trigger Source, Trigger Event, Transfer Group start address for the corresponding Transfer groups can be configured using the corresponding TGxCTRL register.
- Configure LPEND to specify the end address of the last TG.
- Similar to SPIDAT1 register, the 16 bit control fields in every TXRAM buffer in the TG have to be configured.
- Configure one of the eight BUFMODE available for each buffer.
- Fill the data to be transmitted in TXDATA field in TXRAM buffers.
- Configure TGENA bit to enable the required Transfer groups. (In case of Trigger event always setting TGENA will trigger the transfer group).
- At the occurrence of the correct trigger event, the Transfer group will be triggered and data gets transmitted and received one after the other with out any CPU intervention.
- You can poll Transfer group interrupt flag or wait for a transfer-completed interrupt to read and write new data to the buffers.

27.9 Control Registers

This section describes the SPI control, data, and pin registers. The registers support 8-bit, 16-bit and 32-bit writes. The offset is relative to the associated base address of this module in a system. The base address for the control registers is FFF7 F400h for MibSPI1, FFF7 F600h for SPI2, FFF7 F800h for MibSPI3, FFF7 FA00h for SPI4, and FFF7 FC00h for MibSPI5.

NOTE: TI highly recommends that write values corresponding to the reserved locations of registers be maintained as 0 consistently. This allows future enhancements to use these reserved bits as control bits without affecting the functionality of the module with any older versions of software.

Table 27-7. SPI Registers

Offset	Acronym	Register Description	Section
00h	SPIGCR0	SPI Global Control Register 0	Section 27.9.1
04h	SPIGCR1	SPI Global Control Register 1	Section 27.9.2
08h	SPIINT0	SPI Interrupt Register	Section 27.9.3
0Ch	SPILVL	SPI Interrupt Level Register	Section 27.9.4
10h	SPIFLG	SPI Flag Register	Section 27.9.5
14h	SPIPC0	SPI Pin Control Register 0	Section 27.9.6
18h	SPIPC1	SPI Pin Control Register 1	Section 27.9.7
1Ch	SPIPC2	SPI Pin Control Register 2	Section 27.9.8
20h	SPIPC3	SPI Pin Control Register 3	Section 27.9.9
24h	SPIPC4	SPI Pin Control Register 4	Section 27.9.10
28h	SPIPC5	SPI Pin Control Register 5	Section 27.9.11
2Ch	SPIPC6	SPI Pin Control Register 6	Section 27.9.12
30h	SPIPC7	SPI Pin Control Register 7	Section 27.9.13
34h	SPIPC8	SPI Pin Control Register 8	Section 27.9.14
38h	SPIDAT0	SPI Transmit Data Register 0	Section 27.9.15
3Ch	SPIDAT1	SPI Transmit Data Register 1	Section 27.9.16
40h	SPIBUF	SPI Receive Buffer Register	Section 27.9.17
44h	SPIEMU	SPI Emulation Register	Section 27.9.18
48h	SPIDELAY	SPI Delay Register	Section 27.9.19
4Ch	SPIDEF	SPI Default Chip Select Register	Section 27.9.20
50h-5Ch	SPIFMT0-SPIFMT3	SPI Data Format Registers	Section 27.9.21
60h	INTVECT0	Interrupt Vector 0	Section 27.9.22
64h	INTVECT1	Interrupt Vector 1	Section 27.9.23
68h	SPIPC9[1]	SPI Pin Control Register 9	Section 27.9.24
6Ch	SPIPMCTRL	Parallel/Modulo Mode Control Register	Section 27.9.25
70h	MIBSPIE	Multi-buffer Mode Enable Register	Section 27.9.26
74h	TGITENST	TG Interrupt Enable Set Register	Section 27.9.27
78h	TGITENCR	TG Interrupt Enable Clear Register	Section 27.9.28
7Ch	TGITLVST	Transfer Group Interrupt Level Set Register	Section 27.9.29
80h	TGITLVCR	Transfer Group Interrupt Level Clear Register	Section 27.9.30
84h	TGINTFLG	Transfer Group Interrupt Flag Register	Section 27.9.31
88h-8Ch	Reserved	Reserved	
90h	TICKCNT	Tick Count Register	Section 27.9.32
94h	LTGPEND	Last TG End Pointer	Section 27.9.33
98h-D4h	TGxCTRL	TGx Control Registers	Section 27.9.34
D8h-F4h	DMAxCTRL	DMA Channel Control Register	Section 27.9.35

[1] SPIPC9 only applies to SPI2.

Table 27-7. SPI Registers (continued)

Offset	Acronym	Register Description	Section
F8h-114h	ICOUNT	DMAxCOUNT Register	Section 27.9.36
118h	DMACNTLEN	DMA Large Count	Section 27.9.37
11Ch	Reserved	Reserved	
120h	UERRCTRL	Multi-buffer RAM Uncorrectable Parity Error Control Register	Section 27.9.38
124h	UERRSTAT	Multi-buffer RAM Uncorrectable Parity Error Status Register	Section 27.9.39
128h	UERRADDR1	RXRAM Uncorrectable Parity Error Address Register	Section 27.9.40
12Ch	UERRADDR0	TXRAM Uncorrectable Parity Error Address Register	Section 27.9.41
130h	RXOVRN_BUF_ADDR	RXRAM Overrun Buffer Address Register	Section 27.9.42
134h	IOLPBKTSTCR	I/O Loopback Test Control Register	Section 27.9.43
138h	EXTENDED_PRESCALE1	SPI Extended Prescale Register 1	Section 27.9.44
13Ch	EXTENDED_PRESCALE2	SPI Extended Prescale Register 2	Section 27.9.45

27.9.1 SPI Global Control Register 0 (SPIGCR0)

Figure 27-26. SPI Global Control Register 0 (SPIGCR0) [offset = 00]

31-16
Reserved
R-0

15-1	0
Reserved	nRESET
R-0	R/WP-0

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

Table 27-8. SPI Global Control Register 0 (SPIGCR0) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	nRESET		This is the local reset control for the module. This bit needs to be set to 1 before any operation on SPI / MibSPI can be done. Only after setting this bit to 1, the Auto Initialization of Multi-buffer RAM starts. Clearing this bit to 0 will result in all of the control and status register values to return to their default values..
		0	SPI is in the reset state.
		1	SPI is out of the reset state.

27.9.2 SPI Global Control Register 1 (SPIGCR1)

Figure 27-27. SPI Global Control Register 1 (SPIGCR1) [offset = 04h]

31 25	24	23 17	16
Reserved	SPIEN	Reserved	LOOPBACK
R-0	R/W-0	R-0	R/WP-0

15 9	8	7 2	1	0
Reserved	POWERDOWN	Reserved	CLKMOD	MASTER
R-0	R/W-0	R-0	R/W-0	R/W-0

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

Table 27-9. SPI Global Control Register 1 (SPIGCR1) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	SPIEN		SPI enable. This bit enables SPI transfers. This bit must be set to 1 after all other SPI configuration bits have been written. When the SPIEN bit is 0 or cleared to 0, the following SPI registers get forced to their default states: • Both TX and RX shift registers • The TXDATA fields of the SPI Transmit Data Register 0 (SPIDAT0) and the SPI Transmit Data Register 1 (SPIDAT1) • All the fields of the SPI Flag Register (SPIFLG) • Contents of SPIBUF and the internal RXBUF registers
		0	The SPI is not activated for transfers.
		1	Activates SPI.
23-17	Reserved	0	Reads return 0. 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 SPISIMO and SPISOMI pins are configured with SPI functionality, then the SPISIMO[7:0] pins are internally connected to the SPISOMI[7:0] pins (transmit data is looped back as receive data). GIO mode for these pins is not supported in loopback mode. Externally, during loop-back operation, the SPICLK pin outputs an inactive value and SPISOMI[7:0] remains in the high-impedance state. If the SPI is initialized in slave mode or a data transfer is ongoing, errors may result. **Note: This loopback mode can only be used in master mode. Master mode must be selected before setting LOOPBACK. When this mode is selected, the CLKMOD bit should be set to 1, meaning that SPICLK is internally generated.**
		0	Internal loop-back test mode is disabled.
		1	Internal loop-back test mode is enabled.
15-9	Reserved	0	Reads return 0. 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 0. Writes have no effect.
1	CLKMOD		Clock mode. This bit selects either an internal or external clock source. This bit also determines the I/O direction of the $\overline{\text{SPIENA}}$ and $\overline{\text{SPICS}}$ pins in functional mode.
		0	Clock is external. • $\overline{\text{SPIENA}}$ is an output. • $\overline{\text{SPICS}}$ are inputs.
		1	Clock is internally-generated. • $\overline{\text{SPIENA}}$ is an input. • $\overline{\text{SPICS}}$ are outputs.

Table 27-9. SPI Global Control Register 1 (SPIGCR1) Field Descriptions (continued)

Bit	Field	Value	Description
0	MASTER		SPISIMO/SPISOMI pin direction determination. Sets the direction of the SPISIMO and SPISOMI pins. **Note: For master-mode operation of the SPI, MASTER bit should be set to 1 and CLKMOD bit can be set either 1 or 0. The master-mode SPI can run on an external clock on SPICLK.** **For slave mode operation, both the MASTER and CLKMOD bits should be cleared to 0. Any other combinations may result in unpredictable behavior of the SPI. In slave mode. SPICLK will not be generated internally in slave mode.**
		0	SPISIMO[7:0] pins are inputs, SPISOMI[7:0] pins are outputs.
		1	SPISOMI[7:0] pins are inputs, SPISIMO[7:0] pins are outputs.

27.9.3 SPI Interrupt Register (SPIINT0)

Figure 27-28. SPI Interrupt Register (SPIINT0) [offset = 08h]

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		TXINT ENA	RXINT ENA
R-0		R/W-0	R/W-0

7	6	5	4	3	2	1	0
Reserved	RXOVRNINT ENA	Reserved	BITERR ENA	DESYNC ENA	PARERR ENA	TIMEOUT ENA	DLENERR ENA
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 27-10. SPI Interrupt Register (SPIINT0) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	ENABLEHIGHZ		$\overline{\text{SPIENA}}$ pin high-impedance enable. When active, the $\overline{\text{SPIENA}}$ pin (when it is configured as a WAIT functional output signal in a slave SPI) is forced to high-impedance when not driving a low signal. If inactive, then the pin will output both a high and a low signal.
		0	$\overline{\text{SPIENA}}$ pin is pulled high when not active.
		1	$\overline{\text{SPIENA}}$ pin remains high-impedance when not active.
23-17	Reserved	0	Reads return 0. Writes have no effect.
16	DMAREQEN		DMA request enable. Enables the DMA request signal to be generated for both receive and transmit channels. Enable DMA REQ only after setting the SPIEN bit to 1.
		0	DMA is not used.
		1	DMA requests will be generated. **Note: A DMA request will be generated on the TX DMA REQ line each time a word is copied to the shift register either from TXBUF or directly from SPIDAT0/SPIDAT1 writes.** **Note: A DMA request will be generated on the RX DMA REQ line each time a word is copied to the SPIBUF register either from RXBUF or directly from the shift register.**
15-10	Reserved	0	Reads return 0. Writes have no effect.

Table 27-10. SPI Interrupt Register (SPIINT0) Field Descriptions (continued)

Bit	Field	Value	Description
9	TXINTENA		Causes an interrupt to be generated every time data is written to the shift register, so that the next word can be written to TXBUF. Setting this bit will generate an interrupt if the TXINTFLG bit (SPI Flag Register (SPIFLG)[9]) is set to 1.
		0	No interrupt will be generated upon TXINTFLG being set to 1.
		1	An interrupt will be generated upon TXINTFLG being set to 1.
			The transmitter empty interrupt is valid in compatibility mode of SPI only. In multi-buffered mode, this interrupts will not be generated, even if it is enabled.
			Note: An interrupt request will be generated as soon as this bit is set to 1. By default it will be generated on the INT0 line. The SPILVL register can be programmed to change the interrupt line.
8	RXINTENA		Causes an interrupt to be generated when the RXINTFLAG bit (SPI Flag Register (SPIFLG)[8]) is set by hardware.
		0	Interrupt will not be generated.
		1	Interrupt will be generated.
			The receiver full interrupt is valid in compatibility mode of SPI only. In multi-buffered mode, this interrupts will not be generated, even if it is enabled.
7	Reserved	0	Reads return 0. Writes have no effect.
6	RXOVRNINTENA		Overrun interrupt enable.
		0	Overrun interrupt will not be generated.
		1	Overrun interrupt will be generated.
5	Reserved	0	Reads return 0. Writes have no effect.
4	BITERRENA		Enables interrupt on bit error.
		0	No interrupt asserted upon bit error.
		1	Enables interrupt on bit error.
3	DESYNCENA		Enables interrupt on desynchronized slave. DESYNCENA is used in master mode only.
		0	No interrupt asserted upon desynchronization error.
		1	An interrupt is asserted on desynchronization of the slave (DESYNC = 1).
2	PARERRENA		Enables interrupt-on-parity-error.
		0	No interrupt asserted on parity error.
		1	An interrupt is asserted on a parity error.
1	TIMEOUTENA		Enables interrupt on ENA signal time-out.
		0	No interrupt asserted upon ENA signal time-out.
		1	An interrupt is asserted on a time-out of the ENA signal.
0	DLENERRENA		Data length error interrupt enable. A data length error occurs under the following conditions.
			Master: When $\overline{\text{SPIENA}}$ is used, if the $\overline{\text{SPIENA}}$ 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 $\overline{\text{SPIENA}}$ deassertion is detected, then it means that the slave has neither received full data from the master nor has it transmitted complete data.
			Slave: When $\overline{\text{SPICS}}$ pins are used, if the incoming valid $\overline{\text{SPICS}}$ pin is deactivated before the character length counter overflows, then the data length error is set.
		0	No interrupt is generated upon data length error.
		1	An interrupt is asserted when a data-length error occurs.

27.9.4 SPI Interrupt Level Register (SPILVL)

Figure 27-29. SPI Interrupt Level Register (SPILVL) [offset = 0Ch]

31–16
Reserved
R-0

15–10	9	8
Reserved	TXINT LVL	RXINT LVL
R-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
Reserved	RXOVRNINT LVL	Reserved	BITERR LVL	DESYNC LVL	PARERR LVL	TIMEOUT LVL	DLENERR LVL
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 Level Register (SPILVL) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return 0. Writes have no effect.
9	TXINTLVL		Transmit interrupt level.
		0	Transmit interrupt is mapped to interrupt line INT0.
		1	Transmit interrupt is mapped to interrupt line INT1.
8	RXINTLVL		Receive interrupt level.
		0	Receive interrupt is mapped to interrupt line INT0.
		1	Receive interrupt is mapped to interrupt line INT1.
7	Reserved	0	Reads return 0. Writes have no effect.
6	RXOVRNINTLVL		Receive overrun interrupt level.
		0	Receive overrun interrupt is mapped to interrupt line INT0.
		1	Receive overrun interrupt is mapped to interrupt line INT1.
5	Reserved	0	Reads return 0. Writes have no effect.
4	BITERRLVL		Bit error interrupt level.
		0	Bit error interrupt is mapped to interrupt line INT0.
		1	Bit error interrupt is mapped to interrupt line INT1.
3	DESYNCLVL		Desynchronized slave interrupt level. (master mode only).
		0	An interrupt caused by desynchronization of the slave is mapped to interrupt line INT0.
		1	An interrupt caused by desynchronization of the slave is mapped to interrupt line INT1.
2	PARERRLVL		Parity error interrupt level.
		0	A parity error interrupt is mapped to interrupt line INT0.
		1	A parity error interrupt is mapped to interrupt line INT1.
1	TIMEOUTLVL		$\overline{\text{SPIENA}}$ pin time-out interrupt level.
		0	An interrupt on a time-out of the ENA signal (TIMEOUT = 1) is mapped to interrupt line INT0.
		1	An interrupt on a time-out of the ENA signal (TIMEOUT = 1) is mapped to interrupt line INT1.
0	DLENERRLVL		Data length error interrupt level (line) select.
		0	An interrupt on data length error is mapped to interrupt line INT0.
		1	An interrupt on data length error is mapped to interrupt line INT1.

27.9.5 SPI Flag Register (SPIFLG)

Software must check all flag bits when reading this register.

Figure 27-30. SPI Flag Register (SPIFLG) [offset = 10h]

31 25	24	23 16
Reserved	BUFINIT ACTIVE	Reserved
R-0	R-0	R-0

15 10	9	8
Reserved	TXINT FLG	RXINT FLG
R-0	R-0	R/W1C-0

7	6	5	4	3	2	1	0
Reserved	RXOVRNINT FLG	Reserved	BITERR FLG	DESYNC FLG	PARERR FLG	TIMEOUT FLG	DLENERR FLG
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; -*n* = value after reset

Table 27-12. SPI Flag Register (SPIFLG) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	BUFINITACTIVE		Indicates the status of multi-buffer initialization process. Software can poll for this bit to determine if it can proceed with the register configuration of multi-buffer mode registers or buffer handling. **Note: If the SPIFLG register is read while the multi-buffer RAM is being initialized, the BUFINITACTIVE bit will be read as 1. If SPIFLG is read after the internal automatic buffer initialization is complete, this bit will be read as 0. This bit will show a value of 1 as long as the nRESET bit is 0, but does not really indicate that buffer initialization is underway. Buffer initialization starts only when the nRESET bit is set to 1.**
		0	Multi-buffer RAM initialization is complete.
		1	Multi-buffer RAM is still being initialized. Do not attempt to write to either multi-buffer RAM or any multi-buffer mode registers.
23-10	Reserved	0	Reads return 0. 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 word can be written to it. This flag is set when a word is copied to the shift register either directly from SPIDAT0/SPIDAT1 or from the TXBUF register. This bit is cleared by one of following methods: • Writing a new data to either SPIDAT0 or SPIDAT1 • Writing a 0 to SPIEN (SPIGCR1[24])
		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.

Table 27-12. SPI Flag Register (SPIFLG) Field Descriptions (continued)

Bit	Field	Value	Description
8	RXINTFLG		Receiver-full interrupt flag. This flag is set when a word is received and copied into the buffer register (SPIBUF). If RXINTEN is enabled, an interrupt is also generated. This bit is cleared under the following methods: • Reading the SPIBUF register • Reading TGINTVECT0 or TGINTVECT1 register when there is a receive buffer full interrupt • Writing a 1 to this bit • Writing a 0 to SPIEN (SPIGCR1[24]) • System reset During emulation mode, however, a read to the emulation register (SPIEMU) does not clear this flag bit.
		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. In this way, one can ignore a received word. 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 0. Writes have no effect.
6	RXOVRNINTFLG		Receiver overrun flag. The SPI hardware sets this bit 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. The SPI will generate an interrupt request if this bit is set and the RXOVRN INTEN bit (SPIINT0.6) is set high. This bit is cleared under the following conditions in compatibility mode of MibSPI: • Reading TGINTVECT0 or TGINTVECT1 register when there is a receive-buffer-overrun interrupt • Writing a 1 to RXOVRNINTFLG in the SPI Flag Register (SPIFLG) itself • Writing a 0 to SPIEN • Reading the data field of the SPIBUF register **Note: Reading the SPIBUF register does not clear this RXOVRNINTFLG bit. If an RXOVRN 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: There is a special condition under which the RXOVRNINTFLG flag gets set. If both SPIBUF and RXBUF are already full and while another reception is underway, if any errors (TIMEOUT, BITERR, and DLEN_ERR) occur, then RXOVR in RXBUF and RXOVRNINTFLG in SPIFLG registers will be set to indicate that the status flags are getting overwritten by the new transfer. This overrun should be treated like a receive overrun.** In multi-buffer mode of MibSPI, this bit is cleared under the following conditions: • Reading the RXOVRN_BUF_ADDR register • Writing a 1 to RXOVRNINTFLG in the SPI Flag Register (SPIFLG) itself In multi-buffer mode, if RXOVRNINTFLG is set, then the address of the buffer which experienced the overrun is available in RXOVRN_BUF_ADDR.
		0	Overrun condition did not occur.
		1	Overrun condition has occurred.
5	Reserved	0	Reads return 0. Writes have no effect.
4	BITERRFLG		Mismatch of internal transmit data and transmitted data. This flag can be cleared by one of the following methods. • Write a 1 to this bit • Clear the SPIEN bit to 0
		0	No bit error occurred.
		1	A bit error occurred. 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 BITERRFLG is set. If BITERRENA is set an interrupt is asserted. Possible reasons for a bit error can be an excessively high bit rate, capacitive load, or another master/slave trying to transmit at the same time.

Table 27-12. SPI Flag Register (SPIFLG) Field Descriptions (continued)

Bit	Field	Value	Description
3	DESYNCFLG		Desynchronization of slave device. Desynchronization monitor is active in master mode only. This flag can be cleared by one of the following methods: • Write a 1 to this bit • Clear the SPIEN bit to 0
		0	No slave desynchronization is detected.
		1	A slave device is desynchronized. The master monitors the ENAble signal coming from the slave device and sets the DESYNC flag 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.
2	PARERRFLG		Calculated parity differs from received parity bit. If the parity generator is enabled (can be selected individually for each buffer) 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 PARITYERR flag is set and an interrupt is asserted if PARERRENA is set. This flag can be cleared by one of the following methods: • Write a 1 to this bit • Clear the SPIEN bit to 0
		0	No parity error is detected.
		1	A parity error occurred.
1	TIMEOUTFLG		Time-out caused by nonactivation of ENA signal. This flag can be cleared by one of the following methods: • Write a 1 to this bit • Clear the SPIEN bit to 0
		0	No ENA-signal time-out occurred.
		1	An ENA signal time-out occurred. The SPI generates a time-out because the slave hasn't responded in time by activating the 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. In addition the TIMEOUT flag in the status field of the corresponding buffer is set. The transmit request of the concerned buffer is cleared, that is, the SPI does not re-start a data transfer from this buffer.
0	DLENERRFLG		Data-length error flag. This flag can be cleared by one of the following methods: • Write a 1 to this bit • Clear the SPIEN bit to 0 **Note: Whenever any transmission errors (TIMEOUT, BITERR, DLEN_ERR, PARITY_ERR, DESYNC) are detected and the error flags are cleared by writing to the error bit in the SPIFLG register, the corresponding error flag in SPIBUF does not get cleared. Software needs to read the SPIBUF until it becomes empty before proceeding. This ensures that all of the old status bits in SPIBUF are cleared before starting the next transfer.**
		0	No data length error has occurred.
		1	A data length error has occurred.

27.9.6 SPI Pin Control Register 0 (SPIPC0)

NOTE: Register bits vary by device

Register bits 31:24 and 23:16 of SPIPC0 to SPIPC9 reflect the number of SIMO/SOMI data lines per device. On devices with 8 data-line support, all of bits 31 to 16 are implemented. On devices with less than 8 data lines, only a subset of these bits are available. Unimplemented bits return 0 upon read and are not writable.

Figure 27-31. SPI Pin Control Register 0 (SPIPC0) [offset = 14h]

31 24	23 16
SOMIFUN	SIMOFUN
R/W-0	R/W-0

15 12	11	10	9	8
Reserved	SOMIFUN0	SIMOFUN0	CLKFUN	ENAFUN
R-0	R/W-0	R/W-0	R/W-0	R/W-0

7 0
SCSFUN
R/W-0

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

Table 27-13. SPI Pin Control (SPIPC0) Field Descriptions

Bit	Field	Value	Description
31-24	SOMIFUN		Slave out, master in function. Determines whether each SPISOMI[x] pin is to be used as a general-purpose I/O pin or as a SPI functional pin.
			Note: Duplicate Control Bits for SPISOMI[0]. Bit 24 is not physically implemented. It is a mirror of Bit 11. Any write to bit 24 will be reflected on bit 11. When bit 24 and bit 11 are simultaneously written, the value of bit 11 will control the SPISOMI[0] pin. The read value of bit 24 always reflects the value of bit 11.
		0	The SPISOMI[x] pin is a GIO pin.
		1	The SPISOMI[x] pin is a SPI functional pin.
23-16	SIMOFUN		Slave in, master out function. Determines whether each SPISIMO[x] pin is to be used as a general-purpose I/O pin or as a SPI functional pin.
			Note: Duplicate Control Bits for SPISIMO[0]. Bit 16 is not physically implemented. It is a mirror of Bit 10. Any write to bit 16 will be reflected on bit 10. When bit 16 and bit 10 are simultaneously written, the value of bit 10 will control the SPISIMO[0] pin. The read value of bit 16 always reflects the value of bit 10.
		0	The SPISIMO[x] pin is a GIO pin.
		1	The SPISIMO[x] pin is a SPI functional pin.
15-12	Reserved	0	Reads return 0. Writes have no effect.
11	SOMIFUN0		Slave out, master in function. This bit determines whether the SPISOMI[0] pin is to be used as a general-purpose I/O pin or as a SPI functional pin.
		0	The SPISOMI[0] pin is a GIO pin.
		1	The SPISOMI[0] pin is a SPI functional pin.
			Note: Regardless of the number of parallel pins used, the SPISOMI[0] pin will always have to be programmed as functional pins for any SPI transfers.
10	SIMOFUN0		Slave in, master out function. This bits determine whether each SPISIMO[0] pin is to be used as a general-purpose I/O pin or as a SPI functional pin.
		0	The SPISIMO[0] pin is a GIO pin.
		1	The SPISIMO[0] pin is a SPI functional pin.
			Note: Regardless of the number of parallel pins used, the SPISIMO[0] pin will always have to be programmed as functional pins for any SPI transfers.

Table 27-13. SPI Pin Control (SPIPC0) Field Descriptions (continued)

Bit	Field	Value	Description
9	CLKFUN		SPI clock function. This bit determines whether the SPICLK pin is to be used as a general-purpose I/O pin, or as a SPI functional pin.
		0	The SPICLK pin is a GIO pin.
		1	The SPICLK pin is a SPI functional pin.
8	ENAFUN		$\overline{\text{SPIENA}}$ function. This bit determines whether the $\overline{\text{SPIENA}}$ pin is to be used as a general-purpose I/O pin or as a SPI functional pin.
		0	The $\overline{\text{SPIENA}}$ pin is a GIO pin.
		1	The $\overline{\text{SPIENA}}$ pin is a SPI functional pin.
7-0	SCSFUN		$\overline{\text{SPICS}}$ function. Determines whether each $\overline{\text{SPICS}}$ pin is to be used as a general-purpose I/O pin or as a SPI functional pin. If the slave $\overline{\text{SPICS}}$ pins are in functional mode and receive an inactive high signal, the slave SPI will place its output in a high-impedance state and disable shifting.
		0	The $\overline{\text{SPICS}}$ pin is a GIO pin.
		1	The $\overline{\text{SPICS}}$ pin is a SPI functional pin.

27.9.7 SPI Pin Control Register 1 (SPIPC1)

NOTE: Register bits vary by device

Register bits 31:24 and 23:16 of this register reflect the number of SIMO/SOMI data lines per device. On devices with 8 data-line support, all of bits 31 to 16 are implemented. On devices with less than 8 data lines, only a subset of these bits are available. Unimplemented bits return 0 upon read and are not writable.

Figure 27-32. SPI Pin Control Register 1 (SPIPC1) [offset = 18h]

31-24	23-16
SOMIDIR	SIMODIR
R/W-0	R/W-0

15-12	11	10	9	8
Reserved	SOMIDIR0	SIMODIR0	CLKDIR	ENADIR
R-0	R/W-0	R/W-0	R/W-0	R/W-0

7-0
SCSDIR
R/W-0

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

Table 27-14. SPI Pin Control Register (SPIPC1) Field Descriptions

Bit	Field	Value	Description
31-24	SOMIDIR		SPISOMI[x] direction. Controls the direction of each SPISOMI[x] pin when used for general-purpose I/O. If SPISOMI[x] pin is used as a SPI functional pin, the I/O direction is determined by the MASTER bit in the SPIGCR1 register. **Note: Duplicate Control Bits for SPISOMI[0]. Bit 24 is not physically implemented. It is a mirror of Bit 11. Any write to bit 24 will be reflected on bit 11. When bit 24 and bit 11 are simultaneously written, the value of bit 11 will control the SPISOMI[0] pin. The read value of bit 24 always reflects the value of bit 11.**
		0	The SPISOMI[x] pin is an input.
		1	The SPISOMI[x] pin is an output.

Table 27-14. SPI Pin Control Register (SPIPC1) Field Descriptions (continued)

Bit	Field	Value	Description
23-16	SIMODIR		SPISIMO[x] direction. Controls the direction of each SPISIMO[x] pin when used for general-purpose I/O. If SPISIMO[x] pin is used as a SPI functional pin, the I/O direction is determined by the MASTER bit in the SPIGCR1 register.
			Note: Duplicate Control Bits for SPISIMO[0]. Bit 16 is not physically implemented. It is a mirror of Bit 10. Any write to bit 16 will be reflected on bit 10. When bit 16 and bit 10 are simultaneously written, the value of bit 10 will control the SPISIMO[0] pin. The read value of bit 16 always reflects the value of bit 10.
		0	The SPISIMO[x] pin is an input.
		1	The SPISIMO[x] pin is an output.
15-12	Reserved	0	Reads return 0. Writes have no effect.
11	SOMIDIR0		SPISOMI[0] direction. This bit controls the direction of the SPISOMI[0] pin when it is used as a general-purpose I/O pin. If the SPISOMI[0] pin is used as a SPI functional pin, the I/O direction is determined by the MASTER bit in the SPIGCR1 register.
		0	The SPISOMI[0] pin is an input.
		1	The SPISOMI[0] pin is an output.
10	SIMODIR0		SPISIMO[0] direction. This bit controls the direction of the SPISIMO[0] pin when it is used as a general-purpose I/O pin. If the SPISIMO[0] pin is used as a SPI functional pin, the I/O direction is determined by the MASTER bit in the SPIGCR1 register.
		0	The SPISIMO[0] pin is an input.
		1	The SPISIMO[0] pin is an output.
9	CLKDIR		SPICLK direction. This bit controls the direction of the SPICLK pin when it is used as a general-purpose I/O pin. In functional mode, the I/O direction is determined by the CLKMOD bit.
		0	The SPICLK pin is an input.
		1	The SPICLK pin is an output.
8	ENADIR		$\overline{\text{SPIENA}}$ direction. This bit controls the direction of the $\overline{\text{SPIENA}}$ pin when it is used as a general-purpose I/O. If the $\overline{\text{SPIENA}}$ pin is used as a functional pin, then the I/O direction is determined by the CLKMOD bit (SPIGCR1[1]).
		0	The $\overline{\text{SPIENA}}$ pin is an input.
		1	The $\overline{\text{SPIENA}}$ pin is an output.
7-0	SCSDIR		$\overline{\text{SPICS}}$ direction. These bits control the direction of each $\overline{\text{SPICS}}$ pin when it is used as a general-purpose I/O pin. Each pin could be configured independently from the others if the $\overline{\text{SPICS}}$ is used as a SPI functional pin. The I/O direction is determined by the CLKMOD bit (SPIGCR1[1]).
		0	The $\overline{\text{SPICS}}$ pin is an input.
		1	The $\overline{\text{SPICS}}$ pin is an output.

27.9.8 SPI Pin Control Register 2 (SPIPC2)

NOTE: Register bits vary by device

Register bits 31:24 and 23:16 of this register reflect the number of SIMO/SOMI data lines per device. On devices with 8 data-line support, all of bits 31 to 16 are implemented. On devices with less than 8 data lines, only a subset of these bits are available. Unimplemented bits return 0 upon read and are not writable.

Figure 27-33. SPI Pin Control Register 2 (SPIPC2) [offset = 1Ch]

31 24	23 16
SOMIDIN	SIMODIN
R/W-U	R/W-U

15 12	11	10	9	8
Reserved	SOMIDIN0	SIMODIN0	CLKDIN	ENADIN
R-0	R-U	R-U	R-U	R-U

7 0
SCSDIN
R/W-U

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

Table 27-15. SPI Pin Control Register 2 (SPIPC2) Field Descriptions

Bit	Field	Value	Description
31-24	SOMIDIN		SPISOMI[x] data in. The value of each SPISOMI[x] pin.
		0	The SPISOMI[x] pin is logic 0.
		1	The SPISOMI[x] pin is logic 1.
23-16	SIMODIN		SPISIMO[x] data in. The value of each SPISIMO[x] pin.
		0	The SPISIMO[x] pin is logic 0.
		1	The SPISIMO[x] pin is logic 1.
15-12	Reserved	0	Reads return 0. Writes have no effect.
11	SOMIDIN0		SPISOMI[0] data in. The value of the SPISOMI[0] pin.
		0	The SPISOMI[0] pin is logic 0.
		1	The SPISOMI[0] pin is logic 1.
10	SIMODIN0		SPISIMO[0] data in. The value of the SPISIMO[0] pin.
		0	The SPISIMO[0] pin is logic 0.
		1	The SPISIMO[0] pin is logic 1.
9	CLKDIN		Clock data in. The value of the SPICLK pin.
		0	The SPICLK pin is logic 0.
		1	The SPICLK pin is logic 1.
8	ENADIN		$\overline{\text{SPIENA}}$ data in. The the value of the $\overline{\text{SPIENA}}$ pin.
		0	The $\overline{\text{SPIENA}}$ pin is logic 0.
		1	The $\overline{\text{SPIENA}}$ pin is logic 1.
7-0	SCSDIN		$\overline{\text{SPICS}}$ data in. The value of each $\overline{\text{SPICS}}$ pin.
		0	The $\overline{\text{SPICS}}$ pin is logic 0.
		1	The $\overline{\text{SPICS}}$ pin is logic 1.

27.9.9 SPI Pin Control Register 3 (SPIPC3)

NOTE: Register bits vary by device

Register bits 31:24 and 23:16 of this register reflect the number of SIMO/SOMI data lines per device. On devices with 8 data-line support, all of bits 31 to 16 are implemented. On devices with less than 8 data lines, only a subset of these bits are available. Unimplemented bits return 0 upon read and are not writable.

Figure 27-34. SPI Pin Control Register 3 (SPIPC3) [offset = 20h]

31–24	23–16
SOMIDOUT	SIMODOUT
R/W-U	R/W-U

15–12	11	10	9	8
Reserved	SOMIDOUT0	SIMODOUT0	CLKDOUT	ENADOUT
R-0	R/W-U	R/W-U	R/W-U	R/W-U

7–0
SCSDOUT
R/W-U

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

Table 27-16. SPI Pin Control Register 3 (SPIPC3) Field Descriptions

Bit	Field	Value	Description
31-24	SOMIDOUT		SPISOMI[x] data out write. This bit is only active when the SPISOMI[x] 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. **Bit 11 or bit 24 can be used to set the direction for pin SPISOMI[0]. If a 32-bit write is performed, bit 11 will have priority over bit 24.**
		0	Current value on SPISOMI[x] pin is logic 0.
		1	Current value on SPISOMI[x] pin is logic 1
23-16	SIMODOUT		SPISIMO[x] data out write. This bit is only active when the SPISIMO[x] 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. **Bit 10 or bit 16 can be used to set the direction for pin SPISIMO[0]. If a 32-bit write is performed, bit 10 will have priority over bit 16.**
		0	Current value on SPISIMO[x] pin is logic 0.
		1	Current value on SPISIMO[x] pin is logic 1.
15-12	Reserved	0	Reads return 0. Writes have no effect.
11	SOMIDOUT0		SPISOMI[0] data out write. This bit is only active when the SPISOMI[0] 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 on SPISOMI[0] pin is logic 0.
		1	Current value on SPISOMI[0] pin is logic 1.
10	SIMODOUT0		SPISIMO[0] data out write. This bit is only active when the SPISIMO[0] 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 on SPISIMO[0] pin is logic 0.
		1	Current value on SPISIMO[0] pin is logic 1.
9	CLKDOUT		SPICLK data out write. This bit is only active when the SPICLK 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	The SPICLK pin is logic 0.
		1	The SPICLK pin is logic 1.

Table 27-16. SPI Pin Control Register 3 (SPIPC3) Field Descriptions (continued)

Bit	Field	Value	Description
8	ENADOUT		$\overline{\text{SPIENA}}$ data out write. Only active when the $\overline{\text{SPIENA}}$ 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	The $\overline{\text{SPIENA}}$ pin is logic 0.
		1	The $\overline{\text{SPIENA}}$ pin is logic 1.
7-0	SCSDOUT		$\overline{\text{SPICS}}$ data out write. Only active when the $\overline{\text{SPICS}}$ pins are configured as a general-purpose I/O pins and configured as output pins. The value of these bits indicates the value sent to the pins.
		0	The $\overline{\text{SPICS}}$ pin is logic 0.
		1	The $\overline{\text{SPICS}}$ pin is logic 1.

27.9.10 SPI Pin Control Register 4 (SPIPC4)

> **NOTE: Register bits vary by device**
>
> Register bits 31:24 and 23:16 of this register reflect the number of SIMO/SOMI data lines per device. On devices with 8 data-line support, all of bits 31 to 16 are implemented. On devices with less than 8 data lines, only a subset of these bits are available. Unimplemented bits return 0 upon read and are not writable.

Figure 27-35. SPI Pin Control Register 4 (SPIPC4) [offset = 24h]

31-24	23-16
SOMISET	SIMOSET
R/W-U	R/W-U

15-12	11	10	9	8
Reserved	SOMISET0	SIMOSET0	CLKSET	ENASET
R-0	R/W-U	R/W-U	R/W-U	R/W-U

7-0
SCSSET
R/W-U

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

Table 27-17. SPI Pin Control Register 4 (SPIPC4) Field Descriptions

Bit	Field	Value	Description
31-24	SOMISET		SPISOMI[x] data out set. This pin is only active when the SPISOMI[x] pin is configured as a general-purpose output pin. **Bit 11 or bit 24 can be used to set the SPISOMI[0] pin. If a 32-bit write is performed, bit 11 will have priority over bit 24.**
		0	Read: SPISOMI[x] is logic 0. Write: Writing a 0 to this bit has no effect.
		1	Read: SPISOMI[x] is logic 1. Write: Logic 1 is placed on SPISOMI[x] pin, if it is in general-purpose output mode.
23-16	SIMOSET		SPISIMO[x] data out set. This bit is only active when the SPISIMO[x] pin is configured as a general-purpose output pin. **Bit 10 or bit 16 can be used to set the SPISIMO[0] pin. If a 32-bit write is performed, bit 10 will have priority over bit 16.**
		0	Read: SPISIMO[x] is logic 0. Write: Writing a 0 to this bit has no effect.
		1	Read: SPISIMO[x] is logic 1. Write: Logic 1 is placed on SPISIMO[x] pin, if it is in general-purpose output mode.

Table 27-17. SPI Pin Control Register 4 (SPIPC4) Field Descriptions (continued)

Bit	Field	Value	Description
15-12	Reserved	0	Reads return 0. Writes have no effect.
11	SOMISET0	0	SPISOMI[0] data out set. This pin is only active when the SPISOMI[0] pin is configured as a general-purpose output pin. Read: SPISOMI[0] is logic 0. Write: Writing a 0 to this bit has no effect.
		1	Read: SPISOMI[0] is logic 1. Write: Logic 1 is placed on SPISOMI[0] pin, if it is in general-purpose output mode.
10	SIMOSET0	0	SPISIMO[0] data out set. This pin is only active when the SPISIMO[0] pin is configured as a general-purpose output pin. Read: SPISIMO[0] is logic 0. Write: Writing a 0 to this bit has no effect.
		1	Read: SPISIMO[0] is logic 1. Write: Logic 1 is placed on SPISIMO[0] pin, if it is in general-purpose output mode.
9	CLKSET	0	SPICLK data out set. This bit is only active when the SPICLK pin is configured as a general-purpose output pin. Read: SPICLK is logic 0. Write: Writing a 0 to this bit has no effect.
		1	Read: SPICLK is logic 1. Write: Logic 1 is placed on SPICLK pin, if it is in general-purpose output mode.
8	ENASET	0	$\overline{\text{SPIENA}}$ data out set. This bit is only active when the $\overline{\text{SPIENA}}$ pin is configured as a general-purpose output pin. Read: $\overline{\text{SPIENA}}$ is logic 0. Write: Writing a 0 to this bit has no effect.
		1	Read: $\overline{\text{SPIENA}}$ is logic 1. Write: Logic 1 is placed on $\overline{\text{SPIENA}}$ pin, if it is in general-purpose output mode.
7-0	SCSSET	0	$\overline{\text{SPICS}}$ data out set. This bit is only active when the $\overline{\text{SPICS}}$ pin is configured as a general-purpose output pin. A value of 1 written to this bit sets the corresponding SCSDOUT bit to 1. Read: $\overline{\text{SPICS}}$ is logic 0. Write: Writing a 0 to this bit has no effect.
		1	Read: $\overline{\text{SPICS}}$ is logic 1. Write: Logic 1 is placed on $\overline{\text{SPICS}}$ pin, if it is in general-purpose output mode.

27.9.11 SPI Pin Control Register 5 (SPIPC5)

NOTE: Register bits vary by device

Register bits 31:24 and 23:16 of this register reflect the number of SIMO/SOMI data lines per device. On devices with 8 data-line support, all of bits 31 to 16 are implemented. On devices with less than 8 data lines, only a subset of these bits are available. Unimplemented bits return 0 upon read and are not writable.

Figure 27-36. SPI Pin Control Register 5 (SPIPC5) [offset = 28h]

31 24	23 16
SOMICLR	SIMOCLR
R/W-U	R/W-U

15 12	11	10	9	8
Reserved	SOMICLR0	SIMOCLR0	CLKCLR	ENACLR
R-0	R/W-U	R/W-U	R/W-U	R/W-U

7 0
SCSCLR
R/W-U

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

Table 27-18. SPI Pin Control Register 5 (SPIPC5) Field Descriptions

Bit	Field	Value	Description
31-24	SOMICLR		SPISOMI[x] data out clear. This pin is only active when the SPISOMI[x] pin is configured as a general-purpose output pin. **Bit 11 or bit 24 can be used to set the SPISOMI[0] pin. If a 32-bit write is performed, bit 11 will have priority over bit 24.**
		0	Read: The current value on SPISOMI[x] is 0. Write: Writing a 0 to this bit has no effect.
		1	Read: The current value on SPISOMI[x] is 1. Write: Logic 0 is placed on SPISOMI[x] pin, if it is in general-purpose output mode.
23-16	SIMOCLR		SPISIMO[x] data out clear. This bit is only active when the SPISIMO[x] pin is configured as a general-purpose output pin. **Bit 10 or bit 16 can be used to set the SPISIMO[0] pin. If a 32-bit write is performed, bit 10 will have priority over bit 16.**
		0	Read: The current value on SPISIMO[x] is 0. Write: Writing a 0 to this bit has no effect.
		1	Read: The current value on SPISIMO[x] is 1. Write: Logic 0 is placed on SPISIMO[x] pin, if it is in general-purpose output mode.
15-12	Reserved	0	Reads return 0. Writes have no effect.
11	SOMICLR0		SPISOMI[0] data out clear. This pin is only active when the SPISOMI[0] pin is configured as a general-purpose output pin.
		0	Read: The current value on SPISOMI[0] is 0. Write: Writing a 0 to this bit has no effect.
		1	Read: The current value on SPISOMI[0] is 1. Write: Logic 0 is placed on SPISOMI[0] pin, if it is in general-purpose output mode.
10	SIMOCLR0		SPISIMO[0] data out clear. This pin is only active when the SPISIMO[0] pin is configured as a general-purpose output pin.
		0	Read: The current value on SPISIMO[0] is 0. Write: Writing a 0 to this bit has no effect.
		1	Read: The current value on SPISIMO[0] is 1. Write: Logic 0 is placed on SPISIMO[0] pin, if it is in general-purpose output mode.

Table 27-18. SPI Pin Control Register 5 (SPIPC5) Field Descriptions (continued)

Bit	Field	Value	Description
9	CLKCLR		SPICLK data out clear. This bit is only active when the SPICLK pin is configured as a general-purpose output pin.
		0	Read: The current value on SPICLK is 0. Write: Writing a 0 to this bit has no effect.
		1	Read: The current value on SPICLK is 1. Write: Logic 0 is placed on SPICLK pin, if it is in general-purpose output mode.
8	ENACLR		$\overline{\text{SPIENA}}$ data out clear. This bit is only active when the $\overline{\text{SPIENA}}$ pin is configured as a general-purpose output pin. A value of 1 written to this bit clears the corresponding ENABLEDOUT bit to 0.
		0	Read: The current value on $\overline{\text{SPIENA}}$ is 0. Write: Writing a 0 to this bit has no effect.
		1	Read: The current value on $\overline{\text{SPIENA}}$ is 1. Write: Logic 0 is placed on $\overline{\text{SPIENA}}$ pin, if it is in general-purpose output mode.
7-0	SCSCLR		$\overline{\text{SPICS}}$ data out clear. This bit is only active when the $\overline{\text{SPICS}}$ pin is configured as a general-purpose output pin.
		0	Read: The current value on SCSDOUT is 0. Write: Writing a 0 to this bit has no effect.
		1	Read: The current value on SCSDOUT is 1. Write: Logic 0 is placed on $\overline{\text{SPICS}}$ pin, if it is in general-purpose output mode.

27.9.12 SPI Pin Control Register 6 (SPIPC6)

NOTE: Register bits vary by device

Register bits 31:24 and 23:16 of SPIPC0 to SPIPC9 reflect the number of SIMO/SOMI data lines per device. On devices with 8 data-line support, all of bits 31 to 16 are implemented. On devices with less than 8 data lines, only a subset of these bits are available. Unimplemented bits return 0 upon read and are not writable.

Figure 27-37. SPI Pin Control Register 6 (SPIPC6) [offset = 2Ch]

31-24	23-16
SOMIPDR	SIMOPDR
R/W-0	R/W-0

15-12	11	10	9	8
Reserved	SOMIPDR0	SIMOPDR0	CLKPDR	ENAPDR
R-0	R/W-0	R/W-0	R/W-0	R/W-0

7-0
SCSPDR
R/W-0

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

Table 27-19. SPI Pin Control Register 6 (SPIPC6) Field Descriptions

Bit	Field	Value	Description
31-24	SOMIPDR		SPISOMI[x] open drain enable. This bit enables open drain capability for each SPISOMI[x] pin, if the following conditions are met: • SOMIDIRx = 1 (SPISOMI[x] pin is configured in GIO mode as an output pin) • SOMIDOUTx = 1 **Bit 11 or bit 24 can both be used to enable open-drain for SPISOMI[0]. If a 32-bit write is performed, bit 11 will have priority over bit 24.**
		0	Output value on the SPISOMI[x] pin is logic 1.
		1	Output pin SPISOMI[x] is in a high-impedance state.
23-16	SIMOPDR		SPISIMO[x] open drain enable. This bit enables open drain capability for each SPISIMO[x] pin, if the following conditions are met: • SIMODIRx = 1 (SPISIMO[x] pin is configured in GIO mode as an output pin) • SIMODOUTx = 1 **Bit 10 or bit 16 can both be used to enable open-drain for SPISIMO[0]. If a 32-bit write is performed, bit 10 will have priority over bit 16.**
		0	Output value on the SPISIMO[x] pin is logic 1.
		1	Output pin SPISIMO[x] is in a high-impedance state.
15-12	Reserved	0	Reads return 0. Writes have no effect.
11	SOMIPDR0		SPISOMI[0] open-drain enable. This bit enables open-drain capability for the SPISOMI[0] pin, if the following conditions are met: • SPISOMI[0] pin is configured in GIO mode as output pin • Output value on SPISOMI[0] pin is logic 1
		0	Output value on the SPISOMI[0] pin is logic 1.
		1	Output pin SPISOMI[0] is in a high-impedance state.
10	SIMOPDR0		SPISIMO[0] open-drain enable. This bit enables open drain capability for the SPISIMO[0] pin, if the following conditions are met: • SPISIMO[0] pin is configured in GIO mode as output pin • Output value on SPISIMO[0] pin is logic 1
		0	Output value on the SPISIMO[0] pin is logic 1.
		1	Output pin SPISIMO[0] is in a high-impedance state.
9	CLKPDR		SPICLK open drain enable. This bit enables open drain capability for the SPICLK pin, if the following conditions are met: • SPICLK pin is configured in GIO mode as an output pin • SPICLKDOUT = 1
		0	Output value on the SPICLK pin is logic 1.
		1	Output pin SPICLK is in a high-impedance state.
8	ENAPDR		$\overline{\text{SPIENA}}$ open drain enable. This bit enables open drain capability for the $\overline{\text{SPIENA}}$ pin, if the following conditions are met: • $\overline{\text{SPIENA}}$ pin is configured in GIO mode as an output pin • SPIENADOUT = 1
		0	Output value on the $\overline{\text{SPIENA}}$ pin is logic 1.
		1	Output pin $\overline{\text{SPIENA}}$ is in a high-impedance state.
7-0	SCSPDR		$\overline{\text{SPICS}}$ open drain enable. This bit enables open drain capability for each $\overline{\text{SPICS}}$ pin, if the following conditions are met: • $\overline{\text{SPICS}}$ pin is configured in GIO mode as an output pin • SCSDOUT = 1
		0	Output value on the $\overline{\text{SPICS}}$ pin is logic 1.
		1	Output pin $\overline{\text{SPICS}}$ is in a high-impedance state.

27.9.13 SPI Pin Control Register 7 (SPIPC7)

NOTE: Register bits vary by device

Register bits 31:24 and 23:16 of this register reflect the number of SIMO/SOMI data lines per device. On devices with 8 data-line support, all of bits 31 to 16 are implemented. On devices with less than 8 data lines, only a subset of these bits are available. Unimplemented bits return 0 upon read and are not writable.

NOTE: Default Register Value

The default values of these register bits vary by device. See your device datasheet for information about default pin states, which correspond to the register reset values (see the pin-list table).

Figure 27-38. SPI Pin Control Register 7 (SPIPC7) [offset = 30h]

31 24	23 16
SOMIDIS	SIMODIS
R/W-x	R/W-x

15 12	11	10	9	8
Reserved	SOMIPDIS0	SIMOPDIS0	CLKPDIS	ENAPDIS
R-0	R/W-x	R/W-x	R/W-x	R/W-x

7 0
SCSPDIS
R/W-x

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

Table 27-20. SPI Pin Control Register 7 (SPIPC7) Field Descriptions

Bit	Field	Value	Description
31-24	SOMIDIS		SPISOMI[x] pull control disable. This bit disables pull control capability for each SPISOMI[x] pin if it is in input mode, regardless of whether it is in functional or GIO mode. **Note: Bit 11 or bit 24 can be used to set pull-disable for SPISOMI[0]. If a 32-bit write is performed, bit 11 will have priority over bit 24.**
		0	Pull control on the SPISOMI[x] pin is enabled.
		1	Pull control on the SPISOMI[x] pin is disabled.
23-16	SIMODIS		SPISIMO[x] pull control disable. This bit disables pull control capability for each SPISIMO[x] pin if it is in input mode, regardless of whether it is in functional or GIO mode. **Note: Bit 10 or bit 16 can be used to set pull-disable for SPISIMO[0]. If a 32-bit write is performed, bit 10 will have priority over bit 16.**
		0	Pull control on the SPISIMO[x] pin is enabled.
		1	Pull control on the SPISIMO[x] pin is disabled.
15-12	Reserved	0	Reads return 0. Writes have no effect.
11	SOMIPDIS0		SPISOMI[0] pull control disable. This bit disables pull control capability for the SPISOMI[0] pin if it is in input mode, regardless of whether it is in functional or GIO mode.
		0	Pull control on the SPISOMI[0] pin is enabled.
		1	Pull control on the SPISOMI[0] pin is disabled.
10	SIMOPDIS0		SPISIMO[0] pull control disable. This bit disables pull control capability for the SPISIMO[0] pin if it is in input mode, regardless of whether it is in functional or GIO mode.
		0	Pull control on the SPISIMO[0] pin is enabled.
		1	Pull control on the SPISIMO[0] pin is disabled.

Table 27-20. SPI Pin Control Register 7 (SPIPC7) Field Descriptions (continued)

Bit	Field	Value	Description
9	CLKPDIS		SPICLK pull control disable. This bit disables pull control capability for the SPICLK pin if it is in input mode, regardless of whether it is in functional or GIO mode.
		0	Pull control on the SPICLK pin is enabled.
		1	Pull control on the SPICLK pin is disabled.
8	ENAPDIS		$\overline{\text{SPIENA}}$ pull control disable. This bit disables pull control capability for the $\overline{\text{SPIENA}}$ pin if it is in input mode, regardless of whether it is in functional or GIO mode.
		0	Pull control on the $\overline{\text{SPIENA}}$ pin is enabled.
		1	Pull control on the $\overline{\text{SPIENA}}$ pin is disabled.
7-0	SCSPDIS		$\overline{\text{SPICS}}$ pull control disable. This bit disables pull control capability for each $\overline{\text{SPICS}}$ pin if it is in input mode, regardless of whether it is in functional or GIO mode.
		0	Pull control on the $\overline{\text{SPICS}}$ pin is enabled.
		1	Pull control on the $\overline{\text{SPICS}}$ pin is disabled.

27.9.14 SPI Pin Control Register 8 (SPIPC8)

NOTE: Register bits vary by device

Register bits 31:24 and 23:16 of this register reflect the number of SIMO/SOMI data lines per device. On devices with 8 data-line support, all of bits 31 to 16 are implemented. On devices with less than 8 data lines, only a subset of these bits are available. Unimplemented bits return 0 upon read and are not writable.

NOTE: Default Register Value

The default values of these register bits vary by device. See your device datasheet for information about default pin states, which correspond to the register reset values (see the pin-list table).

Figure 27-39. SPI Pin Control Register 8 (SPIPC8) [offset = 34h]

31-24	23-16
SOMIPSEL	SIMOPSEL
R/W-x	R/W-x

15-12	11	10	9	8
Reserved	SOMIPSEL0	SIMOPSEL0	CLKPSEL	ENAPSEL
R-0	R/W-x	R/W-x	R/W-x	R/W-x

7-0
SCSPSEL
R/W-x

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

Table 27-21. SPI Pin Control Register 8 (SPIPC8) Field Descriptions

Bit	Field	Value	Description
31-24	SOMIPSEL		SPISOMI[x] pull select. This bit selects the type of pull logic for each SPISOMI[x] pin. **Note: Bit 11 or bit 24 can be used to set pull-select for SPISOMI[0]. If a 32-bit write is performed, bit 11 will have priority over bit 24.**
		0	Pull down on the SPISOMI[x] pin.
		1	Pull up on the SPISOMI[x] pin.

Table 27-21. SPI Pin Control Register 8 (SPIPC8) Field Descriptions (continued)

Bit	Field	Value	Description
23-16	SIMOPSEL		SPISIMO[x] pull select. This bit selects the type of pull logic for each SPISIMO[x] pin. **Note: Bit 10 or bit 16 can be used to set pull-select for SPISIMO[0]. If a 32-bit write is performed, bit 10 will have priority over bit 16.**
		0	Pull down on the SPISIMO[x] pin.
		1	Pull up on the SPISIMO[x] pin.
15-12	Reserved	0	Reads return 0. Writes have no effect.
11	SOMIPSEL0		SPISOMI[0] pull select. This bit selects the type of pull logic at the SPISOMI[0] pin.
		0	Pull down on the SPISOMI[0] pin.
		1	Pull up on the SPISOMI[0] pin.
10	SIMOPSEL0		SPISIMO[0] pull select. This bit selects the type of pull logic at the SPISIMO[0] pin.
		0	Pull down on the SPISIMO[0] pin.
		1	Pull up on the SPISIMO[0] pin.
9	CLKPSEL		SPICLK pull select. This bit selects the type of pull logic at the SPICLK pin.
		0	Pull down on the SPICLK pin.
		1	Pull up on the SPICLK pin.
8	ENAPSEL		$\overline{\text{SPIENA}}$ pull select. This bit selects the type of pull logic at the $\overline{\text{SPIENA}}$ pin.
		0	Pull down on the $\overline{\text{SPIENA}}$ pin.
		1	Pull up on the $\overline{\text{SPIENA}}$ pin.
7-0	SCSPSEL		$\overline{\text{SPICS}}$ pull select. This bit selects the type of pull logic for each $\overline{\text{SPICS}}$ pin.
		0	Pull down on the $\overline{\text{SPICS}}$ pin.
		1	Pull up on the $\overline{\text{SPICS}}$ pin.

27.9.15 SPI Transmit Data Register 0 (SPIDAT0)

Figure 27-40. SPI Transmit Data Register 0 (SPIDAT0) [offset = 38h]

31	16
Reserved	
R-0	

15	0
TXDATA	
R/W-0	

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

Table 27-22. SPI Transmit Data Register 0 (SPIDAT0) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return 0. 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, TXBUF holds the written data. SPIEN (SPICGR1[24]) must be set to 1 before this register can be written to. Writing a 0 to the SPIEN register forces the lower 16 bits of the SPIDAT0 to 0x00. **Note: When this register is read, the contents TXBUF, which holds the latest written data, will be returned.** **Note: Regardless of character length, the transmit word should be right-justified before writing to the SPIDAT1 register.** **Note: The default data format control register for SPIDAT0 is SPIFMT0. However, it is possible to reprogram the DFSEL[1:0] fields of SPIDAT1 before using SPIDAT0, to select a different SPIFMTx register.** **Note: It is highly recommended to use SPIDAT1 register, SPIDAT0 is supported for compatibility reasons.**

27.9.16 SPI Transmit Data Register 1 (SPIDAT1)

NOTE: Writing to only the control fields, bits 28 through 16, does not initiate any SPI transfer in master mode. This feature can be used to set up SPICLK phase or polarity before actually starting the transfer by only updating the DFSEL bit field to select the required phase and polarity combination.

Figure 27-41. SPI Transmit Data Register 1 (SPIDAT1) [offset = 3Ch]

31 29	28	27	26	25 24	23 16
Reserved	CSHOLD	Rsvd	WDEL	DFSEL	CSNR
R-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0

15 0
TXDATA
R/W-0

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

Table 27-23. SPI Transmit Data Register 1 (SPIDAT1) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Reads return 0. Writes have no effect.
28	CSHOLD		Chip select hold mode. The CSHOLD bit is supported in master mode only in compatibility-mode of SPI, (it is ignored in slave mode). 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. If two consecutive transfers are dedicated to the same chip select this chip select signal will be deactivated for at least 2VCLK cycles before it is activated again.
		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 number equals the previous one, the active chip select signal is extended until the end of transfer with CSHOLD cleared, or until the chip-select number changes.
27	Reserved	0	Reads return 0. Writes have no effect.
26	WDEL		Enable the delay counter at the end of the current transaction. **Note: The WDEL bit is supported in master mode only. In slave mode, this bit will be ignored.**
		0	No delay will be inserted. However, the $\overline{\text{SPICS}}$ pins will still be de-activated for at least for 2VCLK cycles if CSHOLD = 0. **Note: The duration for which the $\overline{\text{SPICS}}$ pin remains deactivated depends upon the time taken to supply a new word after completing the shift operation. If TXBUF is already full, then the $\overline{\text{SPICS}}$ pin will be deasserted for at least two VCLK cycles (if WDEL = 0).**
		1	After a transaction, WDELAY of the corresponding data format will be loaded into the delay counter. No transaction will be performed until the WDELAY counter overflows. The $\overline{\text{SPICS}}$ pins will be de-activated for at least (WDELAY + 2) × VCLK_Period duration.
25-24	DFSEL		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.
23-16	CSNR	0-FFh	Chip select (CS) number. CSNR defines the chip select pins that will be activated during the data transfer. CSNR is a bit-mask that controls all chip select pins. See Table 27-24. **Note: If your MibSPI has less than 8 chip select pins, all unused upper bits will be 0. For example, MiBSPI3 has 6 chip select pins, if you write FFh to CSNR, the actual number stored in CSNR is 3Fh.**

Table 27-23. SPI Transmit Data Register 1 (SPIDAT1) Field Descriptions (continued)

Bit	Field	Value	Description
15-0	TXDATA	0-FFFFh	Transfer data. When written, these bits are copied to the shift register if it is empty. If the shift register is not empty, then they are held in TXBUF. SPIEN must be set to 1 before this register can be written to. Writing a 0 to SPIEN forces the lower 16 bits of SPIDAT1 to 0x0000. A write to this register (or to the TXDATA field only) drives the contents of the CSNR field on the $\overline{\text{SPICS}}$ pins, if the pins are configured as functional pins (automatic chip select, see Section 27.2). When this register is read, the contents of TXBUF, which holds the latest data written, will be returned. **Note: Regardless of the character length, the transmit data should be right-justified before writing to the SPIDAT1 register.**

Table 27-24. Chip Select Number Active

CSNR Value	Chip Select Active:						CSNR Value	Chip Select Active:					
	CS[5][1]	CS[4][1]	CS[3][1]	CS[2][1]	CS[1][1]	CS[0]		CS[5][1]	CS[4][1]	CS[3][1]	CS[2][1]	CS[1][1]	CS[0]
0h	No chip select pin is active.						20h	x					
1h						x	21h	x					x
2h					x		22h	x				x	
3h					x	x	23h	x				x	x
4h				x			24h	x			x		
5h				x		x	25h	x			x		x
6h				x	x		26h	x			x	x	
7h				x	x	x	27h	x			x	x	x
8h			x				28h	x		x			
9h			x			x	29h	x		x			x
Ah			x		x		2Ah	x		x		x	
Bh			x		x	x	2Bh	x		x		x	x
Ch			x	x			2Ch	x		x	x		
Dh			x	x		x	2Dh	x		x	x		x
Eh			x	x	x		2Eh	x		x	x	x	
Fh			x	x	x	x	2Fh	x		x	x	x	x
10h		x					30h	x	x				
11h		x				x	31h	x	x				x
12h		x			x		32h	x	x			x	
13h		x			x	x	33h	x	x			x	x
14h		x		x			34h	x	x		x		
15h		x		x		x	35h	x	x		x		x
16h		x		x	x		36h	x	x		x	x	
17h		x		x	x	x	37h	x	x		x	x	x
18h		x	x				38h	x	x	x			
19h		x	x			x	39h	x	x	x			x
1Ah		x	x		x		3Ah	x	x	x		x	
1Bh		x	x		x	x	3Bh	x	x	x		x	x
1Ch		x	x	x			3Ch	x	x	x	x		
1Dh		x	x	x		x	3Dh	x	x	x	x		x
1Eh		x	x	x	x		3Eh	x	x	x	x	x	
1Fh		x	x	x	x	x	3Fh	x	x	x	x	x	x

[1] If your MibSPI does not have this chip select pin, this bit is 0.

27.9.17 SPI Receive Buffer Register (SPIBUF)

Figure 27-42. SPI Receive Buffer Register (SPIBUF) [offset = 40h]

31	30	29	28	27	26	25	24
RXEMPTY	RXOVR	TXFULL	BITERR	DESYNC	PARITYERR	TIMEOUT	DLENERR
R-1	R-0	R-0	R-0	R-0	R-0	R-0	R-0

23	16
LCSNR	
R-0	

15	0
RXDATA	
R-0	

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

Table 27-25. SPI Receive Buffer Register (SPIBUF) Field Descriptions

Bit	Field	Value	Description
31	RXEMPTY		Receive data buffer empty. When the host reads the RXDATA field or the entire SPIBUF register, it automatically sets the RXEMPTY flag. When a data transfer is completed, the received data is copied into RXDATA and the RXEMPTY flag is cleared.
		0	New data has been received and copied into RXDATA.
		1	No data has been received since the last read of RXDATA.
			This flag gets set to 1 under the following conditions: • Reading the RXDATA field of the SPIBUF register • Writing a 1 to clear the RXINTFLG bit in the SPI Flag Register (SPIFLG) Write-clearing the RXINTFLG bit before reading the SPIBUF indicates the received data is being ignored. Conversely, RXINTFLG can be cleared by reading the RXDATA field of SPIBUF (or the entire register).
30	RXOVR		Receive data buffer overrun. When a data transfer is completed and the received data is copied into RXBUF while it is already full, RXOVR is set. Overruns always occur to RXBUF, not to SPIBUF; the contents of SPIBUF are overwritten only after it is read by the Peripheral (VBUSP) master (CPU, DMA, or other host processor). If enabled, the RXOVRN interrupt is generated when RXBUF is overwritten, and reading either SPI Flag Register (SPIFLG) or SPIVECTx shows the RXOVRN condition. Two read operations from the SPIBUF register are required to reach the overwritten buffer word (one to read SPIBUF, which then transfers RXDATA into SPIBUF for the second read). **Note: This flag is cleared to 0 when the RXDATA field of the SPIBUF register is read.** **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 such as TIMEOUT, BITERR, and DLEN_ERR occur, then RXOVR in RXBUF and SPI Flag Register (SPIFLG) registers will be set to indicate that the status flags are getting overwritten by the new transfer. This overrun should be treated like a normal receive overrun.**
		0	No receive data overrun condition occurred since last read of the data field.
		1	A receive data overrun condition occurred since last read of the data field.
29	TXFULL		Transmit data buffer full. This flag is a read-only flag. Writing into the SPIDAT0 or SPIDAT1 field while the TX shift register is full will automatically set the TXFULL flag. Once the word is copied to the shift register, the TXFULL flag will be cleared. Writing to SPIDAT0 or SPIDAT1 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.

Table 27-25. SPI Receive Buffer Register (SPIBUF) Field Descriptions (continued)

Bit	Field	Value	Description
28	BITERR		Bit error. There was a mismatch of internal transmit data and transmitted data. **Note: This flag is cleared to 0 when the RXDATA field of the SPIBUF register is read.**
		0	No bit error occurred.
		1	A bit error occurred. The SPI samples the signal of the transmit pins (master: SIMOx, slave: SOMIx) at the receive point (one-half clock cycle after the transmit point). If the sampled value differs from the transmitted value, a bit error is detected and the BITERR flag is set. Possible reasons for a bit error include noise, an excessively high bit rate, capacitive load, or another master/slave trying to transmit at the same time.
27	DESYNC		Desynchronization of slave device. This bit is valid in master mode only. The master monitors the ENA signal coming from the slave device and sets the DESYNC flag if 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: In the Compatibility Mode MibSPI, 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 is generated when the buffer transfer is completed. But desynchronization is detected after the buffer transfer is completed. So, if the VBUS master reads the received data quickly when an RXINT is detected, then the status flag may not reflect the correct desync condition. In multi-buffer mode, the desync flag is always assured to be for the current buffer.** **Note: This flag is cleared to 0 when the RXDATA field of the SPIBUF register is read.**
		0	No slave desynchronization is detected.
		1	A slave device is desynchronized.
26	PARITYERR		Parity error. The calculated parity differs from the received parity bit. If the parity generator is enabled (selected individually for each buffer) 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 PARITYERR flag is set. **Note: This flag is cleared to 0 when the RXDATA field of the SPIBUF register is read.**
		0	No parity error is detected.
		1	A parity error occurred.
25	TIMEOUT		Time-out because of non-activation of $\overline{\text{SPIENA}}$ pin. The SPI generates a time-out when the slave does not respond in time by activating the 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. In addition, the TIMEOUT flag in the status field of the corresponding buffer and in the SPI Flag Register (SPIFLG) is set. **Note: This bit is valid only in master mode.** **Note: This flag is cleared to 0 when the RXDATA field of the SPIBUF register is read.**
		0	No $\overline{\text{SPIENA}}$ pin time-out occurred.
		1	An $\overline{\text{SPIENA}}$ signal time-out occurred.
24	DLENERR		Data length error flag. **Note: This flag is cleared to 0 when the RXDATA field of the SPIBUF register is read.**
		0	No data-length error occurred.
		1	A data length error occurred.
23-16	LCSNR	0-FFh	Last chip select number. LCSNR in the status field is a copy of CSNR in the corresponding control field. It contains the chip select number that was activated during the last word transfer.
15-0	RXDATA	0-FFFFh	SPI receive data. This is the received word, transferred from the receive shift-register at the end of a transfer. Regardless of the programmed character length and the direction of shifting, the received data is stored right-justified in the register.

27.9.18 SPI Emulation Register (SPIEMU)

Figure 27-43. SPI Emulation Register (SPIEMU) [offset = 44h]

31 … 16
Reserved
R-8000h

15 … 0
EMU_RXDATA
R-0

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

Table 27-26. SPI Emulation Register (SPIEMU) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	8000h	Reads return 0. Writes have no effect.
15-0	EMU_RXDATA	0-FFFFh	SPI receive data. The SPI emulation register 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.

27.9.19 SPI Delay Register (SPIDELAY)

Figure 27-44. SPI Delay Register (SPIDELAY) [offset = 48h]

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 27-27. SPI Delay Register (SPIDELAY) Field Descriptions

Bit	Field	Value	Description
31-24	C2TDELAY	0-FFh	Chip-select-active to transmit-start delay. See Figure 27-45 for an example. C2TDELAY is used only in master mode. It defines a setup time (for the slave device) that delays the data transmission from the chip select active edge by a multiple of VCLK cycles. The setup time value is calculated as follows. $t_{C2TDELAY}$ = (C2TDELAY + 2) × VCLK Period Example: VCLK = 25 MHz -> VCLK Period = 40ns; C2TDELAY = 07h; > $t_{C2TDELAY}$ = 360 ns When the chip select signal becomes active, the slave has to prepare data transfer within 360 ns. **Note: If phase = 1, the delay between SPICS falling edge to the first edge of SPICLK will have an additional 0.5 SPICLK period delay. This delay is as per the SPI protocol.**

Table 27-27. SPI Delay Register (SPIDELAY) Field Descriptions (continued)

Bit	Field	Value	Description
23-16	T2CDELAY	0-FFh	Transmit-end-to-chip-select-inactive-delay. See Figure 27-46 for an example. T2CDELAY is used only in master mode. It defines a hold time for the slave device that delays the chip select deactivation by a multiple of VCLK cycles after the last bit is transferred. The hold time value is calculated as follows: $t_{T2CDELAY}$ = (T2CDELAY +1) × VCLK Period Example: VCLK = 25 MHz -> VCLK Period = 40ns; 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 SPICLK and rise-edge of SPICS there will be an additional delay of 0.5 SPICLK period. This is as per the SPI protocol.** Both C2TDELAY and T2CDELAY counters do not have any dependency on the $\overline{\text{SPIENA}}$ pin value. Even if the $\overline{\text{SPIENA}}$ pin is asserted by the slave, the master will continue to delay the start of SPICLK until the C2TDELAY counter overflows. Similarly, even if the $\overline{\text{SPIENA}}$ pin is deasserted by the slave, the master will continue to hold the $\overline{\text{SPICS}}$ pins active until the T2CDELAY counter overflows. In this way, it is assured that the setup and hold times of the $\overline{\text{SPICS}}$ 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.
15-8	T2EDELAY	0-FFh	Transmit-data-finished to 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 $\overline{\text{SPIENA}}$ signal has to become inactive and after $\overline{\text{SPICS}}$ becomes inactive. SPICLK depends on which data format is selected. If the slave device is missing one or more clock edges, it becomes de-synchronized. In this case, although the master has finished the data transfer, the slave is still waiting for the missed clock pulses and the ENA signal is not disabled. The T2EDELAY defines a time-out value that triggers the DESYNC flag, if the $\overline{\text{SPIENA}}$ signal is not deactivated in time. The DESYNC flag is set to indicate that the slave device did not de-assert its $\overline{\text{SPIENA}}$ pin in time to acknowledge that it received all bits of the sent word. See Figure 27-47 for an example of this condition. **Note: DESYNC is also set if the SPI detects a de-assertion of $\overline{\text{SPIENA}}$ before the end of the transmission.** 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 ENA signal within 2 µs, otherwise DESYNC is set and an interrupt is asserted (if enabled).
7-0	C2EDELAY	0-FFh	Chip-select-active to ENA-signal-active time-out. C2EDELAY is used only in master mode and it applies only if the addressed slave generates an ENA signal as a hardware handshake response. C2EDELAY defines the maximum time between when the SPI activates the chip-select signal and the addressed slave has to respond by activating the ENA signal. C2EDELAY defines a time-out value as a multiple of SPI clocks. The SPI clock depends on whether data format 0 or data format 1 is selected. See Figure 27-48 for an example of this condition. **Note: If the slave device does not respond with the ENA signal before the time-out value is reached, the TIMEOUT flag in the SPIFLG register is set and a interrupt is asserted (if enabled).** If a time-out occurs, the SPI clears the transmit request of the timed-out buffer, sets the TIMEOUT flag for the current buffer, and continues with the transfer of the next buffer in the sequence that is enabled. The timeout value is calculated as follows: $t_{C2EDELAY}$ = C2EDELAY/SPIclock Example: SPIclock = 8 Mbit/s; C2EDELAY = 30h; > $t_{C2EDELAY}$ = 6 ms The slave device has to activate the ENA signal within 6 ms after the SPI has activated the chip select signal ($\overline{\text{SPICS}}$), otherwise the TIMEOUT flag is set and an interrupt is asserted (if enabled).

Figure 27-45. Example: $t_{C2TDELAY}$= 8 VCLK Cycles

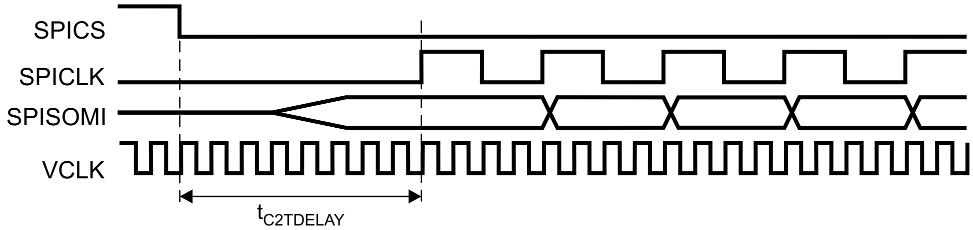


Figure 27-46. Example: $t_{T2CDELAY}$= 4 VCLK Cycles

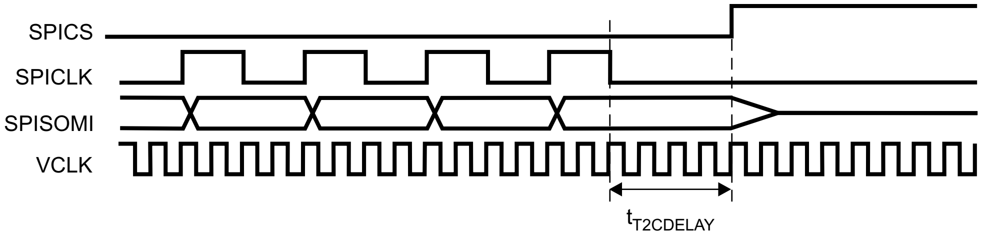


Figure 27-47. Transmit-Data-Finished-to-ENA-Inactive-Timeout

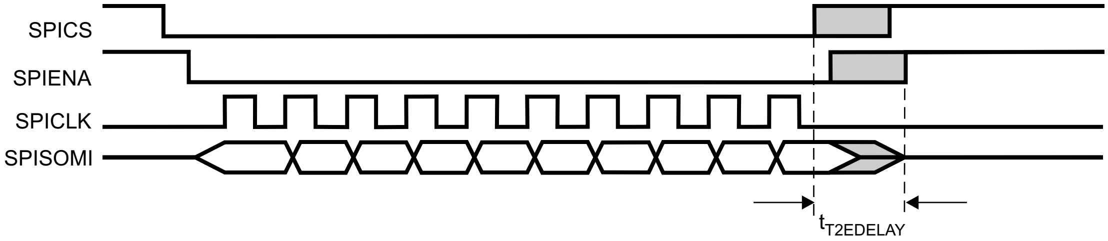
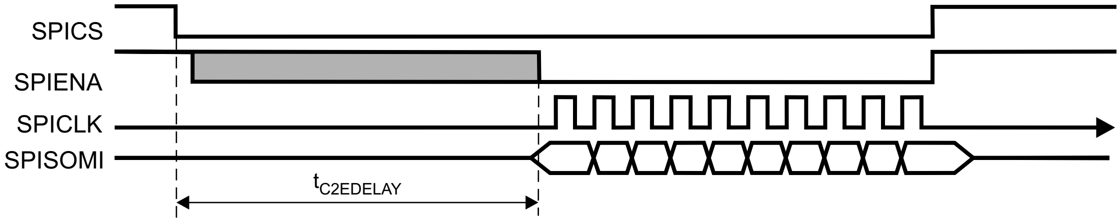


Figure 27-48. Chip-Select-Active-to-ENA-Signal-Active-Timeout



27.9.20 SPI Default Chip Select Register (SPIDEF)

Figure 27-49. SPI Default Chip Select Register (SPIDEF) [offset = 4Ch]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		CSDEF	
R-0		R/W-FFh	

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

Table 27-28. SPI Default Chip Select Register (SPIDEF) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	CDEF	0-FFh	Chip select default pattern. Master-mode only. The CSDEF bits are output to the $\overline{\text{SPICS}}$ pins when no transmission is being performed. It allows the user to set a programmable chip-select pattern that deselects all of the SPI slaves.
		0	$\overline{\text{SPICS}}$ is cleared to 0 when no transfer is active.
		1	$\overline{\text{SPICS}}$ is set to 1 when no transfer is active.

27.9.21 SPI Data Format Registers (SPIFMT)

Figure 27-50. SPI Data Format Registers (SPIFMT[3:0]) [offset = 5Ch-50h]

31 – 24
WDELAY
R/WP-0

23	22	21	20	19	18	17	16
PARPOL	PARITYENA	WAITENA	SHIFTDIR	HDUPLEX_ ENAx	DIS_CS_ TIMERS	POLARITY	PHASE
R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

15 – 8	7 – 5	4 – 0
PRESCALE	Reserved	CHARLEN
R/WP-0	R-0	R/WP-0

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

Table 27-29. SPI Data Format Registers (SPIFMT) Field Descriptions

Bit	Field	Value	Description
31-24	WDELAY	0-FFh	Delay in between transmissions for data format x (x= 0,1,2,3).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_{VCLK} + 2 × P_{VCLK} P $_{VCLK}$ -> Period of VCLK.
23	PARPOL		Parity polarity: even or odd. PARPOLx can be modified in privilege mode only. It can be used for data format x (x= 0,1,2,3).
		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	PARITYENA		Parity enable for data format x.
			No parity generation/ verification is performed for this data format.
		0	A parity bit is transmitted at the end of each transmitted word. 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 RXERR flag is set in the corresponding control field. The parity type (even or odd) can be selected via the PARPOL bit.
		1	**Note: If an uncorrectable error flag is set in a slave-mode SPI, then the wrong parity bit will be transmitted to indicate to the master that there has been some issue with the data parity. The SPISOMI pins will be forced to transmit all 0s, and the parity bit will be transmitted as 1 if even parity is selected and as 0 if odd parity is selected (using the PARPOLx bit of this register). This behavior occurs regardless of an uncorrectable parity error on either TXRAM or RXRAM.**
21	WAITENA		The master waits for the ENA signal from slave for data format x. WAITENA is valid in master mode only. WAITENA enables a flexible SPI network where slaves with ENA signal and slaves without ENA signal can be mixed. WAITENA defines, for each transferred word, whether the addressed slave generates the ENA signal or not.
		0	The SPI does not wait for the ENA signal from the slave and directly starts the transfer.
		1	Before the SPI starts the data transfer it waits for the ENA signal to become low. If the 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 for data format x. With bit SHIFTDIRx, the shift direction for data format x (x=0,1,2,3) can be selected.
		0	MSB is shifted out first.
		1	LSB is shifted out first.

Table 27-29. SPI Data Format Registers (SPIFMT) Field Descriptions (continued)

Bit	Field	Value	Description
19	HDUPLEX_ENAx		Half Duplex transfer mode enable for Data Format x. This bit controls the I/O function of SOMI/SIMO lines for a specific requirement where in the case of Master mode, TX pin - SIMO will act as an RX pin, and in the case of Slave mode, RX pin - SIMO will act as a TX pin.
		0	Normal Full Duplex transfer.
		1	If MASTER = 1, SPISIMO pin will act as an RX pin (No TX possible) If MASTER = 0, SPISIMO pin will act as a TX pin (No RX possible).
			For all normal operations, HDUPLEX_ENAx bits should always remain 0. It is intended for the usage when the SPISIMO pin is used for both TX and RX operations at different times.
18	DIS CS TIMERS		Disable chip-select timers for this format. 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 they are 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.
17	POLARITY		SPI data format x clock polarity. POLARITYx defines the clock polarity of data format x. The following restrictions apply when switching clock phase and/or polarity: • In 3-pin/4-pin with $\overline{\text{SPIENA}}$ pin configuration of a slave SPI, the clock phase and polarity cannot be changed on-the-fly between two transfers. The slave should be reset and reconfigured if clock phase/polarity needs to be switched. In summary, SPI format switching is not fully supported in slave mode. • Even while using chip select pins, the polarity of SPICLK can be switched only while the slave is not selected by a valid chip select. The master SPI should ensure that while switching SPICLK polarity, it has deselected all of its slaves. Otherwise, the switching of SPICLK polarity may be incorrectly treated as a clock edge by some slaves.
		0	If POLARITYx is cleared to 0, the SPI clock signal is low-inactive, that is, before and after data transfer the clock signal is low.
		1	If POLARITYx is set to 1, the SPI clock signal is high-inactive, that is, before and after data transfer the clock signal is high.
16	PHASE		SPI data format x clock delay. PHASEx defines the clock delay of data format x.
		0	If PHASEx is cleared to 0, the 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	If PHASEx is set to 1, the 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		SPI data format x prescaler. PRESCALEx determines the bit transfer rate of data format x if the SPI is the network master. PRESCALEx is use to derive SPICLK from VCLK. If the SPI is configured as slave, PRESCALEx **does not need** to be configured. The clock rate for data format x can be calculated as: $BR_{Formatx}$ = VCLK / (PRESCALEx + 1) **Note: When PRESCALEx is cleared to 0, the SPI clock rate defaults to VCLK/2.**
7-5	Reserved	0	Reads return 0. Writes have no effect.
4-0	CHARLEN	0-1Fh	SPI data format x data-word length. CHARLENx defines the word length of data format x. Legal values are 0x02 (data word length = 2 bit) to 10h (data word length = 16). Illegal values, such as 00 or 1Fh are not allowed; their effect is indeterminate.

27.9.22 Interrupt Vector 0 (INTVECT0)

NOTE: The TG interrupt is not available in MibSPI in compatibility mode. Therefore, there is no possibility to access this register in compatibility mode.

Figure 27-51. Interrupt Vector 0 (NTVECT0) [offset = 60h]

31	16
Reserved	
R-0	

15	6	5	1	0
Reserved		INTVECT0		SUSPEND0
R-0		R-0		R-0

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

Table 27-30. Transfer Group Interrupt Vector 0 (INTVECT0)

Bit	Field	Value	Description
31-6	Reserved	0	Reads return 0. Writes have no effect.
5-1	INTVECT0		INTVECT0. Interrupt vector for interrupt line INT0. Returns the vector of the pending interrupt at interrupt line INT0. If more than one interrupt is pending, INTVECT0 always references the highest prior interrupt source first. **Note: This field reflects the status of the SPIFLG register in vector format. Any updates to the SPIFLG register will automatically cause updates to this field.**
		0	There is no pending interrupt.
		1h + x	Transfer group x (x = 0 to 15) has a pending interrupt. SUSPEND0 reflects the type of interrupt (*suspend* or *finished*).
		11h	Error Interrupt pending. The lower half of SPIFLG contains more details about the type of error.
		13h	The pending interrupt is a Receive Buffer Overrun interrupt.
		12h	**SPI mode:** The pending interrupt is a Receive Buffer Full interrupt. **Mib mode:** Reserved. This bit combination should not occur.
		14h	**SPI mode:** The pending interrupt is a Transmit Buffer Empty interrupt. **Mib mode:** Reserved. This bit combination should not occur.
		All Other Combinations	**SPI mode:** Reserved. These bit combinations should not occur.
0	SUSPEND0		Transfer suspended / Transfer finished interrupt flag. Every time INTVECT0 is read by the host, the corresponding interrupt flag of the referenced transfer group is cleared and INTVECT0 is updated with the vector coming next in the priority chain.
		0	The interrupt type is a transfer finished interrupt. In other words, the buffer array referenced by INTVECT0 has asserted an interrupt because all of data from the transfer group has been transferred.
		1	The interrupt type is a transfer suspended interrupt. In other words, the transfer group referenced by INTVECT0 has asserted an interrupt because the buffer to be transferred next is in suspend-to-wait mode.

NOTE: Reading from the INTVECT0 register when Transmit Empty is indicated does not clear the TXINTFLG flag in the SPI Flag Register (SPIFLG). Writing a new word to the SPIDATx register clears the Transmit Empty interrupt.

NOTE: In multi-buffer mode, INTVECT0 contains the interrupt for the highest priority transfer group. A read from INTVECT0 automatically causes the next-highest priority transfer group's interrupt status to get loaded into INTVECT0 and its corresponding SUSPEND flag to get loaded into SUSPEND0. The transfer group with the lowest number has the highest priority, and the transfer group with the highest number has the lowest priority.

Reading the INTVECT0 register when the RXOVRN interrupt is indicated in multi-buffer mode does not clear the RXOVRN flag and hence does not clear the vector. The RXOVRN interrupt vector may be cleared in multi-buffer mode either by write-clearing the RXOVRN flag in the SPI Flag Register (SPIFLG) or by reading the RXRAM Overrun Buffer Address Register (RXOVRN_BUF_ADDR).

27.9.23 Interrupt Vector 1 (INTVECT1)

NOTE: The TG interrupt is not available in SPI in compatibility mode compatibility mode. Therefore, there is no possibility to access this register in compatibility mode.

Figure 27-52. Interrupt Vector 1 (INTVECT1) [offset = 64h]

31 16
Reserved
R-0

15 6	5 1	0
Reserved	INTVECT1	SUSPEND1
R-0	R-0	R-0

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

Table 27-31. Transfer Group Interrupt Vector 1 (INTVECT1)

Bit	Field	Value	Description
31-6	Reserved	0	Reads return 0. Writes have no effect.
5-1	INTVECT1		INTVECT1. Interrupt vector for interrupt line INT1. Returns the vector of the pending interrupt at interrupt line INT1. If more than one interrupt is pending, INTVECT1 always references the highest prior interrupt source first. **Note: This field reflects the status of the SPIFLG register in vector format. Any updates to the SPIFLG register will automatically cause updates to this field.**
		0	There is no pending interrupt. **SPI mode only.**
		11h	Error Interrupt pending. The lower half of SPIINT1 contains more details about the type of error. **SPI mode only.**
		13h	The pending interrupt is a Receive Buffer Overrun interrupt. **SPI mode only.**
		12h	The pending interrupt is a Receive Buffer Full interrupt. **SPI mode only.**
		14h	The pending interrupt is a Transmit Buffer Empty interrupt. **SPI mode only.**
		All Other Combinations	Reserved. These bit combinations should not occur. **SPI mode only.**
0	SUSPEND1		Transfer suspended / Transfer finished interrupt flag. Every time INTVECT1 is read by the host, the corresponding interrupt flag of the referenced transfer group is cleared and INTVECT1 is updated with the vector coming next in the priority chain.
		0	The interrupt type is a transfer finished interrupt. In other words, the buffer array referenced by INTVECT1 has asserted an interrupt because all of data from the transfer group has been transferred.
		1	The interrupt type is a transfer suspended interrupt. In other words, the transfer group referenced by INTVECT1 has asserted an interrupt because the buffer to be transferred next is in suspend-to-wait mode.

NOTE: Reading from the INTVECT1 register when Transmit Empty is indicated does not clear the TXINTFLG flag in the SPI Flag Register (SPIFLG). Writing a new word to the SPIDATx register clears the Transmit Empty interrupt.

NOTE: In multi-buffer mode, INTVECT1 contains the interrupt for the highest priority transfer group. A read from INTVECT1 automatically causes the next-highest priority transfer group's interrupt status to get loaded into INTVECT1 and its corresponding SUSPEND flag to get loaded into SUSPEND1. The transfer group with the lowest number has the highest priority, and the transfer group with the highest number has the lowest priority.

Reading the INTVECT1 register when the RXOVRN interrupt is indicated in multi-buffer mode does not clear the RXOVRN flag and hence does not clear the vector. The RXOVRN interrupt vector may be cleared in multi-buffer mode either by write-clearing the RXOVRN flag in the SPI Flag Register (SPIFLG) or by reading the RXOVERN_BUF_ADDR register.

27.9.24 SPI Pin Control Register 9 (SPIPC9)

SPIPC9 only applies to SPI2.

Figure 27-53. SPI Pin Control Register 9 (SPIPC9) [offset = 68h]

31 25	24	23 17	16
Reserved	SOMISRS0	Reserved	SIMOSRS0
R-0	R/W-0	R-0	R/W-0

15 12	11	10	9	8 0
Reserved	SOMISRS0	SIMOSRS0	CLKSRS	Reserved
R-0	R/W-0	R/W-0	R/W-0	R-0

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

Table 27-32. SPI Pin Control Register 9 (SPIPC9) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return the value that was last written. Writes have no effect.
24	SOMISRS0		SPI2 SOMI[0] slew control. This bit controls between the fast or slow slew mode. **Note: Duplicate Control Bits for SPI2 SOMI[0]. Bit 24 is not physically implemented. It is a mirror of bit 11. Any write to bit 24 will be reflected on bit 11. When bit 24 and bit 11 are simultaneously written, the value of bit 11 will control the SPI2 SOMI[0] pin. The read value of bit 24 always reflects the value of bit 11.**
		0	Fast mode is enabled; the normal output buffer is used for this pin.
		1	Slow mode is enabled; slew rate control is used for this pin.
23-17	Reserved	0	Reads return the value that was last written. Writes have no effect.
16	SIMOSRS0		SPI2 SIMO[0] slew control. This bit controls between the fast or slow slew mode. **Note: Duplicate Control Bits for SPI2 SIMO[0]. Bit 16 is not physically implemented. It is a mirror of bit 10. Any write to bit 16 will be reflected on bit 10. When bit 16 and bit 10 are simultaneously written, the value of bit 10 will control the SPI2 SIMO[0] pin. The read value of bit 16 always reflects the value of bit 10.**
		0	Fast mode is enabled; the normal output buffer is used for this pin.
		1	Slow mode is enabled; slew rate control is used for this pin.
15-12	Reserved	0	Reads return 0. Writes have no effect.
11	SOMISRS0		SPI2 SOMI[0] slew control. This bit controls between the fast or slow slew mode.
		0	Fast mode is enabled; the normal output buffer is used for this pin.
		1	Slow mode is enabled; slew rate control is used for this pin.
10	SIMOSRS0		SPI2 SIMO[0] slew control. This bit controls between the fast or slow slew mode.
		0	Fast mode is enabled; the normal output buffer is used for this pin.
		1	Slow mode is enabled; slew rate control is used for this pin.
9	CLKSRS		SPI2 CLK slew control. This bit controls between the fast or slow slew mode.
		0	Fast mode is enabled; the normal output buffer is used for this pin.
		1	Slow mode is enabled; slew rate control is used for this pin.
8-0	Reserved	0	Reads return the value that was last written. Writes have no effect.

27.9.25 Parallel/Modulo Mode Control Register (SPIPMCTRL)

NOTE: Do not configure MODCLKPOLx and MMODEx bits since this device does not support modulo mode.

NOTE: The bits of this register are used in conjunction with the SPIFMTx registers. Each byte of this register corresponds to one of the SPIFMTx registers.

1. Byte0 (Bits 7:0) are used when SPIFMT0 register is selected by DFSEL[1:0] = 00 in the control field of a buffer.

2. Byte1 (Bits 15:8) are used when SPIFMT1 register is selected by DFSEL[1:0] = 01 in the control field of a buffer.

3. Byte2 (Bits 23:16) are used when SPIFMT2 register is selected by DFSEL[1:0] = 10 in the control field of a buffer.

4. Byte3 (Bits 31:24) are used when SPIFMT3 register is selected by DFSEL[1:0] = 11 in the control field of a buffer.

Figure 27-54. Parallel/Modulo Mode Control Register (SPIPMCTRL) [offset = 6Ch]

31	30	29	28	26	25	24
Reserved		MODCLKPOL3	MMODE3		PMODE3	
R-0		R/WP-0	R/WP-0		R/WP-0	

23	22	21	20	18	17	16
Reserved		MODCLKPOL2	MMODE2		PMODE2	
R-0		R/WP-0	R/WP-0		R/WP-0	

15	14	13	12	10	9	8
Reserved		MODCLKPOL1	MMODE1		PMODE1	
R-0		R/WP-0	R/WP-0		R/WP-0	

7	6	5	4	2	1	0
Reserved		MODCLKPOL0	MMODE0		PMODE0	
R-0		R/WP-0	R/WP-0		R/WP-0	

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

Table 27-33. SPI Parallel/Modulo Mode Control Register (SPIPMCTRL) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Reads return 0. Writes have no effect.
29	MODCLKPOL3		Modulo mode SPICLK polarity. This bit determines the polarity of the SPICLK in modulo mode only. If the MMODE3 bits are 000, this bit will be ignored.
		0	Normal SPICLK in all the modes.
		1	Polarity of the SPICLK will be inverted if Modulo mode is selected.
28-26	MMODE3		These bits determine whether the SPI/MibSPI operates with 1, 2, 4, 5, or 6 data lines (if modulo option is supported by the module).
		0	Normal single data line mode - default (PMODE3 should be set to 00)
		1h	2-data line mode (PMODE3 should be set to 00)
		2h	3-data line mode (PMODE3 should be set to 00)
		3h	4-data line mode (PMODE3 should be set to 00)
		4h	5-data line mode (PMODE3 should be set to 00)
		5h	6-data line mode (PMODE3 should be set to 01)
		6h-7h	Reserved

Table 27-33. SPI Parallel/Modulo Mode Control Register (SPIPMCTRL) Field Descriptions (continued)

Bit	Field	Value	Description
25-24	PMODE3		Parallel mode bits determine whether the SPI/MibSPI operates with 1, 2, 4, or 8 data lines.
		0	Normal operation/1-data line (MMODE3 should be set to 000)
		1h	2-data line mode (MMODE3 should be set to 000)
		2h	4-data line mode (MMODE3 should be set to 000)
		3h	8-data line mode (MMODE3 should be set to 000)
23-22	Reserved	0	Reads return 0. Writes have no effect.
21	MODCLKPOL2		Modulo mode SPICLK polarity. This bit determines the polarity of the SPICLK in modulo mode only. If the MMODE2 bits are 000, this bit will be ignored.
		0	Normal SPICLK in all the modes.
		1	Polarity of the SPICLK will be inverted if Modulo mode is selected.
20-18	MMODE2		These bits determine whether the SPI/MibSPI operates with 1, 2, 4, 5, or 6 data lines (if modulo option is supported by the module).
		0	1-data line Mode - default (PMODE2 should be set to 00)
		1h	2-data line Mode (PMODE2 should be set to 00)
		2h	3-data line mode (PMODE2 should be set to 00)
		3h	4-data line mode (PMODE2 should be set to 00)
		4h	5-data line mode (PMODE2 should be set to 00)
		5h	6-data line mode (PMODE2 should be set to 01)
		6h-7h	Reserved
17-16	PMODE2		Parallel mode bits determine whether the SPI/MibSPI operates with 1, 2, 4, or 8 data lines.
		0	Normal operation/1-data line (MMODE2 should be set to 000)
		1h	2-data line mode (MMODE2 should be set to 000)
		2h	4-data line mode (MMODE2 should be set to 000)
		3h	8-data line mode (MMODE2 should be set to 000)
15-14	Reserved	0	Reads return 0. Writes have no effect.
13	MODCLKPOL1		Modulo mode SPICLK polarity. This bit determines the polarity of the SPICLK in modulo mode only. If the MMODE1 bits are 000, this bit will be ignored.
		0	Normal SPICLK in all the modes.
		1	Polarity of the SPICLK will be inverted if Modulo mode is selected.
12-10	MMODE1		These bits determine whether the SPI/MibSPI operates with 1, 2, 4, 5, or 6 data lines (if modulo option is supported by the module).
		0	1-data line mode - default (PMODE1 should be set to 00)
		1h	2-data line mode (PMODE1 should be set to 00)
		2h	3-data line mode (PMODE1 should be set to 00)
		3h	4-data line mode (PMODE1 should be set to 00)
		4h	5-data line mode (PMODE1 should be set to 00)
		5h	6-data line mode (PMODE1 should be set to 01)
		6h-7h	Reserved
9-8	PMODE1		Parallel mode bits determine whether the SPI/MibSPI operates with 1, 2, 4, or 8 data lines.
		0	Normal operation/1-data line (MMODE1 should be set to 000)
		1h	2-data line mode (MMODE1 should be set to 000)
		2h	4-data line mode (MMODE1 should be set to 000)
		3h	8-data line mode (MMODE1 should be set to 000)
7-6	Reserved	0	Reads return 0. Writes have no effect.
5	MODCLKPOL0		Modulo mode SPICLK polarity. This bit determines the polarity of the SPICLK in modulo mode only. If the MMODE0 bits are 000, this bit will be ignored.
		0	Normal SPICLK in all the modes.
		1	Polarity of the SPICLK will be inverted if Modulo mode is selected.

Table 27-33. SPI Parallel/Modulo Mode Control Register (SPIPMCTRL) Field Descriptions (continued)

Bit	Field	Value	Description
4-2	MMODE0		These bits determine whether the SPI/MibSPI operates with 1, 2, 4, 5, or 6 data lines (if modulo option is supported by the module).
		0	1-data line mode - default (PMODE0 should be set to 00)
		1h	2-data line mode (PMODE0 should be set to 00)
		2h	3-data line mode (PMODE0 should be set to 00)
		3h	4-data line mode (PMODE0 should be set to 00)
		4h	5-data line mode (PMODE0 should be set to 00)
		5h	6-data line mode (PMODE0 should be set to 01)
		6h-7h	Reserved
1-0	PMODE0		Parallel mode bits determine whether the SPI/MibSPI operates with 1, 2, 4, or 8 data lines.
		0	Normal operation/1-data line (MMODE0 should be set to 000)
		1h	2-data line mode (MMODE0 should be set to 000)
		2h	4-data line mode (MMODE0 should be set to 000)
		3h	8-data line mode (MMODE0 should be set to 000)

27.9.26 Multi-buffer Mode Enable Register (MIBSPIE)

NOTE: Accessibility of Multi-Buffer RAM

The multi-buffer RAM is not accessible unless the MSPIENA bit set to 1. The only exception to this is in test mode, where, by setting RXRAMACCESS to 1, the multi-buffer RAM can be fully accessed for both read and write.

Figure 27-55. Multi-buffer Mode Enable Register (MIBSPIE) [offset = 70h]

31 — 17	16
Reserved	RXRAM_ACCESS
R-0	R/WP-0

15 — 1	0
Reserved	MSPIENA
R-0	R/WP-0

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

Table 27-34. Multi-buffer Mode Enable Register (MIBSPIE) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Reads return 0. Writes have no effect.
16	RXRAM ACCESS		Receive-RAM access control. During normal operating mode of SPI, the receive data/status portion of multi-buffer RAM is read-only. To enable testing of receive RAM, direct read/write access is enabled by setting this bit.
		0	The RX portion of multi-buffer RAM is not writable by the CPU.
		1	The whole of multi-buffer RAM is fully accessible for read/write by the CPU.
			Note: The RX RAM ACCESS bit remains 0 after reset and it should remain cleared to 0 at all times, except when testing the RAM. SPI should be given a local reset by using the nRESET (SPIGCR0[0]) bit after RAM testing is performed so that the multi-buffer RAM gets re-initialized.
15-1	Reserved	0	Reads return 0. Writes have no effect.
0	MSPIENA		Multi-buffer mode enable. After power-up or reset, MSPIENA remains cleared, which means that the SPI runs in compatibility mode by default. If multi-buffer mode is desired, this register should be configured first after configuring the SPIGCR0 register. If MSPIENA is not set to 1, the multi-buffer mode registers are not writable.
		0	The SPI runs in compatibility mode, that is, in this mode the MibSPI is fully code-compliant to the standard device SPI. No multi-buffered-mode features are supported.
		1	The SPI is configured to run in multi-buffer mode.

NOTE: Accessibility of Registers

Registers from this offset address onwards are not accessible in SPI compatibility mode. They are accessible only in the multi-buffer mode.

27.9.27 TG Interrupt Enable Set Register (TGITENST)

The register TGITENST contains the TG interrupt enable flags for transfer-finished and for transfer-suspended events. Each of the enable bits in the higher half-word and the lower half-word of TGITENST belongs to one TG.

The register map shown in Figure 27-56 and Table 27-35 represents a super-set device with the maximum number of TGs (16) assumed. The actual number of bits available varies per device.

Figure 27-56. TG Interrupt Enable Set Register (TGITENST) [offset = 74h]

31	16
SETINTENRDY[15:0]	
R/W-0	

15	0
SETINTENSUS[15:0]	
R/W-0	

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

Table 27-35. TG Interrupt Enable Set Register (TGITENST) Field Descriptions

Bit	Field	Value	Description
31-16	SETINTENRDY[*n*]		TG interrupt set (enable) when transfer finished. Bit 16 corresponds to TG0, bit 17 corresponds to TG1, and so on.
		0	Read: The TGx-completed interrupt is disabled. This interrupt does not get generated when TGx completes. Write: A write of 0 to this bit has no effect.
		1	Read: The TGx-completed interrupt is enabled. The interrupt gets generated when TGx completes. Write: Enable the TGx-completed interrupt. The interrupt gets generated when TGx completes.
15-0	SETINTENSUS[*n*]		TG interrupt set (enabled) when transfer suspended. Bit 0 corresponds to TG0, bit 1 corresponds to TG1, and so on.
		0	Read: The TGx-completed interrupt is disabled. This interrupt does not get generated when TGx is suspended. Write: A write of 0 to this bit has no effect.
		1	Read: The TGx-completed interrupt is enabled. The interrupt gets generated when TGx is suspended. Write: Enable the TGx-completed interrupt. The interrupt gets generated when TGx is suspended.

27.9.28 TG Interrupt Enable Clear Register (TGITENCR)

The register TGITENCR is used to clear the interrupt enables for the TG-completed interrupt and the TG-suspended interrupts.

The register map shown in Figure 27-57 and Table 27-36 represents a super-set device with the maximum number of TGs (16) assumed. The actual number of bits available varies per device.

Figure 27-57. TG Interrupt Enable Clear Register (TGITENCR) [offset = 78h]

31	16
CLRINTENRDY[15:0]	
R/W-0	

15	0
CLRINTENSUS[15:0]	
R/W-0	

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

Table 27-36. TG Interrupt Enable Clear Register (TGITENCR) Field Descriptions

Bit	Field	Value	Description
31-16	CLRINTENRDY[*n*]		TG interrupt clear (disabled) when transfer finished. Bit 16 corresponds to TG0, bit 17 corresponds to TG1, and so on.
		0	Read: The TGx-completed interrupt is disabled. This interrupt does not get generated when TGx completes. Write: A write of 0 to this bit has no effect.
		1	Read: The TGx-completed interrupt is enabled. The interrupt gets generated when TGx completes. Write: Disable the TGx-completed interrupt. The interrupt does not get generated when TGx completes.
15-0	CLRINTENSUS[*n*]		TG interrupt clear (disabled) when transfer suspended. Bit 0 corresponds to TG0, bit 1 corresponds to TG1, and so on.
		0	Read: The TGx-completed interrupt is disabled. This interrupt does not get generated when TGx is suspended. Write: A write of 0 to this bit has no effect.
		1	Read: The TGx-completed interrupt is enabled. The interrupt gets generated when TGx is suspended. Write: Disable the TGx-completed interrupt. The interrupt does not get generated when TGx is suspended.

27.9.29 Transfer Group Interrupt Level Set Register (TGITLVST)

The register TGITLVST sets the level of interrupts for transfer completed interrupt and for transfer suspended interrupt to level 1.

The register map shown in Figure 27-58 andTable 27-37 represents a super-set device with the maximum number of TGs (16) assumed. The actual number of bits available varies per device.

Figure 27-58. Transfer Group Interrupt Level Set Register (TGITLVST) [offset = 7Ch]

31	16
SETINTLVLRDY[15:0]	
R/W-0	

15	0
SETINTLVLSUS[15:0]	
R/W-0	

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

Table 27-37. Transfer Group Interrupt Level Set Register (TGITLVST) Field Descriptions

Bit	Field	Value	Description
31-16	SETINTLVLRDY[*n*]		Transfer-group completed interrupt level set. Bit 16 corresponds to TG0, bit 17 corresponds to TG1, and so on.
		0	Read: The TGx-completed interrupt is set to INT0. Write: A write of 0 to this bit has no effect.
		1	Read: The TGx-completed interrupt is set to INT1. Write: Set the TGx-completed interrupt to INT1.
15-0	SETINTLVLSUS[*n*]		Transfer-group suspended interrupt level set. Bit 0 corresponds to TG0, bit 1 corresponds to TG1, and so on.
		0	Read: The TGx-suspended interrupt is set to INT0. Write: A write of 0 to this bit has no effect.
		1	Read: The TGx-suspended interrupt is set to INT1. Write: Set the TGx-suspended interrupt to INT1.

27.9.30 Transfer Group Interrupt Level Clear Register (TGITLVCR)

The register TGITLVCR clears the level of interrupts for transfer completed interrupt and for transfer suspended interrupt to level 0.

The register map shown in Figure 27-59 and Table 27-38 represents a super-set device with the maximum number of TGs (16) assumed. The actual number of bits available varies per device.

Figure 27-59. Transfer Group Interrupt Level Clear Register (TGITLVCR) [offset = 80h]

31	16
CLRINTLVLRDY[15:0]	
R/W-0	

15	0
CLRINTLVLSUS[15:0]	
R/W-0	

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

Table 27-38. Transfer Group Interrupt Level Clear Register (TGITLVCR) Field Descriptions

Bit	Field	Value	Description
31-16	CLRINTLVLRDY[*n*]		Transfer-group completed interrupt level clear. Bit 16 corresponds to TG0, bit 17 corresponds to TG1, and so on.
		0	Read: The TGx-completed interrupt is set to INT0. Write: A write of 0 to this bit has no effect.
		1	Read: The TGx-completed interrupt is set to INT1. Write: Clear the TGx-completed interrupt to INT0.
15-0	CLRINTLVLSUS[*n*]		Transfer group suspended interrupt level clear. Bit 0 corresponds to TG0, bit 1 corresponds to TG1, and so on.
		0	Read: The TGx-suspended interrupt is set to INT0. Write: A write of 0 to this bit has no effect.
		1	Read: The TGx-suspended interrupt is set to INT1. Write: Clear the TGx-suspended interrupt to INT0.

27.9.31 Transfer Group Interrupt Flag Register (TGINTFLG)

The TGINTFLG register comprises the transfer group interrupt flags for transfer-completed interrupts (INTFLGRDYx) and for transfer-suspended interrupts (INTFLGSUSx). Each of the interrupt flags in the higher half-word and the lower half-word of TGINTFLG belongs to one TG.

The register map shown in Figure 27-60 and Table 27-39 represents a super-set device with the maximum number of TGs (16) assumed. The actual number of bits available varies per device.

Figure 27-60. Transfer Group Interrupt Flag Register (TGINTFLG) [offset = 84h]

31 … 16
INTFLGRDY[15:0]
R/W1C-0

15 … 0
INTFLGSUS[15:0]
R/W1C-0

LEGEND: R/W = Read/Write; W1C = Write 1 to clear; -*n* = value after reset

Table 27-39. Transfer Group Interrupt Flag Register (TGINTFLG) Field Descriptions

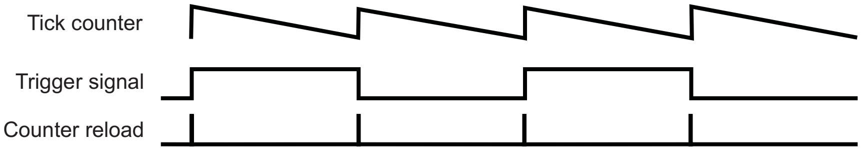
Bit	Field	Value	Description
31-16	INTFLGRDY[*n*]		Transfer-group interrupt flag for a transfer-completed interrupt. Bit 16 corresponds to TG0, bit 17 corresponds to TG1, and so on. **Note: Read Clear Behavior. Reading the interrupt vector registers TGINTVECT0 or TGINTVECT1 automatically clears the interrupt flag bit INTFLGRDYx referenced by the vector number given by INTVECT0/INTVECT1 bits, if the SUSPEND[0:1] bit in the vector registers is 0.**
		0	Read: No transfer-completed interrupt occurred since last clearing of the INTFLGRDYx flag. Write: A write of 0 to this bit has no effect.
		1	Read: A transfer finished interrupt from transfer group x occurred. No matter whether the interrupt is enabled or disabled (INTENRDYx = don't care) or whether the interrupt is mapped to INT0 or INT1, INTFLGRDYx is set right after the transfer from TGx is finished. Write: The corresponding bit flag is cleared.
15-0	INTFLGSUS[*n*]		Transfer-group interrupt flag for a transfer-suspend interrupt. Bit 0 corresponds to TG0, bit 1 corresponds to TG1, and so on. **Note: Read Clear Behavior. Reading the interrupt vector registers TGINTVECT0 or TGINTVECT1 automatically clears the interrupt flag bit INTFLGSUSx referenced by the vector number given by INTVECT0/INTVECT1 bits, if the SUSPEND[0:1] bit in the corresponding vector registers is 1.**
		0	Read: No transfer-suspended interrupt occurred since the last clearing of the INTFLGSUSx flag. Write: A write of 0 to this bit has no effect.
		1	Read: A transfer-suspended interrupt from TGx occurred. No matter whether the interrupt is enabled or disabled (INTENSUSx = don't care) or whether the interrupt is mapped to INT0 or INT1, INTFLGSUSx is set right after the transfer from transfer group x is suspended. Write: The corresponding bit flag is cleared.

27.9.32 Tick Count Register (TICKCNT)

One of the trigger sources for TGs is an internal periodic time trigger. This time trigger is called a tick counter and is basically a down-counter with a preload/reload value. Every time the tick counter detects an underflow it reloads the initial value and toggles the trigger signal provided to the TGs.

The trigger signal, shown in Figure 27-61 as a square wave, illustrates the different trigger event types for the TGs (for example, rising edge, falling edge, and both edges).

Figure 27-61. Tick Counter Operation



This register is shown in Figure 27-62 and described in Table 27-40.

Figure 27-62. Tick Count Register (TICKCNT) [offset = 90h]

31	30	29-28	27-16
TICKENA	RELOAD	CLKCTRL	Reserved
R/W-0	R/S-0	R/W-0	R-0

15-0
TICKVALUE
R/W-0

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

Table 27-40. Tick Count Register (TICKCNT) Field Descriptions

Bit	Field	Value	Description
31	TICKENA		Tick counter enable.
		0	The internal tick counter is disabled. The counter value remains unchanged. **Note: When the tick counter is disabled, the trigger signal is forced low.**
		1	The internal tick counter is enabled and is clocked by the clock source selected by CLKCTRL. When TICKENA goes from 0 to 1, the tick counter is automatically loaded with the contents of TICKVALUE.
30	RELOAD		Pre-load the tick counter. RELOAD is a set-only bit; writing a 1 to it reloads the tick counter with the value stored in TICKVALUE. Reading RELOAD always returns a 0. **Note: When the tick counter is reloaded by the RELOAD bit, the trigger signal is not toggled.**
29-28	CLKCTRL		Tick counter clock source control. CLKCTRL defines the clock source that is used to clock the internal tick counter.
		0	SPICLK of data word format 0 is selected as the clock source of the tick counter.
		1h	SPICLK of data word format 1 is selected as the clock source of the tick counter.
		2h	SPICLK of data word format 2 is selected as the clock source of the tick counter.
		3h	SPICLK of data word format 3 is selected as the clock source of the tick counter.
27-16	Reserved	0	Reads return 0. Writes have no effect.
15-0	TICKVALUE	0-FFFFh	Initial value for the tick counter. TICKVALUE stores the initial value for the tick counter. The tick counter is loaded with the contents of TICKVALUE every time an underflow condition occurs and every time the RELOAD flag is set by the host.

27.9.33 Last TG End Pointer (LTGPEND)

Figure 27-63. Last TG End Pointer (LTGPEND) [offset = 94h]

31 29	28 24	23 16
Reserved	TG_IN_SERVICE	Reserved
R-0	R-0	R-0

15	14 8	7 0
Rsvd	LPEND	Reserved
R-0	R/W-0	R-0

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

Table 27-41. Last TG End Pointer (LTGPEND) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Reads return 0. Writes have no effect.
28-24	TG_IN_SERVICE		The TG number currently being serviced by the sequencer. These bits indicate the current TG that is being serviced. This field can generally be used for code debugging.
		0	No TG is being serviced by the sequencer.
		1h	TG0 is being serviced by the sequencer.
		:	:
		10h	TG15 is being serviced by the sequencer.
			Note: The number of transfer groups varies by device.
		11h-1Fh	Invalid values.
23-15	Reserved	0	Reads return 0. Writes have no effect.
14-8	LPEND	0-7Fh	Last TG end pointer. Usually the TG end address (PEND) is inherently defined by the start value of the starting pointer of the subsequent TG (PSTART). The TG ends one word before the next TG starts (PEND[x] = PSTART[x+1] - 1). For a full configuration of MibSPI, the last TG has no subsequent TG, that is, no end address is defined. Therefore, LPEND has to be programmed to specify explicitly the end address of the last TG. **Note: When using all 8 transfer groups, program the LPEND bits to define the end of the last transfer group. When using less than 8 transfer groups, leave the LPEND bits programmed to point to the end of the buffer and create a dummy transfer group that defines the end of your last intentional transfer group and occupies all the remaining buffer space.**
7-0	Reserved	0	Reads return 0. Writes have no effect.

27.9.34 TGx Control Registers (TGxCTRL)

Each TG can be configured via one dedicated control register. The register description shows one control register (x) that is identical for all TGs. For example, the control register for TG2 is named TG2CTRL and is located at *base address* + *98h* + *4* × *2*. The actual number of available control registers varies by device.

Figure 27-64. MibSPI TG Control Registers (TGxCTRL) [offsets = 98h-D4h]

31	30	29	28	27	24
TGENA	ONESHOT	PRST	TGTD	Reserved	
R/W-0	R/W-0	R/W-0	R-0	R-0	

23	20	19	16
TRIGEVT		TRIGSRC	
R/W-0		R/W-0	

15	14	8	7	6	0
Rsvd	PSTART		Rsvd	PCURRENT	
R-0	R/W-0		R-0	R-0	

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

Table 27-42. TG Control Registers (TGxCTRL) Field Descriptions

Bit	Field	Value	Description
31	TGENA		TGx enable. If the correct event (TRIGEVTx) occurs at the selected source (TRIGSRCx), a group transfer is initiated if no higher-priority TG is in active-transfer mode or if one or more higher-priority TGs are in transfer-suspend mode. Disabling a TG while a transfer is ongoing will finish the ongoing word transfer but not the whole group transfer.
		0	TGx is disabled.
		1	TGx is enabled.
30	ONESHOTx		Single transfer for TGx.
		0	TGx initiates a transfer every time a trigger event occurs and TGENA is set.
		1	A transfer from TGx will be performed only once (one shot) after a valid trigger event at the selected trigger source. After the transfer is finished the TGENAx control bit will be cleared and therefore no additional transfer can be triggered before the host enables the TG again. This one shot mode ensures that after one group transfer the host has enough time to read the received data and to provide new transmit data.
29	PRSTx		TGx pointer reset mode. Configures the way to resolve trigger events during an ongoing transfer. This bit is meaningful only for level-triggered TGs. Edge-triggered TGs cannot be restarted before their completion by another edge. The PRST bit will have no effect on this behavior. **Note: When the PRST bit is set, if the buffer being transferred at the time of a new trigger event is a LOCK, CSHOLD or NOBRK buffer, then only after finishing those transfers, the TG will be restarted. This means that even if the TG is retriggered, the TG will only be restarted after finishing the transfer of the first non-LOCK or non-CSHOLD buffer. In the case of the NOBRK buffer, after completing the ICOUNT number of transfers, the TG will be restarted from its PSTART.** This means that TX control fields such as LOCK and CSHOLD, and DMA control fields such as NOBRK have higher priority over anything else. They have the capability to delay the restart of the TG even if it is retriggered when PRST is 1.
		0	If a trigger event occurs during a transfer from TGx, the event is ignored and is not stored internally. The TGx transfer has priority over additional trigger events.
		1	The TGx pointer (PCURRENTx) will be reset to the start address (PSTARTx) when a valid trigger event occurs at the selected trigger source while a transfer from the same TG is ongoing. Every trigger event resets PCURRENTx no matter whether the concerned TG is in transfer mode or not. The trigger events have priority over the ongoing transfer.
28	TGTDx		TG triggered.
		0	TGx has not been triggered or is no longer waiting for service.
		1	TGx has been triggered and is either currently being serviced or waiting for servicing.

Table 27-42. TG Control Registers (TGxCTRL) Field Descriptions (continued)

Bit	Field	Value	Description	
27-24	Reserved	0	Reads return 0. Writes have no effect.	
23-20	TRIGEVTx		Type of trigger event. A level-triggered TG can be stopped by de-activating the level trigger. However, the following restrictions apply. • Deactivating the level trigger for a TG during a NOBRK transfer does not stop the transfers until all of the ICOUNT number of buffers are transferred for the NOBRK buffer. Once a NOBRK buffer is prefetched, the trigger event loses control over the TG until the NOBRK buffer transfer is completed. • Once the transfer of a buffer with CSHOLD or LOCK bit set starts, deactivating the trigger level does not stop the transfer until the sequencer completes the transfer of the next non-CSHOLD or non-LOCK buffer in the same TG. • Once the last buffer in a TG is pre-fetched, de-activating the trigger level does not stop the transfer group until the last buffer transfer is completed. This means even if the trigger level is deactivated at the beginning of the penultimate (one-before-last) buffer transfer, the sequencer continues with the same TG until it is completed.	
		0	never	Never trigger TGx. This is the default value after reset.
		1h	rising edge	A rising edge (0 to 1) at the selected trigger source (TRIGSRCx) initiates a transfer for TGx
		2h	falling edge	A falling edge (1 to 0) at the selected trigger source (TRIGSRCx) initiates a transfer for TGx
		3h	both edges	Rising and falling edges at the selected trigger source (TRIGSRCx) initiates a transfer for TGx
		4h	Rsvd	Reserved
		5h	high-active	While the selected trigger source (TRIGSRCx) is at a logic-high level (1), the group transfer is continued and at the end of one group, transfer is restarted at the beginning. If the logic level changes to low (0) during an ongoing group transfer, the whole group transfer will be stopped. **Note: If ONESHOTx is set, the transfer is performed only once.**
		6h	low-active	While the selected trigger source (TRIGSRCx) is at a logic-low level (0), the group transfer is continued and at the end of one group, transfer is restarted at the beginning. If the logic level changes to high (1) during an ongoing group transfer, the whole group transfer will be stopped. **Note: If ONESHOTx is set, the transfer is performed only once.**
		7h	always	A repetitive group transfer will be performed. **Note: By setting the TRIGSRC to 0, the TRIGEVT to 7h (ALWAYS), and the ONESHOTx bit to 1, software can trigger this TG. Upon setting the TGENA bit, the TG is immediately triggered.** **Note: If ONESHOTx is set, the transfer is performed only once.**
		8h-Fh	Rsvd	Reserved

Table 27-42. TG Control Registers (TGxCTRL) Field Descriptions (continued)

Bit	Field	Value	Description	
19-16	TRIGSRCx		Trigger source. After reset, the trigger sources of all TGs are disabled.	
		0	Disabled	
		1h	EXT0	External trigger source 0. The actual source varies per device (for example, HET I/O channel, event pin).
		2h	EXT1	External trigger source 1. The actual source varies per device (for example, HET I/O channel, event pin).
		3h	EXT2	External trigger source 2. The actual source varies per device (for example, HET I/O channel, event pin).
		4h	EXT3	External trigger source 3. The actual source varies per device (for example, HET I/O channel, event pin).
		5h	EXT4	External trigger source 4. The actual source varies per device (for example, HET I/O channel, event pin).
		6h	EXT5	External trigger source 5. The actual source varies per device (for example, HET I/O channel, event pin).
		7h	EXT6	External trigger source 6. The actual source varies per device (for example, HET I/O channel, event pin).
		8h	EXT7	External trigger source 7. The actual source varies per device (for example, HET I/O channel, event pin).
		9h	EXT8	External trigger source 8. The actual source varies per device (for example, HET I/O channel, event pin).
		Ah	EXT9	External trigger source 9. The actual source varies per device (for example, HET I/O channel, event pin).
		Bh	EXT10	External trigger source 10. The actual source varies per device (for example, HET I/O channel, event pin).
		Ch	EXT11	External trigger source 11. The actual source varies per device (for example, HET I/O channel, event pin).
		Dh	EXT12	External trigger source 12. The actual source varies per device (for example, HET I/O channel, event pin).
		Eh	EXT13	External trigger source 13. The actual source varies per device (for example, HET I/O channel, event pin).
		Fh	TICK	Internal periodic event trigger. The tick counter can initiate periodic group transfers.
15	Reserved	0	Reads return 0. Writes have no effect.	
14-8	PSTARTx	0-7Fh	TG start address. PSTARTx stores the start address of the corresponding TG. The corresponding end address is inherently defined by the subsequent TG start address minus 1 (PENDx[TGx] = PSTARTx[TGx+1]-1). PSTARTx is copied into PCURRENTx when: • The TG is enabled • The end of the TG is reached during a transfer • A trigger event occurs while PRST is set to 1	
7	Reserved	0	Reads return 0. Writes have no effect.	
6-0	PCURRENTx	0-7Fh	Pointer to current buffer. PCURRENT is read-only. PCURRENTx stores the address (0...127) of the buffer that corresponds to this TG. If the TG switches from active-transfer mode to suspend-to-wait mode, PCURRENTx contains the address of the currently suspended word. After the TG resumes from suspend-to-wait mode, the next buffer will be transferred; that is, no buffer data is transferred because of suspend-to-wait mode.	

NOTE: Register bits vary by device

TG0 has the highest priority and TG15 has the lowest priority. Under the following conditions, a lower-priority TG cannot be interrupted by a higher-priority TG:

1. When there is a CSHOLD or LOCK buffer, until the completion of the next buffer transfer that is a non-CSHOLD or non-LOCK buffer.
2. An entire sequence of words transferred for a NOBRK DMA buffer.
3. Once the last word in a TG is pre-fetched.

27.9.35 DMA Channel Control Register (DMAxCTRL)

Each DMA channel can be configured via one dedicated control register. The register description below shows one exemplary control register that is identical for all DMA channels; for example, the control register for DMA channel 0 is named DMA0CTRL. The MibSPI supports up to 8 bidirectional DMA channels.

The number of bidirectional DMA channels varies by device. The number of DMA channels and hence the number of DMA channel control registers may vary.

Figure 27-65. DMA Channel Control Register (DMAxCTRL) [offset = D8h-F4h]

31	30 24	23 20	19 16
ONESHOT	BUFID	RXDMA_MAP	TXDMA_MAP
R/W-0	R/W-0	R/W-0	R/W-0

15	14	13	12 8
RXDMAENA	TXDMAENA	NOBRK	ICOUNT
R/W-0	R/W-0	R/W-0	R/W-0

7	6	5 0
Reserved	COUNT_BIT17	COUNT
R-0	R-0	R-0

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

Table 27-43. DMA Channel Control Register (DMAxCTRL) Field Descriptions

Bit	Field	Value	Description
31	ONESHOT		Auto-disable of DMA channel after ICOUNT + 1 transfers. **Note: This ONESHOT applies to the DMA channel identified by x and will autodisable based on ICOUNTx.**
		0	The length of the block transfer is fully controlled by the DMA controller. The enable bits RXDMAENAx and TXDMAENAx are not modified by the MibSPI.
		1	ONESHOT allows a block transfer of defined length (ICOUNTx + 1), mainly controlled by the MibSPI and not by the DMA controller. After ICOUNTx + 1 transfers, the enable bits RXDMAENAx and TXDMAENAx are automatically cleared by the MibSPI, hence no more DMA requests are generated. In conjunction with NOBRKx, a burst transfer can be initiated without any other transfer through another buffer.
30-24	BUFIDx	0-7Fh	Buffer utilized for DMA transfer. BUFIDx defines the buffer that is utilized for the DMA transfer. In order to synchronize the transfer with the DMA controller with the NOBRK condition the "suspend to wait until..." modes must be used.
23-20	RXDMA_MAPx	0-Fh	Each MibSPI DMA channel can be linked to two physical DMA Request lines of the DMA controller. One request line for receive data and the other for request line for transmit data. RXDMA_MAPx defines the number of the physical DMA Request line that is connected to the receive path of the MibSPI DMA channel. If RXDMAENAx and TXDMAENAx are both set to 1, then RXDMA_MAPx shall differ from TXDMA_MAPx and shall differ from any other used physical DMA Request line. Otherwise, unexpected interference may occur.
19-16	TXDMA_MAPx	0-Fh	Each MibSPI DMA channel can be linked to two physical DMA Request lines of the DMA controller. One request line for receive data and the other for request line for transmit data. TXDMA_MAPx defines the number of the physical DMA Request line that is connected to the transmit path of the MibSPI DMA channel. If RXDMAENAx and TXDMAENAx are both set to 1, then TXDMA_MAPx shall differ from RXDMA_MAPx and shall differ from any other used physical DMA Request line. Otherwise, unexpected interference may occur.
15	RXDMAENAx		Receive data DMA channel enable.
		0	No DMA request upon new receive data.
		1	The physical DMA channel for the receive path is enabled. The first DMA request pulse is generated after the first transfer from the referenced buffer (BUFIDx) is finished. The buffer should be configured in as "skip until RXEMPTY is set" or "suspend to wait until RXEMPTY is set" in order to ensure synchronization between the DMA controller and the MibSPI sequencer.

Table 27-43. DMA Channel Control Register (DMAxCTRL) Field Descriptions (continued)

Bit	Field	Value	Description
14	TXDMAENAx		Transmit data DMA channel enable.
		0	No DMA request upon new transmit data.
		1	The physical DMA channel for the transmit path is enabled. The first DMA request pulse is generated right after setting TXDMAENAx to load the first transmit data. The buffer should be configured in the as "skip until TXFULL is set" or "suspend to wait until TXFULL is set" in order to ensure synchronization between the DMA controller and the MibSPI sequencer.
13	NOBRKx		Non-interleaved DMA block transfer. This bit is available in master mode only. **Note: Special Conditions during a NOBRK Buffer Transfer. If a NOBRK DMA buffer is currently being serviced by the sequencer, then it is not allowed to be disabled prematurely.** During a NOBRK transfer, the following operations are not allowed: • Clearing the NOBRKx bit to 0 • Clearing the RXDMAENAx to 0 (if it is already 1) • Clearing the TXDMAENAx to 0 (if it is already 1) • Clearing the BUFMODE[2:0] bits in TXRAM to 000 **Note: Any attempts to perform these actions during a NOBRK transfer will produce unpredictable results.**
		0	DMA transfers through the buffer referenced by BUFIDx are interleaved by data transfers from other active buffers or TGs. Every time the sequencer checks the DMA buffer, it performs one transfer and then steps to the next buffer.
		1	NOBRKx ensures that ICOUNTx + 1 data transfers are performed from the buffer referenced by BUFIDx without a data transfer from any other buffer. The sequencer remains at the DMA buffer until ICOUNTx + 1 transfers have been processed. For example, this can be used to generate a burst transfer to one device without disabling the chip select signal in-between (the concerned buffer has to be configured with CSHOLD = 1). Another example would be to have a defined block data transfer in slave mode, synchronous to the master SPI. **Note: Triggering of higher priority TGs or enabling of higher priority DMA channels will not interrupt a NOBRK block transfer.**
12-8	ICOUNTx	0-1Fh	Initial count of DMA transfers. ICOUNTx is used to preset the transfer counter COUNTx. Every time COUNTx hits 0, it is reloaded with ICOUNTx. The real number of transfers equals ICOUNTx plus 1. If ONESHOTx is set, ICOUNTx defines the number of DMA transfers that are performed before the MibSPI automatically disables the DMA channels. If NOBRKx is set, ICOUNTx defines the number of DMA transfers that are performed in one sequence without a transfer from any other buffer. If ONESHOTx and NOBRKx are not set, ICOUNTx should be 0. **Note: See Section 27.9.36 (ICOUNT) and Section 27.9.37 (DMACNTLEN) about how to increase the ICOUNT to a 16-bit value. With this extended capability, MibSPI can transfer a block of up to 65535 (65K) words without interleaving (if NOBRK is used) or without deasserting the chip select between the buffers (if CSHOLD is used).**
7	Reserved	0	Reads return 0. Writes have no effect.
6	COUNT_BIT17x		The 17th bit of the COUNT field of DMAxCOUNT register.
5-0	COUNTx	0-3Fh	Actual number of remaining DMA transfers. This field contains the actual number of DMA transfers that remain, until the DMA channel is disabled, if ONESHOTx is set. **Note: If the TX and RX DMA requests are enabled, the COUNT register will be decremented when the RX has been serviced.**

27.9.36 DMAxCOUNT Register (ICOUNT)

NOTE: These registers are used only if the LARGE COUNT bit in the DMACNTLEN register is set.

The number of bidirectional DMA channels varies by device. The number of DMA channels and hence the number of DMA registers varies by device.

Figure 27-66. DMAxCOUNT Register (ICOUNT) [offset = F8h-114h]

31	16
ICOUNTx	
R/W-0	

15	0
COUNTx	
R-0	

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

Table 27-44. MibSPI DMAxCOUNT Register (ICOUNT) Field Descriptions

Bit	Field	Value	Description
31-16	ICOUNTx	0-FFFFh	Initial number of DMA transfers. ICOUNTx is used to preset the transfer counter COUNTx. Every time COUNTx hits 0, it is reloaded with ICOUNTx. The real number of transfer equals ICOUNTx plus 1. If ONESHOTx is set, ICOUNTx defines the number of DMA transfers that are performed before the MibSPI automatically disables the corresponding DMA channel. If NOBRKx is set, ICOUNTx defines the number of DMA transfers that are performed in one sequence without a transfer from any other buffer.
15-0	COUNTx	0-FFFFh	Actual number of remaining DMA transfers. COUNTx Contains the actual number of DMA transfers that remain, until the DMA channel is disabled, if ONESHOTx is set. Since the real counter value is always ICOUNTx +1, the 17th bit of COUNTx is available on DMACTRLx[6] bit. **Note: Usage Tip for Block Transfer Using a Single DMA Request. It is possible to use the multi-buffer RAM to transfer chunks of data to/from an external SPI. A DMA Controller can be used to handle the data in bursts. Suppose a chunk of 64 bytes of data needs to be transferred and a single DMA request needs to be generated at the end of transferring the 64 bytes. This can be easily achieved by configuring a TG register for the 64 buffer locations and using the DMAxCTRL/DMAxCOUNT registers to configure the last buffer (64th) of the TG as the BUFID and enable RXDMA (NOBRK = 0). At the end of the transfer of the 64th buffer, a DMA request will be generated on the selected DMA request channel. The DMA controller can do a burst read of all 64 bytes from RXRAM and/or then do a burst write to all 64 bytes to the TXRAM for the next chunk.**

27.9.37 DMA Large Count (DMACNTLEN)

Figure 27-67. DMA Large Count Register (DMACNTLEN) [offset = 118h]

31		16
Reserved		
R-0		

15	1	0
Reserved		LARGE_COUNT
R-0		R/WP-0

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

Table 27-45. MibSPI DMA Large Count Register (DMACNTLEN) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	LARGE_COUNT		Select either the 16-bit DMAxCOUNT counters or the smaller counters in DMAxCTRL.
		0	Select the DMAxCTRL counters. Writes to the DMAxCTRL register will modify the ICOUNT value. Reading ICOUNT and COUNT can be done from the DMAxCTRL register. The DMAxCOUNT register should not be used since any write to this register will be overwritten by a subsequent write to the DMAxCTRL register to set the TXDMAENA or RXDMAENA bits.
		1	Select the DMAxCOUNT counters. Writes to the DMAxCTRL register will not modify the ICOUNT value. The ICOUNT value must be written to in the DMAxCOUNT register before the RXDMAENA or TXDMAENA bits are set in the DMAxCTRL register. The DMAxCOUNT register should be used for reading COUNT or ICOUNT.

27.9.38 Multi-buffer RAM Uncorrectable Parity Error Control Register (UERRCTRL)

Figure 27-68. Multi-buffer RAM Uncorrectable Parity Error Control Register (UERRCTRL) [offset = 120h]

31		16
Reserved		
R-0		

15	9	8	7	4	3	0
Reserved		PTESTEN	Reserved		EDEN	
R-0		R/WP-0	R-0		R/WP-5h	

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

Table 27-46. Multi-buffer RAM Uncorrectable Parity Error Control Register (UERRCTRL) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reads return 0. Writes have no effect.
8	PTESTEN		Parity memory test enable. This bit maps the parity bits corresponding to multi-buffer RAM locations into the peripheral RAM frame to make them accessible by the CPU. See Section 27.11 for further details about parity memory testing.
		0	Parity bits are not memory-mapped.
		1	Parity bits are memory-mapped.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	EDEN		Error detection enable. These bits enable parity error detection.
		5h	Parity error detection logic (default) is disabled.
		All Other Values	Parity error detection logic is enabled.
			Note: It is recommended to write a 1010 to enable error detection, to guard against a soft error from disabling parity error detect

27.9.39 Multi-buffer RAM Uncorrectable Parity Error Status Register (UERRSTAT)

Figure 27-69. Multi-buffer RAM Uncorrectable Parity Error Status Register (UERRSTAT) [offset = 124h]

31-16
Reserved
R-0

15-2	1	0
Reserved	EDFLG1	EDFLG0
R-0	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 27-47. Multi-buffer RAM Uncorrectable Parity Error Status Register (UERRSTAT) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reads return 0. Writes have no effect.
1	EDFLG1		Uncorrectable parity error detection flag. This flag indicates if a parity error occurred in the RXRAM. **Note: Reading the UERRADDR1 register clears the EDFLG1 bit.**
		0	Read: No error has occurred. Write: Writing a 0 to this bit has no effect.
		1	Read: An error was detected and the address is captured in the UERRADDR1 register. Write: The bit is cleared to 0.
0	EDFLG0		Uncorrectable parity error detection flag. This flag indicates if a parity error occurred in the TXRAM. **Note: Reading the UERRADDR0 register clears the EDFLG0 bit.**
		0	Read: No error has occurred. Write: Writing a 0 to this bit has no effect.
		1	Read: An error was detected and the address is captured in the UERRADDR0 register. Write: The bit is cleared to 0.

27.9.40 RXRAM Uncorrectable Parity Error Address Register (UERRADDR1)

Figure 27-70. RXRAM Uncorrectable Parity Error Address Register (UERRADDR1) [offset = 128h]

31	16
Reserved	
R-0	

15	10	9	0
Reserved		OVERADDR1	
R-0		R/C-x	

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

Table 27-48. RXRAM Uncorrectable Parity Error Address Register (UERRADDR1) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return 0. Writes have no effect.
9-0	OVERADDR1	200h-3FFh	Uncorrectable parity error address for RXRAM. This register holds the address where a parity error is generated while reading RXRAM. Only the CPU or DMA can read from RXRAM locations. The address captured is byte-aligned. This error address is frozen from being updated until it is read by the CPU. The offset address of RXRAM varies from 200h-3FFh. The register does not clear its contents during or after module-level reset, system-level reset or even power-on reset. A read operation to this register clears its contents to the default value 200h. After a power-on reset the contents will be unpredictable. A read operation can be performed after power-up to keep the register at its default value, if required. However, the contents of this register are meaningful only when EDFLG1 is set to 1. **Note: A read of the UERRADDR1 register will clear EDFLG1 in the UERRSTAT register. However, in emulation mode when the SUSPEND signal is high, a read from the UERRADDR1 register does not clear EDFLG1.**

27.9.41 TXRAM Uncorrectable Parity Error Address Register (UERRADDR0)

Figure 27-71. TXRAM Uncorrectable Parity Error Address Register (UERRADDR0) [offset = 12Ch]

31–16
Reserved
R-0

15–9	8–0
Reserved	UERRADDR0
R-0	R/C-x

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

Table 27-49. TXRAM Uncorrectable Parity Error Address Register (UERRADDR0) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved	0	Reads return 0. Writes have no effect.
8-0	UERRADDR0	0-1FFh	Uncorrectable parity error address for TXRAM. This register holds the address where a parity error is generated while reading from TXRAM. The TXRAM can be read either by CPU or by the MibSPI sequencer logic for transmission. The address captured is byte-aligned. This error address is frozen from being updated until it is read by the CPU. The offset address of TXRAM varies from 0-1FFh. The register does not clear its contents during or after module-level reset, system-level reset, or even power-on reset. A read operation to this register clears its contents to all 0s. After a power-on reset, the contents of this register will be unpredictable. A read operation can be performed after power-up to clear the this register's contents, if required. However, the contents of this register are meaningful only when EDFLG0 is set to 1. **Note: A read from the UERRADDR0 register will clear EDFLG0 in the UERRSTAT register. However, in emulation mode when the SUSPEND signal is high, a read from the UERRADDR0 register does not clear EDFLG0.**

27.9.42 RXRAM Overrun Buffer Address Register (RXOVRN_BUF_ADDR)

In multi-buffer mode, if a particular RXRAM location is written by the MibSPI sequencer logic after the completion of a new transfer when that location already contains valid data, the RX_OVR bit will be set to 1 while the data is being written. The RXOVRN_BUF_ADDR register captures the address of the RXRAM location for which a receiver overrun condition occurred.

Figure 27-72. RXRAM Overrun Buffer Address Register (RXOVRN_BUF_ADDR) [offset = 130h]

31-16
Reserved
R-0

15-10	9-0
Reserved	RXOVRN_BUF_ADDR
R-0	R-200h

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

Table 27-50. RXRAM Overrun Buffer Address Register (RXOVRN_BUF_ADDR) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return 0. Writes have no effect.
9-0	RXOVRN_BUF_ADDR	200h-3FCh	Address in RXRAM at which an overwrite occurred. This address value will show only the offset address of the RAM location in the multi-buffer RAM address space. Refer to the device-specific data sheet for the actual absolute address of RXRAM. This word-aligned address can vary from 200h-3FCh. Contents of this register are valid only when any of the INTVECT0 or INTVECT1 and SPIFLG registers show an RXOVRN error vector while in multi-buffer mode. If there are multiple overrun errors, then this register holds the address of first overrun address until it is read. **Note: Reading this register clears the RXOVRN interrupt flag in the SPIFLG register and the TGINTVECTx.** **Note: Receiver overrun errors in multi-buffer mode can be completely avoided by using the SUSPEND until RXEMPTY feature, which can be programmed into each buffer of any TG. However, using the SUSPEND until RXEMPTY feature will make the sequencer wait until the current RXRAM location is read by the VBUS master before it can start the transfer for the same buffer location again. This may affect the overall throughput of the SPI transfer. By enabling the interrupt on RXOVRN in multi-buffer mode, the user can rely on interrupts to know if a receiver overrun has occurred. The address of the overrun in RXRAM is indicated in this RXOVRN_BUF_ADDR register.**

27.9.43 I/O-Loopback Test Control Register (IOLPBKTSTCR)

This register controls test mode for I/O pins. It also controls whether loop-back should be digital or analog. In addition, it contains control bits to induce error conditions into the module. These are to be used only for module testing.

All of the control/status bits in this register are valid only when the IOLPBKTSTENA field is set to Ah.

Figure 27-73. I/O-Loopback Test Control Register (IOLPBKTSTCR) [offset = 134h]

31 25	24
Reserved	SCS_FAIL_ FLG
R-0	R/W1C-0

23 21	20	19	18	17	16
Reserved	CTRL_ BITERR	CTRL_ DESYNC	CTRL_ PARERR	CTRL_ TIMEOUT	CTRL_ DLENERR
R-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

15 12	11 8
Reserved	IOLPBKTSTENA
R-0	R/WP-0

7 6	5 3	2	1	0
Reserved	ERR_SCS_PIN	CTRL_SCS_ PIN_ERR	LPBKTYPE	RXPENA
R-0	R/WP-0	R/WP-0	R/WP-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; WP = Write in privilege mode only; -*n* = value after reset

Table 27-51. I/O-Loopback Test Control Register (IOLPBKTSTCR) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return 0. Writes have no effect.
24	SCS FAIL FLG		Bit indicating a failure on $\overline{\text{SPICS}}$ pin compare during analog loopback.
		0	Read: No miscompares occurred on any of the eight chip select pins (vs. the internal chip select number CSNR during transfers). Write: Writing a 0 to this bit has no effect.
		1	Read: A comparison between the internal CSNR field and the analog looped-back value of one or more of the $\overline{\text{SPICS}}$ pins failed. A stuck-at fault is detected on one of the $\overline{\text{SPICS}}$ pins. Comparison is done only on the pins that are configured as functional and during transfer operation. Write: This flag bit is cleared.
23-21	Reserved	0	Reads return 0. Writes have no effect.
20	CTRL BITERR		Controls inducing of BITERR during I/O loopback test mode.
		0	Do not interfere with looped-back data.
		1	Induces bit errors by inverting the value of the incoming data during loopback.
19	CTRL DESYNC		Controls inducing of the desync error during I/O loopback test mode.
		0	Do not cause a desync error.
		1	Induce a desync error by forcing the incoming $\overline{\text{SPIENA}}$ pin (if functional) to remain 0 even after the transfer is complete. This forcing will be retained until the kernel reaches the idle state.
18	CTRL PARERR		Controls inducing of the parity errors during I/O loopback test mode.
		0	Do not cause a parity error.
		1	Induce a parity error by inverting the polarity of the parity bit.
17	CTRL TIMEOUT		Controls inducing of the timeout error during I/O loopback test mode.
		0	Do not cause a timeout error.
		1	Induce a timeout error by forcing the incoming $\overline{\text{SPIENA}}$ pin (if functional) to remain 1 when transmission is initiated. The forcing will be retained until the kernel reaches the idle state.

Table 27-51. I/O-Loopback Test Control Register (IOLPBKTSTCR) Field Descriptions (continued)

Bit	Field	Value	Description
16	CTRL DLENERR		Controls inducing of the data length error during I/O loopback test mode.
		0	Do not cause a data-length error.
		1	Induce a data-length error.
			Master mode: The $\overline{\text{SPIENA}}$ pin (if functional) is forced to 1 when the module starts shifting data.
			Slave mode: The incoming $\overline{\text{SPICS}}$ pin (if functional) is forced to 1 when the module starts shifting data.
15-12	Reserved	0	Reads return 0. Writes have no effect.
11-8	IOLPBKSTENA		Module I/O loopback test enable key.
		Ah	Enable I/O loopback test mode.
		All Other Values	Disable I/O loopback test mode.
7-6	Reserved	0	Reads return 0. Writes have no effect.
5-3	ERR SCS PIN		Inject error on chip-select pin number x. The value in this field is decoded as the number of the chip select pin on which to inject an error. During analog loopback, if CTRL SCS PIN ERR bit is set to 1, then the chip select pin selected by this field is forced to the opposite of its value in the CSNR.
		0	Select $\overline{\text{SPICS}}$[0] for injecting error.
		1h	Select $\overline{\text{SPICS}}$[1] for injecting error.
		:	:
		7h	Select $\overline{\text{SPICS}}$[7] for injecting error.
2	CTRL SCS PIN ERR		Enable/disable the injection of an error on the $\overline{\text{SPICS}}$ pins. The individual $\overline{\text{SPICS}}$ pins can be chosen using the ERR SCS PIN field.
		0	Disable the $\overline{\text{SPICS}}$ error-inducing logic.
		1	Enable the $\overline{\text{SPICS}}$ error-inducing logic.
1	LPBK TYPE		Module I/O loopback type (analog/digital). See Figure 27-22 for the different types of loopback modes.
		0	Enable Digital loopback when IOLPBKTSTENA = 1010.
		1	Enable Analog loopback when IOLPBKTSTENA = 1010.
0	RXPENA		Enable analog loopback through the receive pin.
			Note: This bit is valid only when LPBK TYPE = 1, which chooses analog loopback mode.
		0	Analog loopback is through the transmit pin.
		1	Analog loopback is through the receive pin.

27.9.44 SPI Extended Prescale Register 1 (EXTENDED_PRESCALE1 for SPIFMT0 and SPIFMT1)

This register provides an extended Prescale values for SPICLK generation to be able to interface with much slower SPI Slaves. This is an extension of SPIFMT0 and SPIFMT1 registers. For example, EPRESCALE_FMT1[7:0] of EXTENDED_PRESCALE1 and PRESCALE1 of SPIFMT1 register will always reflect the same contents. Similarly, EPRESCALE_FMT0[7:0] and PRESCALE0 of SPIFMT0 reflect the same contents.

Figure 27-74. SPI Extended Prescale Register 1 (EXTENDED_PRESCALE1 for SPIFMT0 and SPIFMT1) [offset = 138h]

31–27	26–16
Reserved	EPRESCALE_FMT1
R-0	R/WP-0

15–11	10–0
Reserved	EPRESCALE_FMT0
R-0	R/WP-0

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

Table 27-52. SPI Extended Prescale Register 1 (EXTENDED_PRESCALE1) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reads return 0. Writes have no effect.
26-16	EPRESCALE_FMT1	0-7FFh	EPRESCALE_FMT1. Extended Prescale value for SPIFMT1. EPRESCALE_FMT1 determines the bit transfer rate of data format 1 if the SPI/MibSPI is the network master. EPRESCALE_FMT1 is use to derive SPICLK from VCLK. If the SPI is configured as slave, EPRESCALE_FMT1 **does not need** to be configured. These EPRESCALE_FMT1[7:0] bits and PRESCALE1 bits of SPIFMT1 register will point to the same physically implemented register. The clock rate for data format 1 can be calculated as: $BR_{Format1}$ = VCLK / (EPRESCALE_FMT1 + 1) Write: This register field should be written if a SPICLK prescaler of more VCLK/256 is required. This field provides a prescaler of up to VCLK/2048 for SPICLK. Writing to this register field will also get reflected in the PRESCALE1 bits of SPIFMT1 register. Read: Reading this field will reflect the PRESCALE value based on the last written register field, that is, EXTENDED_PRESCALE1[26:16] or SPIFMT1[15:8] register. **Note: If Extended Prescaler is required, it should be ensured that EXTENDED_PRESCALE1 register is programmed after SPIFMT1 register is programmed. This is to ensure that the final SPICLK prescale value is controlled by EXTENDED_PRESCALE1 register when a prescale of more 256 is intended on SPICLK. Writing to PRESCALE1 field of SPIFMT1 will automatically clear EPRESCALE_FMT1[10:8] bits to 000 so that the integrity of PRESCALE value is maintained.**
15-11	Reserved	0	Reads return 0. Writes have no effect.

Table 27-52. SPI Extended Prescale Register 1 (EXTENDED_PRESCALE1) Field Descriptions (continued)

Bit	Field	Value	Description
10-0	EPRESCALE_FMT0	0-7FFh	EPRESCALE_FMT0. Extended Prescale value for SPIFMT0. EPRESCALE_FMT0 determines the bit transfer rate of data format 0 if the SPI/MibSPI is the network master. EPRESCALE_FMT0 is use to derive SPICLK from VCLK. If the SPI is configured as slave, EPRESCALE_FMT0 **does not need** to be configured. These EPRESCALE_FMT0[7:0] bits and PRESCALE0 bits of SPIFMT0 register will point to the same physically implemented register. The clock rate for data format 0 can be calculated as: $BR_{Format0}$ = VCLK / (EPRESCALE_FMT0 + 1) Write: This register field should be written if a SPICLK prescaler of more VCLK/256 is required. This field provides a prescaler of up to VCLK/2048 for SPICLK. Writing to this register field will also get reflected in the PRESCALE0 bits of SPIFMT0 register. Read: Reading this field will reflect the PRESCALE value based on the last written register field, that is, EXTENDED_PRESCALE0[10:0] or SPIFMT0[15:8] register. **Note: If Extended Prescaler is required, it should be ensured that EXTENDED_PRESCALE1 register is programmed after SPIFMT0 register is programmed. This is to ensure that the final SPICLK prescale value is controlled by EXTENDED_PRESCALE1 register when a prescale of more 256 is intended on SPICLK. Writing to PRESCALE0 field of SPIFMT0 will automatically clear EPRESCALE_FMT0[10:8] bits to 000 so that the integrity of PRESCALE value is maintained.**

27.9.45 SPI Extended Prescale Register 2 (EXTENDED_PRESCALE2 for SPIFMT2 and SPIFMT3)

This register provides an extended Prescale values for SPICLK generation to be able to interface with much slower SPI Slaves. This is an extension of SPIFMT2 and SPIFMT3 registers. For example, EPRESCALE_FMT3[7:0] of EXTENDED_PRESCALE2 and PRESCALE3 of SPIFMT3 register will always reflect the same contents. Similarly, EPRESCALE_FMT2[7:0] and PRESCALE2 of SPIFMT2 reflect the same contents.

Figure 27-75. SPI Extended Prescale Register 2 (EXTENDED_PRESCALE2 for SPIFMT2 and SPIFMT3) [offset = 13Ch]

31 27	26 16
Reserved	EPRESCALE_FMT3
R-0	R/WP-0

15 11	10 0
Reserved	EPRESCALE_FMT2
R-0	R/WP-0

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

Table 27-53. SPI Extended Prescale Register 2 (EXTENDED_PRESCALE2) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reads return 0. Writes have no effect.
26-16	EPRESCALE_FMT3	0-7FFh	EPRESCALE_FMT3. Extended Prescale value for SPIFMT3. EPRESCALE_FMT3 determines the bit transfer rate of data format 3 if the SPI/MibSPI is the network master. EPRESCALE_FMT3 is use to derive SPICLK from VCLK. If the SPI is configured as slave, EPRESCALE_FMT3 **does not need** to be configured. These EPRESCALE_FMT3[7:0] bits and PRESCALE3 bits of SPIFMT3 register will point to the same physically implemented register. The clock rate for data format 1 can be calculated as: $BR_{Format3}$ = VCLK / (EPRESCALE_FMT3 + 1) Write: This register field should be written if a SPICLK prescaler of more VCLK/256 is required. This field provides a prescaler of up to VCLK/2048 for SPICLK. Writing to this register field will also get reflected in the PRESCALE3 bits of SPIFMT3 register. Read: Reading this field will reflect the PRESCALE value based on the last written register field, that is, EXTENDED_PRESCALE3[26:16] or SPIFMT3[15:8] register. **Note: If Extended Prescaler is required, it should be ensured that EXTENDED_PRESCALE2 register is programmed after SPIFMT3 register is programmed. This is to ensure that the final SPICLK prescale value is controlled by EXTENDED_PRESCALE2 register when a prescale of more 256 is intended on SPICLK. Writing to PRESCALE3 field of SPIFMT3 will automatically clear EPRESCALE_FMT3[10:8] bits to 000 so that the integrity of PRESCALE value is maintained.**
15-11	Reserved	0	Reads return 0. Writes have no effect.

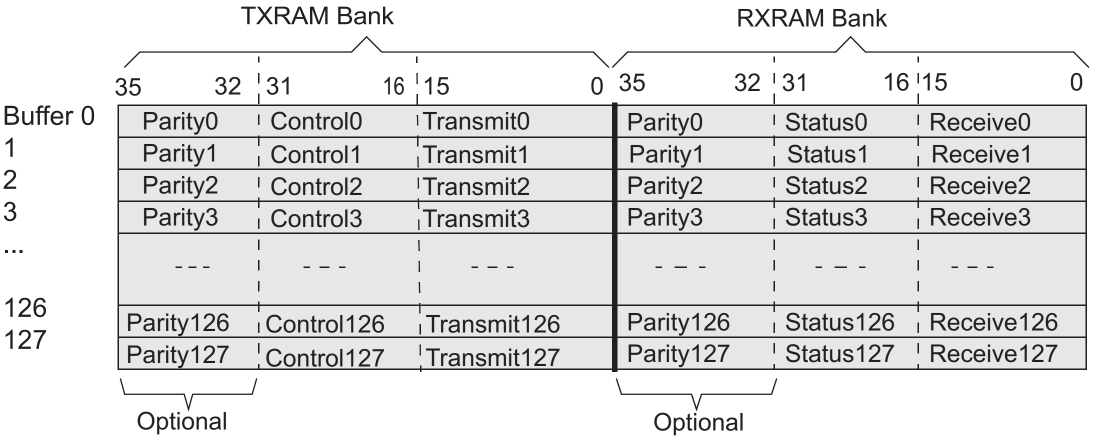
Table 27-53. SPI Extended Prescale Register 2 (EXTENDED_PRESCALE2) Field Descriptions (continued)

Bit	Field	Value	Description
10-0	EPRESCALE_FMT2	0-7FFh	EPRESCALE_FMT2. Extended Prescale value for SPIFMT2. EPRESCALE_FMT2 determines the bit transfer rate of data format 2 if the SPI/MibSPI is the network master. EPRESCALE_FMT2 is use to derive SPICLK from VCLK. If the SPI is configured as slave, EPRESCALE_FMT2 **does not need** to be configured. These EPRESCALE_FMT2[7:0] bits and PRESCALE2 bits of SPIFMT2 register will point to the same physically implemented register. The clock rate for data format 0 can be calculated as: $BR_{Format2}$ = VCLK / (EPRESCALE_FMT2 + 1) Write: This register field should be written if a SPICLK prescaler of more VCLK/256 is required. This field provides a prescaler of up to VCLK/2048 for SPICLK. Writing to this register field will also get reflected in the PRESCALE2 bits of SPIFMT2 register. Read: Reading this field will reflect the PRESCALE value based on the last written register field, that is, EXTENDED_PRESCALE2[10:0] or SPIFMT2[15:8] register. **Note: If Extended Prescaler is required, it should be ensured that EXTENDED_PRESCALE2 register is programmed after SPIFMT2 register is programmed. This is to ensure that the final SPICLK prescale value is controlled by EXTENDED_PRESCALE2 register when a prescale of more 256 is intended on SPICLK. Writing to PRESCALE2 field of SPIFMT2 will automatically clear EPRESCALE_FMT2[10:8] bits to 000 so that the integrity of PRESCALE value is maintained.**

27.10 Multi-Buffer RAM

The multi-buffer RAM is used for holding transit and received data, control and status information. The multi-buffer RAM contains two banks of up to 128 32-bit words for a maximum configuration. One bank (TXRAM) contains entries for transmit data (replicating the SPIDAT1 register). The other bank (RXRAM) contains received data (replicating the SPIBUF register). The buffers can be partitioned into multiple TGs, each containing a programmable number of buffers. Each of the buffers can be subdivided into 16-bit transmit field, 16-bit receive field, 16-bit control field, and 16-bit status field, as displayed in Figure 27-76. A 4-bit parity field per word is also included in each bank of RAM.

Figure 27-76. Multi-Buffer RAM Configuration



	35 32	31 16	15 0	35 32	31 16	15 0
Buffer 0	Parity0	Control0	Transmit0	Parity0	Status0	Receive0
1	Parity1	Control1	Transmit1	Parity1	Status1	Receive1
2	Parity2	Control2	Transmit2	Parity2	Status2	Receive2
3	Parity3	Control3	Transmit3	Parity3	Status3	Receive3
...	- - -	- - -	- - -	- - -	- - -	- - -
126	Parity126	Control126	Transmit126	Parity126	Status126	Receive126
127	Parity127	Control127	Transmit127	Parity127	Status127	Receive127

All fields can be read and written with 8-bit, 16-bit, or 32-bit accesses.

The transmit fields can be written and read in the address range 000h to 1FFh. The transmit words contain data and control fields.

The receive RAM fields are read-only and can be accessed through the address range 200h to 3FCh. The receive words contain data and status fields.

The chip select number (CSNR) bit field of the control field for a given word is mirrored into the corresponding receive-buffer status field after transmission.

The Parity is automatically calculated and copied to Parity location

NOTE: Please refer to the specific device datasheet for the actual number of transmit and receive buffers.

Write to unimplemented buffer is overwriting the corresponding implemented buffer. In MIBSPI, if the RAM SIZE specified is 32 buffers, write to 33rd buffer overwrites 1st buffer, write to 34th buffer overwrites 2st buffer and so on.

27.10.1 Multi-Buffer RAM Auto Initialization

When the MIBSPI is out of reset mode, auto initialization of multi-buffer RAM starts. The application code must check for BUFINITACTIVE bit to be 0 (Multi-buffer RAM initialization is complete) before configuring multi-buffer RAM.

Besides the default auto initialization after reset, the auto-initialization sequence can also be done by:

1. Enable the global hardware memory initialization key by programming a value of 1010b to the bits [3:0] of the MINITGCR register of the System module.
2. Set the control bit for the multi-buffer RAM in the MSINENA System module register. This bit is device-specific for each memory that support auto-initialization. Please refer to the device datasheet to identify the control bit for the multi-buffer RAM. This starts the initialization process. The BUFINITACTIVE bit will get set to reflect that the initialization is ongoing.
3. When the memory initialization is completed, the corresponding status bit in the MINISTAT register will be set. Also, the BUFINITACTIVE bit will get cleared.
4. Disable the global hardware memory initialization key by programming a value of 0101 to the bits [3:0] of the MINITGCR register of the System module.

Please refer to the *Architecture* chapter for more details on the memory auto-initialization process.

NOTE: During Auto Initialization process, all the Multi-buffer mode registers (except MIBSPIE) will be reset to their default values. So, it should be ensured that Auto Initialization is completed before configuring the Multi-buffer mode registers.

27.10.2 Multi-Buffer RAM Register Summary

This section describes the multi-buffer RAM control and transmit-data fields of each word of TXRAM, and the status and receive-data fields of each word of RXRAM. The base address for multi-buffer RAM is FF0E 0000h for MibSPI1 RAM, FF0C 000h for MibSPI3 RAM, and FF0A 0000h for MibSPI5 RAM.

Table 27-54. Multi-Buffer RAM Register Summary

Offset	Acronym	Register Description	Section
Base + 0h-1FFh	TXRAM	Multi-Buffer RAM Transmit Data Register	Section 27.10.3
Base + 200h-3FFh	RXRAM	Multi-Buffer RAM Receive Buffer Register	Section 27.10.4

27.10.3 Multi-Buffer RAM Transmit Data Register (TXRAM)

Each word of TXRAM is a transmit-buffer register.

NOTE: Writing to only the control fields, bits 28 through 16, does not initiate any SPI transfer in master mode. This feature can be used to set up SPICLK phase or polarity before actually starting the transfer by only updating the DFSEL bit field to select the required phase and polarity combination.

Figure 27-77. Multi-Buffer RAM Transmit Data Register (TXRAM) [offset = RAM Base + 0h-1FFh]

31 29	28	27	26	25 24	23 16
BUFMODE	CSHOLD	LOCK	WDEL	DFSEL	CSNR
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15 0
TXDATA
R/W-0

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

Table 27-55. Multi-Buffer RAM Transmit Data Register (TXRAM) Field Descriptions

Bit	Field	Value	Description
31-29	BUFMODE		Specify conditions that are recognized by the sequencer to initiate transfers of each buffer word. When one of the "skip" modes is selected, the sequencer checks the buffer status every time it reads from this buffer. If the current buffer status (TXFULL, RXEMPTY) does not match, the buffer is skipped without a data transfer. When one of the "suspend" modes is selected, the sequencer checks the buffer status when it reads from this buffer. If TXFULL and/or RXEMPTY do not match, the sequencer waits until a match occurs. No data transfer is initiated until the status condition of this buffer changes.
		0	**disabled.** The buffer is disabled.
		1h	**skip single-transfer mode.** Skip this buffer until the corresponding TXFULL flag is set (new transmit data is available).
		2h	**skip overwrite-protect mode.** Skip this buffer until the corresponding RXEMPTY flag is set (new receive data can be stored in RXDATA without data loss).
		3h	**skip single-transfer overwrite-protect mode**. Skip this buffer until both of the corresponding TXFULL and RXEMPTY flags are set. (new transmit data available and previous data received by the host).
		4h	**continuous mode.** Initiate a transfer each time the sequencer checks this buffer. Data words are retransmitted if the buffer has not been updated. Receive data is overwritten, even if it has not been read.
		5h	**suspend single-transfer mode.** Suspend-to-wait until the corresponding TXFULL flag is set (the sequencer stops at the current buffer until new transmit data is written in the TXDATA field).
		6h	**suspend overwrite-protect mode.** Suspend-to-wait until the corresponding RXEMPTY flag is set (the sequencer stops at the current buffer until the previously-received data is read by the host.
		7h	**suspend single-transfer overwrite-protect mode.** Suspend-to-wait until the corresponding TXFULL and RXEMPTY flags are set (the sequencer stops at the current buffer until new transmit data is written into the TXDATA field and the previously-received data is read by the host).
28	CSHOLD		Chip select hold mode. The CSHOLD bit is supported in master mode only, it is ignored in slave mode. 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. If two consecutive transfers are dedicated to the same chip select this chip select signal will be deactivated for at least 2VCLK cycles before it is activated again.
		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 number equals the previous one, the active chip select signal is extended until the end of transfer with CSHOLD cleared, or until the chip-select number changes.

Table 27-55. Multi-Buffer RAM Transmit Data Register (TXRAM) Field Descriptions (continued)

Bit	Field	Value	Description
27	LOCK		Lock two consecutive buffer words. Do not allow interruption by TGs with higher priority.
		0	Any higher-priority TG can begin at the end of the current transaction.
		1	A higher-priority TG cannot occur until after the next unlocked buffer word is transferred.
26	WDEL		Enable the delay counter at the end of the current transaction. **Note: The WDEL bit is supported in master mode only. In slave mode, this bit is ignored.**
		0	No delay will be inserted. However, $\overline{\text{SPICS}}$ pins will still be de-activated for at least for 2VCLK cycles if CSHOLD = 0. **Note: The duration for which the $\overline{\text{SPICS}}$ pin remains deactivated also depends upon the time taken to supply a new word after completing the shift operation (in compatibility mode). If TXBUF is already full, then the $\overline{\text{SPICS}}$ pin will be deasserted for at least two VCLK cycles (if WDEL = 0).**
		1	After a transaction, WDELAY of the corresponding data format will be loaded into the delay counter. No transaction will be performed until the WDELAY counter overflows. The $\overline{\text{SPICS}}$ pins will be de-activated for at least (WDELAY + 2) × VCLK_Period duration.
25-24	DFSEL		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.
23-16	CSNR	0-FFh	Chip select (CS) number. CSNR defines the chip select pins that will be activated during the data transfer. CSNR is a bit-mask that controls all chip select pins. See Table 27-56. **Note: If your MibSPI has less than 8 chip select pins, all unused upper bits will be 0. For example, MiBSPI3 has 6 chip select pins, if you write FFh to CSNR, the actual number stored in CSNR is 3Fh.**
15-0	TXDATA	0-7FFFh	Transfer data. When written, these bits are copied to the shift register if it is empty. If the shift register is not empty, then they are held in TXBUF. SPIEN must be set to 1 before this register can be written to. Writing a 0 to SPIEN forces the lower 16 bits of TXDATA to 0. A write to this register (or to the TXDATA field only) drives the contents of the CSNR field on the $\overline{\text{SPICS}}$ pins, if the pins are configured as functional pins (automatic chip select, see Section 27.2). When this register is read, the contents of TXBUF, which holds the latest data written, will be returned. **Note: Regardless of the character length, the transmit data should be right-justified before writing to the SPIDAT1 register.**

Table 27-56. Chip Select Number Active

CSNR Value	Chip Select Active:						CSNR Value	Chip Select Active:					
	CS[5][1]	CS[4][1]	CS[3][1]	CS[2][1]	CS[1][1]	CS[0]		CS[5][1]	CS[4][1]	CS[3][1]	CS[2][1]	CS[1][1]	CS[0]
0h	No chip select pin is active.						20h	x					
1h						x	21h	x					x
2h					x		22h	x				x	
3h					x	x	23h	x				x	x
4h				x			24h	x			x		
5h				x		x	25h	x			x		x
6h				x	x		26h	x			x	x	
7h				x	x	x	27h	x			x	x	x
8h			x				28h	x		x			
9h			x			x	29h	x		x			x
Ah			x		x		2Ah	x		x		x	
Bh			x		x	x	2Bh	x		x		x	x
Ch			x	x			2Ch	x		x	x		
Dh			x	x		x	2Dh	x		x	x		x
Eh			x	x	x		2Eh	x		x	x	x	
Fh			x	x	x	x	2Fh	x		x	x	x	x
10h		x					30h	x	x				
11h		x				x	31h	x	x				x
12h		x			x		32h	x	x			x	
13h		x			x	x	33h	x	x			x	x
14h		x		x			34h	x	x		x		
15h		x		x		x	35h	x	x		x		x
16h		x		x	x		36h	x	x		x	x	
17h		x		x	x	x	37h	x	x		x	x	x
18h		x	x				38h	x	x	x			
19h		x	x			x	39h	x	x	x			x
1Ah		x	x		x		3Ah	x	x	x		x	
1Bh		x	x		x	x	3Bh	x	x	x		x	x
1Ch		x	x	x			3Ch	x	x	x	x		
1Dh		x	x	x		x	3Dh	x	x	x	x		x
1Eh		x	x	x	x		3Eh	x	x	x	x	x	
1Fh		x	x	x	x	x	3Fh	x	x	x	x	x	x

[1] If your MibSPI does not have this chip select pin, this bit is 0.

27.10.4 Multi-Buffer RAM Receive Buffer Register (RXRAM)

Each word of RXRAM is a receive-buffer register.

Figure 27-78. Multi-Buffer RAM Receive Buffer Register (RXRAM) [offset = RAM Base + 200h-3FFh]

31	30	29	28	27	26	25	24
RXEMPTY	RXOVR	TXFULL	BITERR	DESYNC	PARITYERR	TIMEOUT	DLENERR
RS-1	RC-0	R-0	RC-0	RC-0	RC-0	RC-0	RC-0

23	16
LCSNR	
R-0	

15	0
RXDATA	
R/W-0	

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

Table 27-57. Multi-Buffer Receive Buffer Register (RXRAM) Field Descriptions

Bit	Field	Value	Description
31	RXEMPTY		Receive data buffer empty. When the host reads the RXDATA field or the entire RXRAM register, it automatically sets the RXEMPTY flag. When a data transfer is completed, the received data is copied into RXDATA, and the RXEMPTY flag is cleared.
		0	New data has been received and copied into RXDATA.
		1	No data has been received since the last read of RXDATA.
			This flag gets set to 1 under the following conditions: • Reading the RXDATA field of the RXRAM register • Writing a 1 to clear the RXINTFLG bit in the SPI Flag Register (SPIFLG) Write-clearing the RXINTFLG bit before reading RXDATA indicates the received data is being ignored. Conversely, RXINTFLG can be cleared by reading the RXDATA field of RXRAM (or the entire register).
30	RXOVR		Receive data buffer overrun. When a data transfer is completed and the received data is copied into RXBUF while it is already full, RXOVR is set. Overruns always occur to RXBUF, not to RXRAM; the contents of RXRAM are overwritten only after it is read by the Peripheral (VBUSP) master (CPU, DMA, or other host processor). If enabled, the RXOVRN interrupt is generated when RXBUF is overwritten, and reading either SPI Flag Register (SPIFLG) or SPIVECTx shows the RXOVRN condition. Two read operations from the RXRAM register are required to reach the overwritten buffer word (one to read RXRAM, which then transfers RXDATA into RXRAM for the second read). **Note: This flag is cleared to 0 when the RXDATA field of the RXRAM register is read.** **Note: A special condition under which RXOVR flag gets set. If both RXRAM and RXBUF are already full and while another buffer receive is underway, if any errors such as TIMEOUT, BITERR, and DLEN_ERR occur, then RXOVR in RXBUF and SPI Flag Register (SPIFLG) registers will be set to indicate that the status flags are getting overwritten by the new transfer. This overrun should be treated like a normal receive overrun.**
		0	No receive data overrun condition occurred since last read of the data field.
		1	A receive data overrun condition occurred since last read of the data field.
29	TXFULL		Transmit data buffer full. This flag is a read-only flag. Writing into the SPIDAT0 or SPIDAT1 field while the TX shift register is full will automatically set the TXFULL flag. Once the word is copied to the shift register, the TXFULL flag will be cleared. Writing to SPIDAT0 or SPIDAT1 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.

Table 27-57. Multi-Buffer Receive Buffer Register (RXRAM) Field Descriptions (continued)

Bit	Field	Value	Description
28	BITERR		Bit error. There was a mismatch of internal transmit data and transmitted data.
			Note: This flag is cleared to 0 when the RXDATA field of the RXRAM register is read.
		0	No bit error occurred.
		1	A bit error occurred. The SPI samples the signal of the transmit pins (master: SIMOx, slave: SOMIx) at the receive point (one-half clock cycle after the transmit point). If the sampled value differs from the transmitted value, a bit error is detected and the BITERR flag is set. Possible reasons for a bit error include noise, an excessively high bit rate, capacitive load, or another master/slave trying to transmit at the same time.
27	DESYNC		Desynchronization of slave device. This bit is valid in master mode only.
			The master monitors the ENA signal coming from the slave device and sets the DESYNC flag if 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: In the Compatibility Mode MibSPI, 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 is generated when the buffer transfer is completed. But desynchronization is detected after the buffer transfer is completed. So, if the VBUS master reads the received data quickly when an RXINT is detected, then the status flag may not reflect the correct desync condition. In multi-buffer mode, the desync flag is always guaranteed to be for the current buffer.
			Note: This flag is cleared to 0 when the RXDATA field of the RXRAM register is read.
		0	No slave desynchronization is detected.
		1	A slave device is desynchronized.
26	PARITYERR		Parity error. The calculated parity differs from the received parity bit.
			If the parity generator is enabled (selected individually for each buffer) 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 PARITYERR flag is set.
			Note: This flag is cleared to 0 when the RXDATA field of the RXRAM register is read.
		0	No parity error is detected.
		1	A parity error occurred.
25	TIMEOUT		Time-out because of non-activation of $\overline{\text{SPIENA}}$ pin.
			The SPI generates a time-out when the slave does not respond in time by activating the 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. In addition, the TIMEOUT flag in the status field of the corresponding buffer and in the SPI Flag Register (SPIFLG) is set.
			Note: This bit is valid only in master mode.
			Note: This flag is cleared to 0 when the RXDATA field of the RXRAM register is read.
		0	No $\overline{\text{SPIENA}}$ pin time-out occurred.
		1	An $\overline{\text{SPIENA}}$ signal time-out occurred.
24	DLENERR		Data length error flag.
			Note: This flag is cleared to 0 when the RXDATA field of the RXRAM register is read.
		0	No data-length error occurred.
		1	A data length error occurred.
23-16	LCSNR	0-FFh	Last chip select number. LCSNR in the status field is a copy of CSNR in the corresponding control field. It contains the chip select number that was activated during the last word transfer.
15-0	RXDATA	0-FFFFh	SPI receive data. This is the received word, transferred from the receive shift-register at the end of a transfer. Regardless of the programmed character length and the direction of shifting, the received data is stored right-justified in the register.

27.11 Parity Memory

The parity portion of multi-buffer RAM is not accessible by the CPU during normal operating modes. However, each read or write operation to the control/data/status portion of the multi-buffer RAM causes reads/writes to the parity portion as well.

- Each write to the multi-buffer RAM (either from the Peripheral interface or by the MibSPI itself) causes a write operation to the parity portion of RAM simultaneously to update the equivalent parity bits.
- Each read operation from the multi-buffer RAM (either from the Peripheral interface or by the MibSPI itself) causes a read operation from the parity portion of the RAM for parity comparison purpose.
- Reads/Writes to multi-buffer RAM can either be caused by any CPU/DMA accesses or by the sequencer logic of MibSPI itself.
- Incase of Parity error ESM module is notified to generate MIBSPI Parity ESM interrupt. User can check the error status and address location captured in the UERRSTAT and UERRADDRx registers respectively.

For testing the parity portion of the multi-buffer RAM, which is a 4-bit field per word address (1 bit per byte), a separate parity memory test mode is available. Parity memory test mode can be enabled and disabled by the PTESTEN bit in the UERRCTRL register.

During the parity test mode, the parity locations are addressable at the address between RAM_BASE_ADDR + 0x400h and RAM_BASE_ADDR + 0x7FFh. Each location corresponds, sequentially, to each TXRAM word, then to each RXRAM word. See Figure 27-79 for a diagram of the memory map of parity memory during normal operating mode and during parity test mode.

During parity test mode, after writing the data/control portion of the RAM, the parity locations can be written with incorrect parity bits to intentionally cause parity errors.

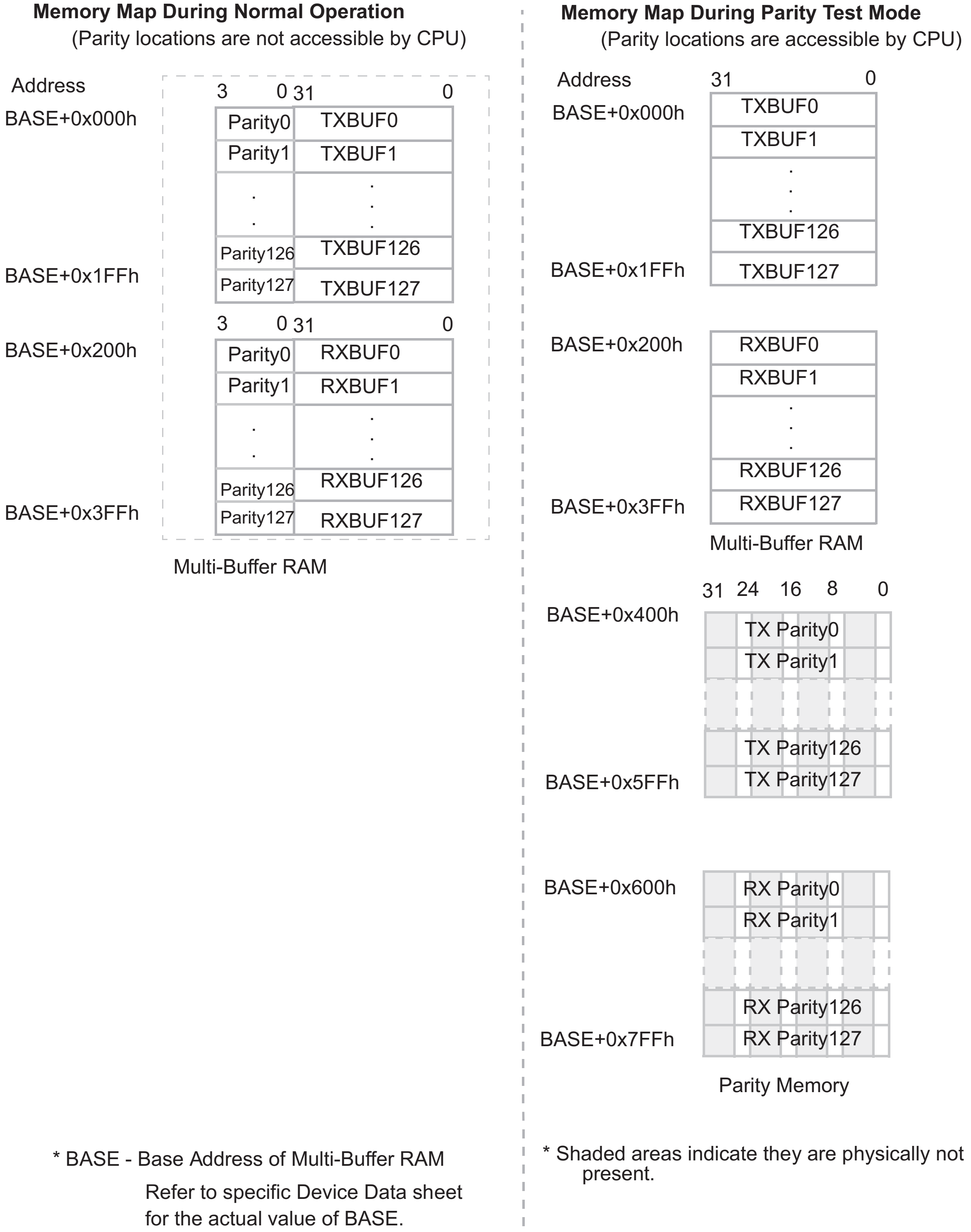
See the device-specific data sheet to get the actual base address of the multi-buffer RAM.

NOTE: The RX_RAM_ACCESS bit can also be set to 1 during the parity test mode to be enable writes to RXRAM locations. Both parity RAM testing and RXRAM testing can be done together.

There are 4 bits of parity corresponding to each of the 32-bit multi-buffer locations. Individual bits in the parity memory are byte-addressable in parity test mode. See the example in Figure 27-80 for further details.

NOTE: Polarity of the parity (odd/even) varies by device. In some devices, a control register in the system module can be used to select odd or even parity.

Figure 27-79. Memory Map for Parity Locations During Normal and Test Mode

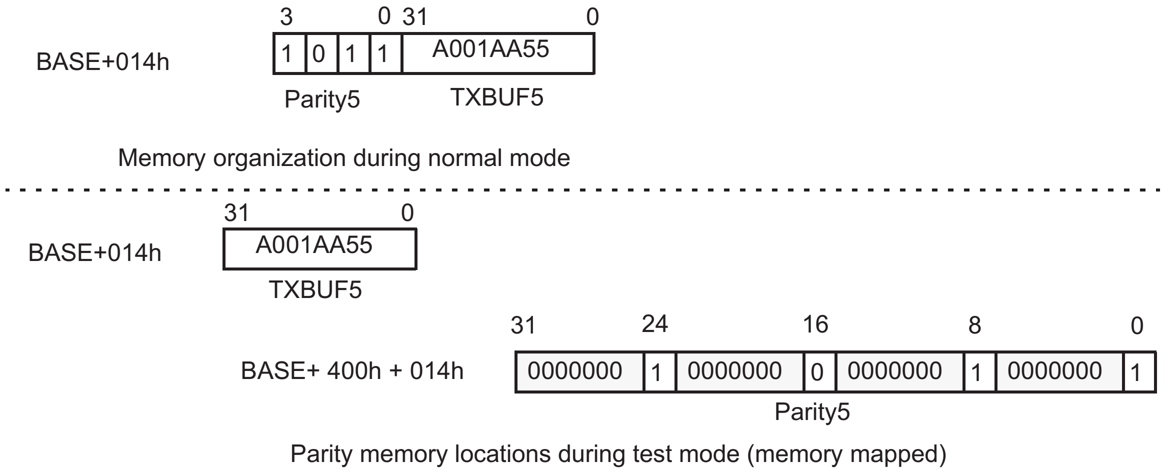


27.11.1 Example of Parity Memory Organization

Suppose TXBUF5 (6th location in TXRAM) in the multi-buffer RAM is written with a value of A001_AA55. If the polarity of the parity is set to odd, the corresponding parity location parity5 will get updated with equivalent parity of 1011 in its field.

During parity-memory test mode, these bits can be individually byte addressed. The return data will be a byte adjusted with actual parity bit in the LSB of the byte. If a word is read from the word-boundary address of parity locations, then each bit of the 4-bit parity is byte-adjusted and a 32-bit word is returned. 0s will be padded into the parity bits to get each byte. See Figure 27-80 for a diagram.

Figure 27-80. Example of Memory-Mapped Parity Locations During Test Mode



1 Shaded areas indicate reads return 0, writes have no effect. These registers are not physically present.

NOTE: Read Access to Parity Memory Locations

Parity memory locations can be read even without entering into parity memory test mode. Their address remains as in memory test mode. It is only to enter parity-memory test mode to enable write access to the parity memory locations.

27.12 MibSPI Pin Timing Parameters

The pin timings of SPI can be classified based on its mode of operation. In each mode, different configurations like Phase and Polarity affect the pin timings.

The pin directions are based on the mode of operation.

Master mode SPI:

- SPICLK (SPI Clock) - Output
- SPISIMO (SPI Slave In Master Out) - Output
- $\overline{\text{SPICS}}$ (SPI Slave Chip Selects) - Output
- SPISOMI (SPI Slave Out Master In) - Input
- $\overline{\text{SPIENA}}$ (SPI slave ready Enable) - Input

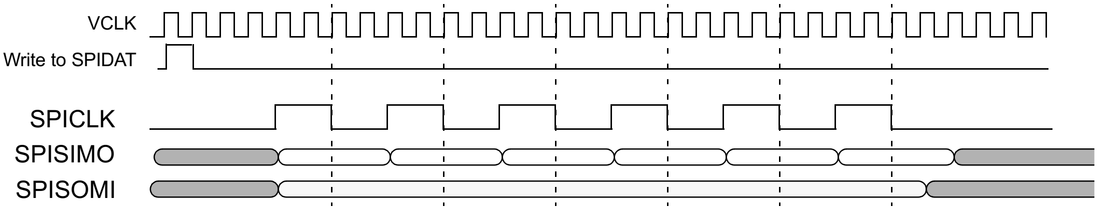
Slave mode SPI:

- SPICLK - Input
- SPISIMO - Input
- $\overline{\text{SPICS}}$ - Input
- SPISOMI - Output
- $\overline{\text{SPIENA}}$ - Output

NOTE: All the timing diagrams given below are with Phase = 0 and Polarity = 0. unless explicitly stated otherwise.

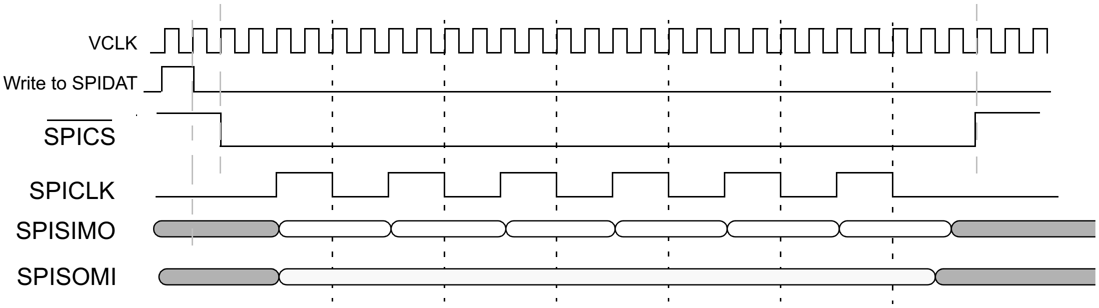
27.12.1 Master Mode Timings for SPI/MibSPI

Figure 27-81. SPI/MibSPI Pins During Master Mode 3-pin Configuration



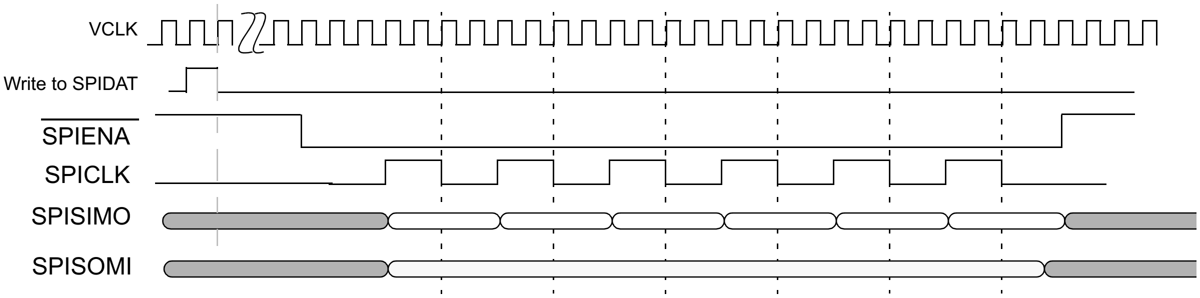
* Dotted vertical lines indicate the receive edges

Figure 27-82. SPI/MibSPI Pins During Master Mode 4-pin with $\overline{\text{SPICS}}$ Configuation



* Dotted vertical lines indicate the receive edges

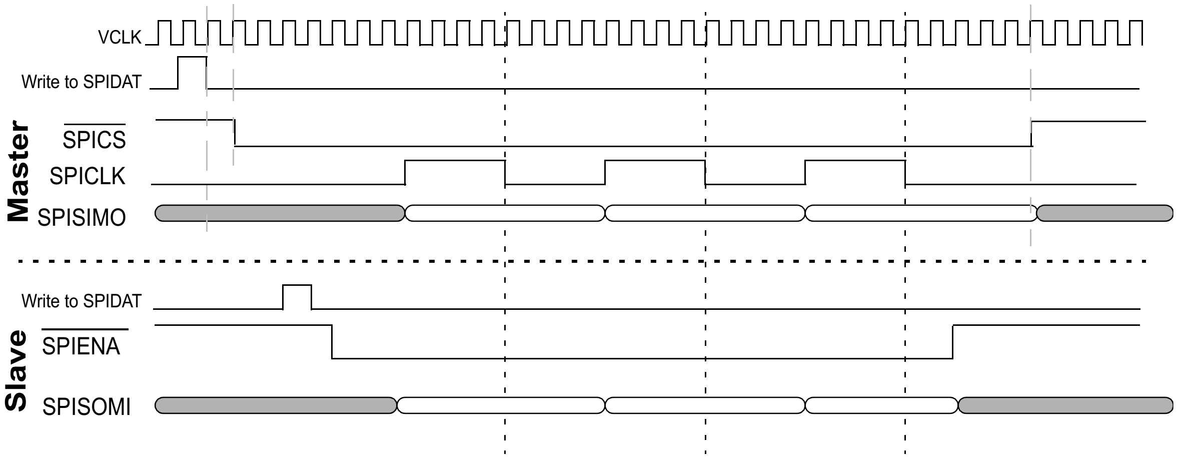
Figure 27-83. SPI/MibSPI Pins During Master Mode 4-pin with SPIENA Configuration



* De-activation of SPIENA pin is controlled by the Slave.

* Dotted vertical lines indicate the receive edges

Figure 27-84. SPI/MibSPI Pins During Master/Slave Mode with 5-pin Configuration

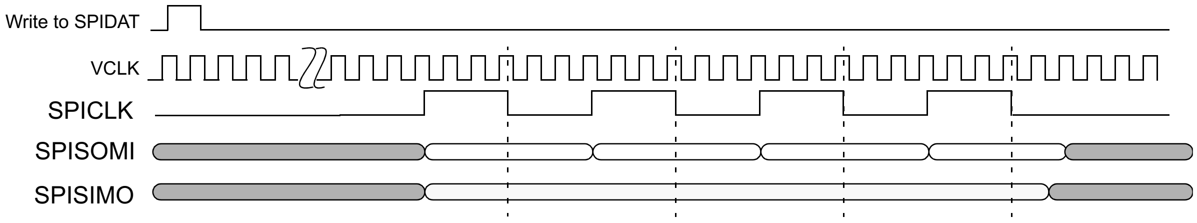


* Dotted vertical lines indicate the receive edges for the Master

* ENABLE_HIGHZ is cleared to 0 in Slave SPI

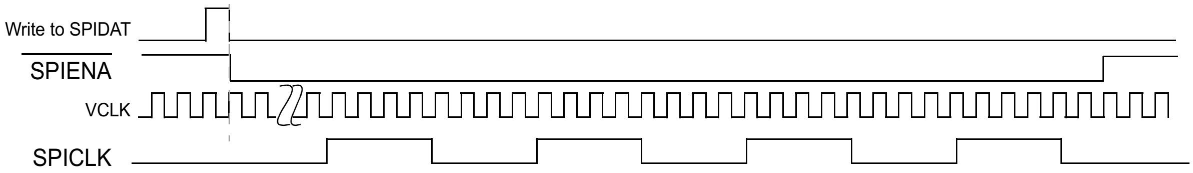
27.12.2 Slave Mode Timings for SPI/MibSPI

Figure 27-85. SPI/MibSPI Pins During Slave Mode 3-pin Configuration



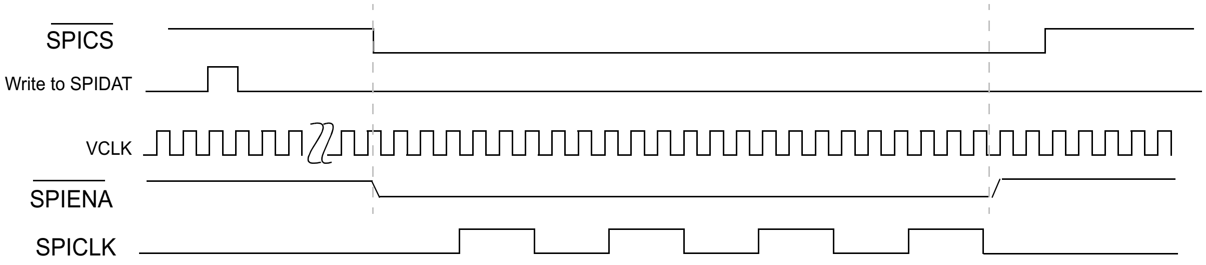
* Dotted vertical lines indicate the receive edges

Figure 27-86. SPI/MibSPI Pins During Slave Mode 4-pin with SPIENA Configuration



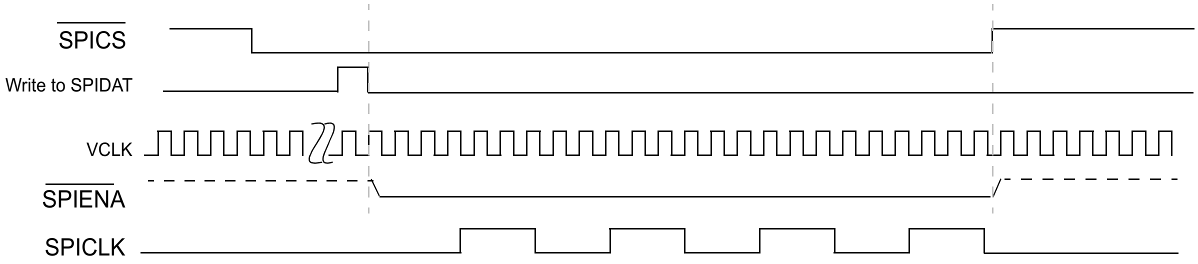
* Diagram shows a relationship between the SPIENA from Slave and SPICLK from Master

Figure 27-87. SPI/MibSPI Pins During Slave Mode in 5-pin Configuration - (Single Slave)



* ENABLE_HIGHZ is cleared to 0 in Slave SPI
* Diagram shows relationship between the SPICS from a Master to SPIENA from Slave SPI when SPIENA is configured in Push-Pull mode

Figure 27-88. SPI/MibSPI Pins During Slave Mode in 5-pin Configuration - (Single/Multi Slave)



* ENABLE_HIGHZ is set to 1 in Slave SPI
* Diagram shows relationship between the SPICS from a Master to SPIENA from Slave SPI when SPIENA is configured in High-Impedance mode

27.12.3 Master Mode Timing Parameter Details

In case of Master, the module drives out SPICLK. It also drives out the Transmit data on SPISIMO with respect to its internal SPICLK. In case of Master mode, the RX data on the SPISOMI pin is registered with respect to SPICLK received through the input buffer from the I/O pad.

If the chip select pin is functional, then the Master will drive out the $\overline{\text{SPICS}}$ pins before starting the SPICLK. If the $\overline{\text{SPIENA}}$ pin is functional, then the Master will wait for an active low from the Slave on the input pin to start the SPICLK.

27.12.4 Slave Mode Timing Parameter Details

In case of Slave mode, the module will drive only the SPISOMI and $\overline{\text{SPIENA}}$ pins. All other pins are inputs to it. The RX data on the SPISIMO pin will be registered with respect to the SPICLK pin. The Slave will use the $\overline{\text{SPICS}}$ pin to drive out the $\overline{\text{SPIENA}}$ pin if both are functional. If 4-pin with $\overline{\text{SPIENA}}$ is configured, then the Slave will drive out an active-low signal on the $\overline{\text{SPIENA}}$ pin when new data is written to the TX Shift Register. Irrespective of 4-pin with $\overline{\text{SPIENA}}$ or 5-pin configuration, the Slave will deassert the $\overline{\text{SPIENA}}$ pin after the last bit is received. If ENABLE_HIGHZ (SPIINT0.24) bit is 0, the de-asserted value of the $\overline{\text{SPIENA}}$ pin will be 1. Otherwise, it will depend upon the internal pull up or pull down resistor (if implemented) depending upon the Specification of the Chip.

Chapter 28
SPNU514C–March 2018

Serial Communication Interface (SCI)/ Local Interconnect Network (LIN) Module

This chapter describes the serial communication interface (SCI) / local interconnect network (LIN) module. The SCI/LIN is compliant to the LIN 2.1 protocol specified in the *LIN Specification Package*. This module can be configured to operate in either SCI (UART) or LIN mode.

NOTE: This chapter describes a superset implementation of the LIN/SCI module that includes features and functionality that require DMA. Since not all devices have DMA capability, consult your device-specific datasheet to determine applicability of these features and functions to your device being used.

Topic **Page**

28.1 Introduction and Features 1513
28.2 SCI Communication Formats 1518
28.3 SCI Interrupts 1526
28.4 SCI DMA Interface 1529
28.5 SCI Configurations 1530
28.6 SCI Low-Power Mode 1532
28.7 LIN Communication Formats 1533
28.8 LIN Interrupts 1551
28.9 LIN DMA Interface 1551
28.10 LIN Configurations 1552
28.11 Low-Power Mode 1554
28.12 Emulation Mode 1556
28.13 SCI/LIN Control Registers 1557
28.14 GPIO Functionality 1606

28.1 Introduction and Features

The SCI/LIN module can be programmed to work either as an SCI or as a LIN. The core of the module is an SCI. The SCI's hardware features are augmented to achieve LIN compatibility.

The SCI module is a universal asynchronous receiver-transmitter that implements the standard nonreturn to zero format. The SCI can be used to communicate, for example, through an RS-232 port or over a K-line.

The LIN standard is based on the SCI (UART) serial data link format. The communication concept is single-master/multiple-slave with a message identification for multi-cast transmission between any network nodes.

Throughout the chapter Compatibility Mode refers to SCI Mode functionary of SCI/LIN Module. The initial part of the chapter explains about the SCI functionality and later part about the LIN functionality. Though the register are common for LIN and SCI, the register descriptions has notes to identify the register / bit usage in different modes.

28.1.1 SCI Features

The following are the features of the SCI module:

- Standard universal asynchronous receiver-transmitter (UART) communication
- Supports full- or half-duplex operation
- Standard nonreturn to zero (NRZ) format
- Double-buffered receive and transmit functions in compatibility mode
- Supports two individually enabled interrupt lines: level 0 and level 1
- Configurable frame format of 3 to 13 bits per character based on the following:
  - Data word length programmable from one to eight bits
  - Additional address bit in address-bit mode
  - Parity programmable for zero or one parity bit, odd or even parity
  - Stop programmable for one or two stop bits
- Asynchronous or isosynchronous communication modes
- Two multiprocessor communication formats allow communication between more than two devices
- Sleep mode is available to free CPU resources during multiprocessor communication and then wake up to receive an incoming message
- The 24-bit programmable baud rate supports 2^{24} different baud rates provide high accuracy baud rate selection
- At 100MHz Peripheral Clock, 3.125 Mbits/s is the Max Baud Rate achievable
- Capability to use Direct Memory Access (DMA) for transmit and receive data
- Five error flags and Seven status flags provide detailed information regarding SCI events
- Two external pins: LINRX and LINTX
- Multi-buffered receive and transmit units

NOTE: SCI/LIN module does not support UART hardware flow control. This feature can be implemented in software using a general purpose I/O pin.

28.1.2 LIN Features

The following are the features of the LIN module:

- Compatibility with LIN 1.3, 2.0 and 2.1protocols
- Configurable Baud Rate up to 20 Kbits/s
- Two external pins: LINRX and LINTX.
- Multi-buffered receive and transmit units
- Identification masks for message filtering
- Automatic master header generation
  - Programmable synchronization break field
  - Synchronization field
  - Identifier field
- Slave automatic synchronization
  - Synchronization break detection
  - Optional baud rate update
  - Synchronization validation
- 2^{31} programmable transmission rates with 7 fractional bits
- Wakeup on LINRX dominant level from transceiver
- Automatic wakeup support
  - Wakeup signal generation
  - Expiration times on wakeup signals
- Automatic bus idle detection
- Error detection
  - Bit error
  - Bus error
  - No-response error
  - Checksum error
  - Synchronization field error
  - Parity error
- Capability to use Direct Memory Access (DMA) for transmit and receive data.
- 2 Interrupt lines with priority encoding for:
  - Receive
  - Transmit
  - ID, error, and status
- Support for LIN 2.0 checksum
- Enhanced synchronizer finite state machine (FSM) support for frame processing
- Enhanced handling of extended frames
- Enhanced baud rate generator
- Update wakeup/go to sleep

28.1.3 Block Diagram

The SCI/LIN module contains core SCI block with added sub-blocks to support LIN protocol.

Three Major components of the SCI Module are:

- Transmitter
- Baud Clock Generator
- Receiver

Transmitter (TX) contains two major registers to perform the double- buffering:

- The transmitter data buffer register (SCITD) contains data loaded by the CPU to be transferred to the shift register for transmission.
- The transmitter shift register (SCITXSHF) loads data from the data buffer (SCITD) and shifts data onto the LINTX pin, one bit at a time.

Baud Clock Generator

- A programmable baud generator produces either a baud clock scaled from VBUSP CLK.

Receiver (RX) contains two major registers to perform the double- buffering:

- The receiver shift register (SCIRXSHF) shifts data in from the LINRX pin one bit at a time and transfers completed data into the receive data buffer.
- The receiver data buffer register (SCIRD) contains received data transferred from the receiver shift register

The SCI receiver and transmitter are double-buffered, and each has its own separate enable and interrupt bits. The receiver and transmitter may each be operated independently or simultaneously in full duplex mode.

To ensure data integrity, the SCI checks the data it receives for breaks, parity, overrun, and framing errors. The bit rate (baud) is programmable to over 16 million different rates through a 24-bit baud-select register. Figure 28-1 shows the detailed SCI block diagram.

The SCI/LIN module is based on the standalone SCI with the addition of an error detector (parity calculator, checksum calculator, and bit monitor), a mask filter, a synchronizer, and a multi-buffered receiver and transmitter. The SCI interface, the DMA control subblocks and the baud generator are modified as part of the hardware enhancements for LIN compatibility. Figure 28-2 shows the SCI/LIN block diagram.

Figure 28-1. Detailed SCI Block Diagram

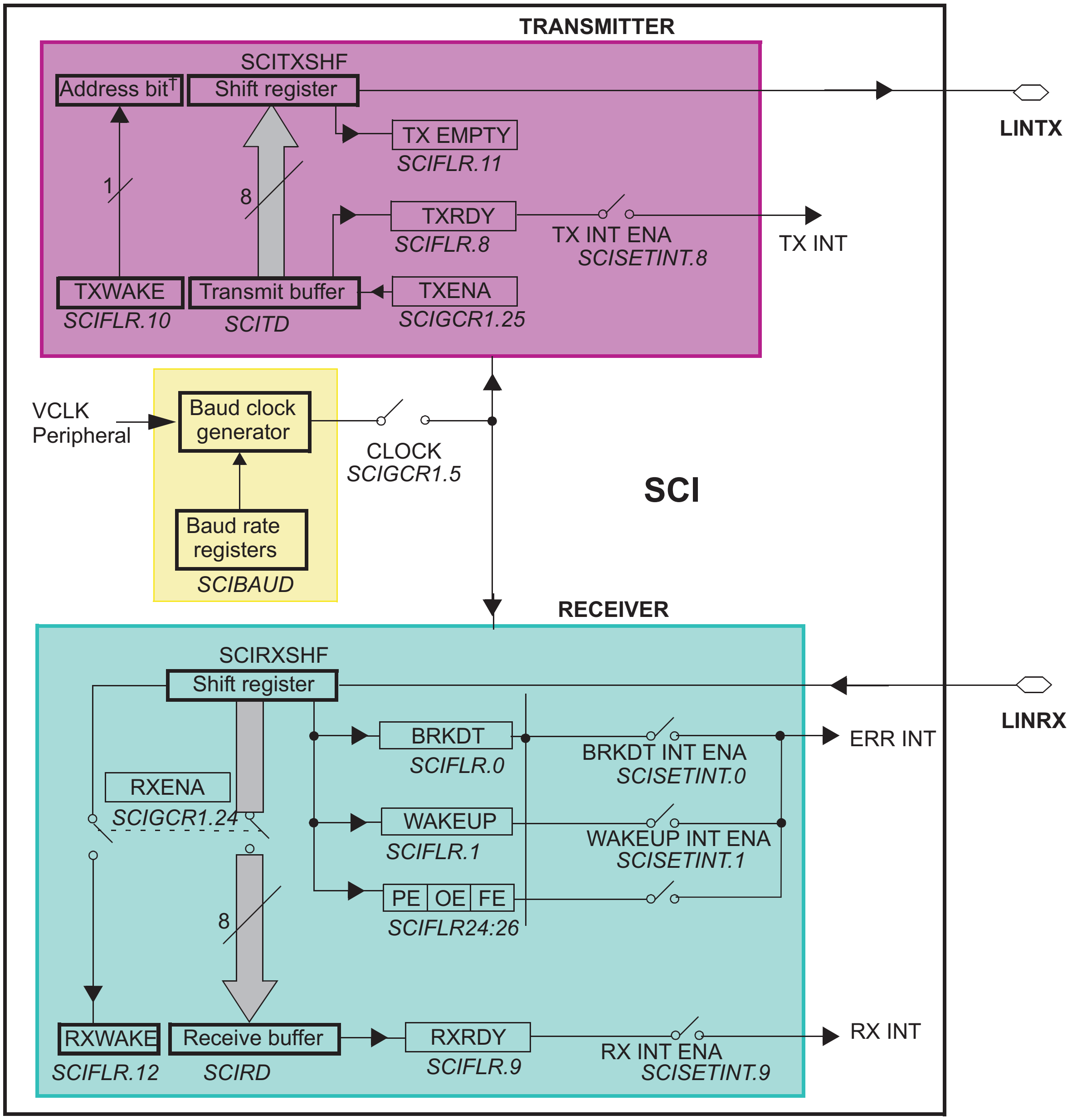
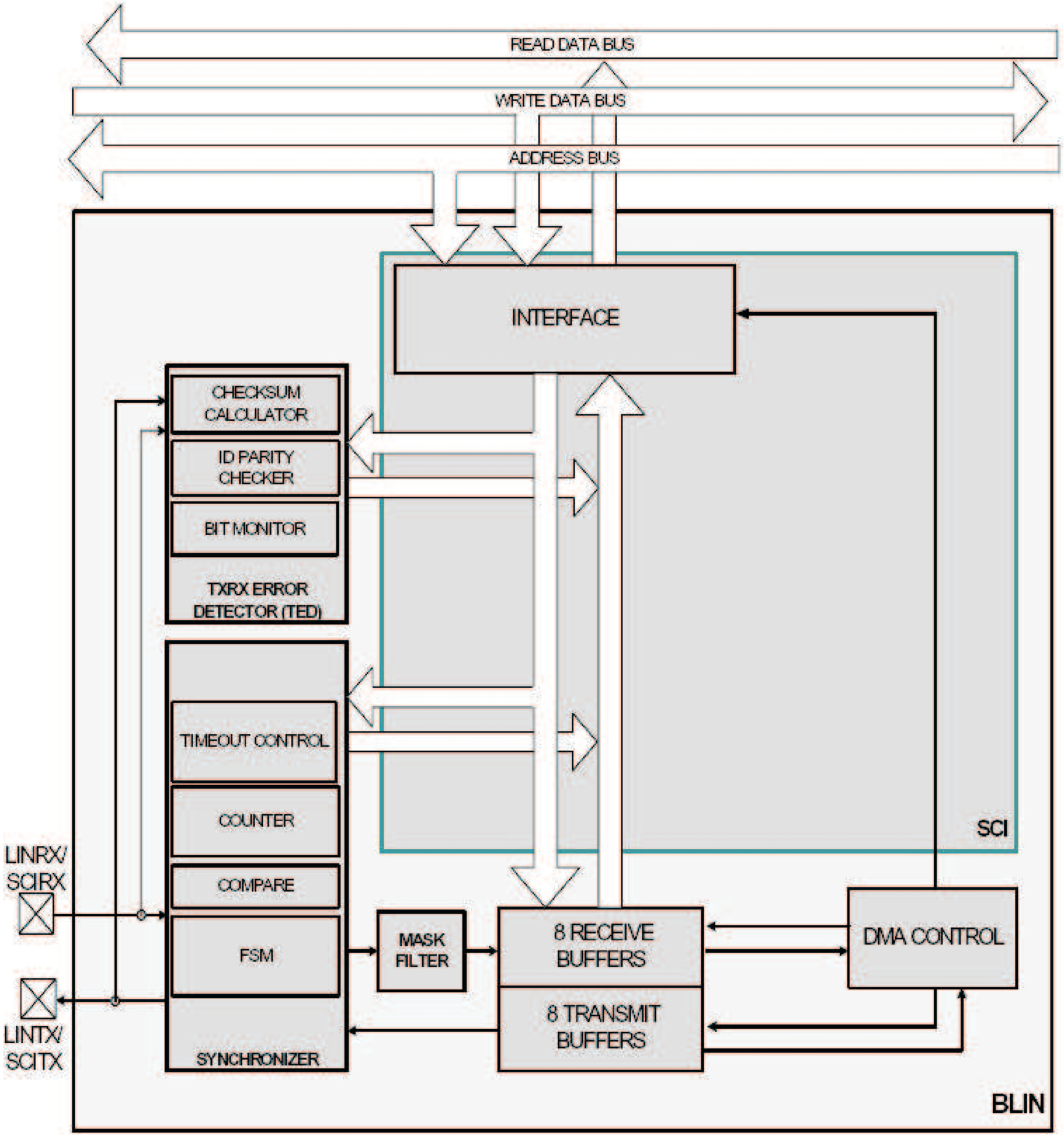


Figure 28-2. SCI/LIN Block Diagram



28.2 SCI Communication Formats

The SCI module can be configured to meet the requirements of many applications. Because communication formats vary depending on the specific application, many attributes of the SCI/LIN are user configurable. The list below describes these configuration options:

- SCI Frame format
- SCI Timing modes
- SCI Baud rate
- SCI Multiprocessor modes

28.2.1 SCI Frame Formats

The SCI uses a programmable frame format. All frames consist of the following:

- One start bit
- One to eight data bits
- Zero or one address bit
- Zero or one parity bit
- One or two stop bits

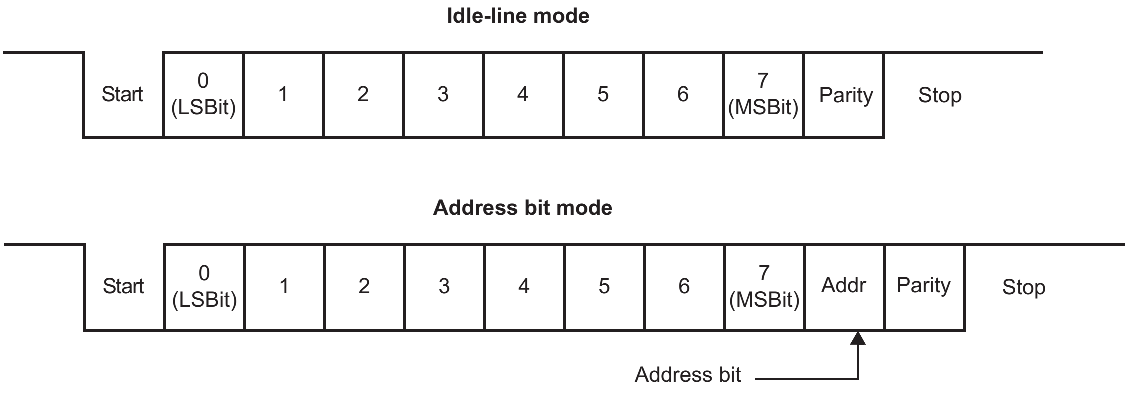
The frame format for both the transmitter and receiver is programmable through the bits in the SCIGCR1 register. Both receive and transmit data is in nonreturn to zero (NRZ) format, which means that the transmit and receive lines are at logic high when idle. Each frame transmission begins with a start bit, in which the transmitter pulls the SCI line low (logic low). Following the start bit, the frame data is sent and received least significant bit first (LSB).

An address bit is present in each frame if the SCI is configured to be in address-bit mode but is not present in any frame if the SCI is configured for idle-line mode. The format of frames with and without the address bit is illustrated in Figure 28-3.

A parity bit is present in every frame when the PARITY ENA bit is set. The value of the parity bit depends on the number of one bits in the frame and whether odd or even parity has been selected via the PARITY ENA bit. Both examples in Figure 28-3 have parity enabled.

All frames include one stop bit, which is always a high level. This high level at the end of each frame is used to indicate the end of a frame to ensure synchronization between communicating devices. Two stop bits are transmitted if the STOP bit in SCIGCR1 register is set. The examples shown in Figure 28-3 use one stop bit per frame.

Figure 28-3. Typical SCI Data Frame Formats



Idle-line mode

Address bit mode

28.2.2 SCI Timing Mode

The SCI can be configured to use asynchronous or isosynchronous timing using TIMING MODE bit in SCIGCR1 register.

28.2.2.1 Asynchronous Timing Mode

The asynchronous timing mode uses only the receive and transmit data lines to interface with devices using the standard universal asynchronous receiver- transmitter (UART) protocol.

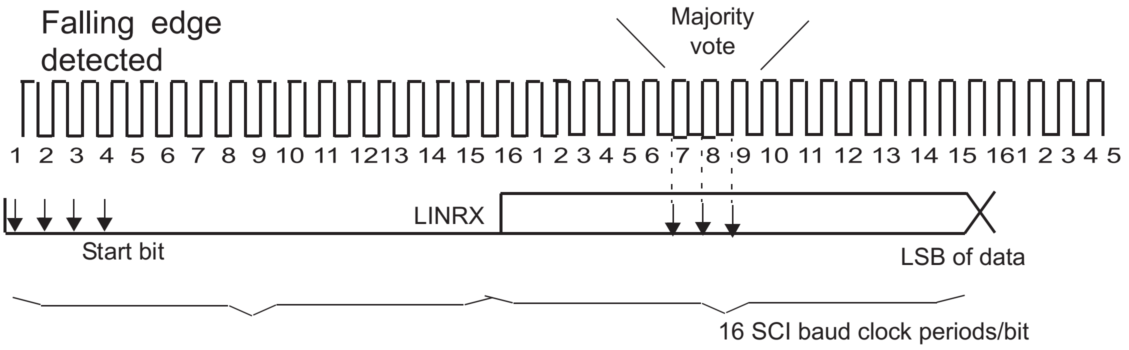
In the asynchronous timing mode, each bit in a frame has a duration of 16 SCI baud clock periods. Each bit therefore consists of 16 samples (one for each clock period). When the SCI is using asynchronous mode, the baud rates of all communicating devices must match as closely as possible. Receive errors result from devices communicating at different baud rates.

With the receiver in the asynchronous timing mode, the SCI detects a valid start bit if the first four samples after a falling edge on the LINRX pin are of logic level 0. As soon as a falling edge is detected on LINRX, the SCI assumes that a frame is being received and synchronizes itself to the bus.

To prevent interpreting noise as Start bit SCI expects LINRX line to be low for at least four contiguous SCI baud clock periods to detect a valid start bit. The bus is considered idle if this condition is not met. When a valid start bit is detected, the SCI determines the value of each bit by sampling the LINRX line value during the seventh, eighth, and ninth SCI baud clock periods. A majority vote of these three samples is used to determine the value stored in the SCI receiver shift register. By sampling in the middle of the bit, the SCI reduces errors caused by propagation delays and rise and fall times and data line noises. Figure 28-4 illustrates how the receiver samples a start bit and a data bit in asynchronous timing mode.

The transmitter transmits each bit for a duration of 16 SCI baud clock periods. During the first clock period for a bit, the transmitter shifts the value of that bit onto the LINTX pin. The transmitter then holds the current bit value on LINTX for 16 SCI baud clock periods.

Figure 28-4. Asynchronous Communication Bit Timing



28.2.2.2 Isosynchronous Timing Mode

In isosynchronous timing mode, each bit in a frame has a duration of exactly 1 baud clock period and therefore consists of a single sample. With this timing configuration, the transmitter and receiver are required to make use of the SCICLK pin to synchronize communication with other SCI. **This mode is not supported on this device because SCICLK pin is not available.**

28.2.3 SCI Baud Rate

The SCI/LIN has an internally-generated serial clock determined by the peripheral VCLK and the prescalers P and M in this register. The SCI uses the 24-bit integer prescaler P value of the BRS register to select the required baud rates. The additional 4-bit fractional divider M refines the baud rate selection.

In asynchronous timing mode, the SCI generates a baud clock according to the following formula:

$$SCICLK\ Frequency = \frac{\text{VCLK Frequency}}{P + 1 + \frac{M}{16}}$$

$$Asynchronous\ baud\ value = \frac{\text{SCICLK Frequency}}{16}$$

For P = 0,

$$Asynchronous\ baud\ value = \frac{\text{VCLK Frequency}}{32} \tag{42}$$

28.2.3.1 Superfractional Divider, SCI Asynchronous Mode

The superfractional divider is available in SCI asynchronous mode (idle-line and address-bit mode). Building on the 4-bit fractional divider M (BRS[27:24]), the super fractional divider uses an additional 3-bit modulating value, illustrated in Table 28-2. The bits with a 1 in the table will have an additional VCLK period added to their T_{bit}. If the character length is more than 10, then the modulation table will be a rolled-over version of the original table (Table 28-1), as shown in Table 28-2.

The baud rate will vary over a data field to average according to the BRS[30:28] value by a “**d**” fraction of the peripheral internal clock: **0<d<1.** Refer Figure 28-5 for a simple Average “d’ calculation based on “U” value (BRS[30:28]).

The instantaneous bit time is expressed in terms of T_{VCLK} as follows:

For all P other than 0, and all M and d (0 or 1),

$$T^{i}bit = \left[16\left(P + 1 + \frac{M}{16}\right) + d\right]T_{VCLK} \tag{43}$$

For P = 0 $T_{bit} = 32T_{VCLK}$

The averaged bit time is expressed in terms of T_{VCLK} as follows:

For all P other than 0, and all M and d (0<d<1),

$$T^{a}bit = \left[16\left(P + 1 + \frac{M}{16}\right) + d\right]T_{VCLK} \tag{44}$$

For P = 0 $T_{bit} = 32T_{VCLK}$

Table 28-1. Superfractional Bit Modulation for SCI Mode (Normal Configuration)[1]

BRS[30:28]	Start Bit	D[0]	D[1]	D[2]	D[3]	D[4]	D[5]	D[6]	D[7]	Stop Bit
0h	0	0	0	0	0	0	0	0	0	0
1h	1	0	0	0	0	0	0	0	1	0
2h	1	0	0	0	1	0	0	0	1	0
3h	1	0	1	0	1	0	0	0	1	0
4h	1	0	1	0	1	0	1	0	1	0
5h	1	1	1	0	1	0	1	0	1	1
6h	1	1	1	0	1	1	1	0	1	1
7h	1	1	1	1	1	1	1	0	1	1

[1] Normal configuration = Start + 8 Data Bits + Stop Bit

Table 28-2. Superfractional Bit Modulation for SCI Mode (Maximum Configuration)[1]

BRS[30:28]	Start Bit	D[0]	D[1]	D[2]	D[3]	D[4]	D[5]	D[6]	D[7]	Addr	Parity	Stop0	Stop1
0h	0	0	0	0	0	0	0	0	0	0	0	0	0
1h	1	0	0	0	0	0	0	0	1	0	0	0	0
2h	1	0	0	0	1	0	0	0	1	0	0	0	1
3h	1	0	1	0	1	0	0	0	1	0	1	0	1
4h	1	0	1	0	1	0	1	0	1	0	1	0	1
5h	1	1	1	0	1	0	1	0	1	1	1	0	1
6h	1	1	1	0	1	1	1	0	1	1	1	0	1
7h	1	1	1	1	1	1	1	0	1	1	1	1	1

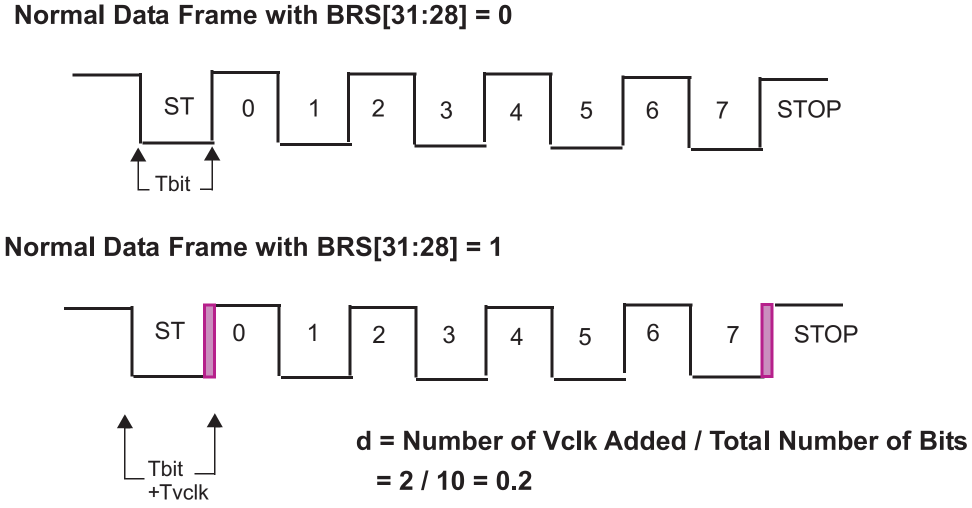
[1] Maximum configuration = Start + 8 Data Bits + Addr Bit + Parity Bit + Stop Bit 0 + Stop Bit 1

Table 28-3. SCI Mode (Minimum Configuration)[1]

BRS[30:28]	Start Bit	D[0]	Stop
0h	0	0	0
1h	1	0	0
2h	1	0	0
3h	1	0	1
4h	1	0	1
5h	1	1	1
6h	1	1	1
7h	1	1	1

[1] Minimum configuration = Start + 1 Data Bits + Stop Bit

Figure 28-5. Superfractional Divider Example


Normal Data Frame with BRS[31:28] = 0
ST
0
1
2
3
4
5
6
7
STOP
Tbit
Normal Data Frame with BRS[31:28] = 1
ST
0
1
2
3
4
5
6
7
STOP
Tbit
+Tvclk
d = Number of Vclk Added / Total Number of Bits
= 2 / 10 = 0.2

28.2.4 SCI Multiprocessor Communication Modes

In some applications, the SCI may be connected to more than one serial communication device. In such a multiprocessor configuration, several frames of data may be sent to all connected devices or to an individual device. In the case of data sent to an individual device, the receiving devices must determine when they are being addressed. When a message is not intended for them, the devices can ignore the following data. When only two devices make up the SCI network, addressing is not needed, so multiprocessor communication schemes are not required.

SCI supports two multiprocessor communication modes which can be selected using COMM MODE bit:

- Idle-Line Mode
- Address Bit Mode

When the SCI is not used in a multiprocessor environment, software can consider all frames as data frames. In this case, the only distinction between the idle-line and address-bit modes is the presence of an extra bit (the address bit) in each frame sent with the address-bit protocol.

The SCI allows full-duplex communication where data can be sent and received via the transmit and receive pins simultaneously. However, the protocol used by the SCI assumes that only one device transmits data on the same bus line at any one time. No arbitration is done by the SCI.

28.2.4.1 Idle-Line Multiprocessor Modes

In idle-line multiprocessor mode, a frame that is preceded by an idle period (10 or more idle bits) is an address frame. A frame that is preceded by fewer than 10 idle bits is a data frame. Figure 28-6 illustrates the format of several blocks and frames with idle-line mode.

There are two ways to transmit an address frame using idle-line mode:

Method 1: In software, deliberately leave an idle period between the transmission of the last data frame of the previous block and the address frame of the new block.

Method 2: Configure the SCI to automatically send an idle period between the last data frame of the previous block and the address frame of the new block.

Although Method 1 is only accomplished by a delay loop in software, Method 2 can be implemented by using the transmit buffer and the TXWAKE bit in the following manner:

Step 1: Write a 1 to the TXWAKE bit.

Step 2: Write a dummy data value to the SCITD register. This triggers the SCI to begin the idle period as soon as the transmitter shift register is empty.

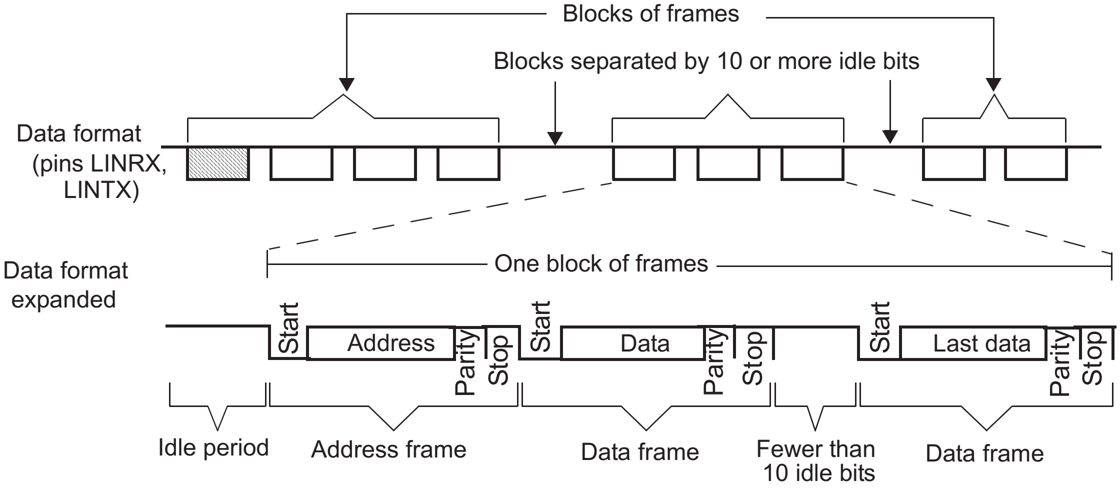
Step 3: Wait for the SCI to clear the TXWAKE flag.

Step 4: Write the address value to SCITD.

As indicated by Step 3, software should wait for the SCI to clear the TXWAKE bit. However, the SCI clears the TXWAKE bit at the same time it sets TXRDY (that is, transfers data from SCITD into SCITXSHF). Therefore, if the TX INT ENA bit is set, the transfer of data from SCITD to SCITXSHF causes an interrupt to be generated at the same time that the SCI clears the TXWAKE bit. If this interrupt method is used, software is not required to poll the TXWAKE bit waiting for the SCI to clear it.

When idle-line multiprocessor communications are used, software must ensure that the idle time exceeds 10 bit periods before addresses (using one of the methods mentioned above), and software must also ensure that data frames are written to the transmitter quickly enough to be sent without a delay of 10 bit periods between frames. Failure to comply with these conditions will result in data interpretation errors by other devices receiving the transmission.

Figure 28-6. Idle-Line Multiprocessor Communication Format



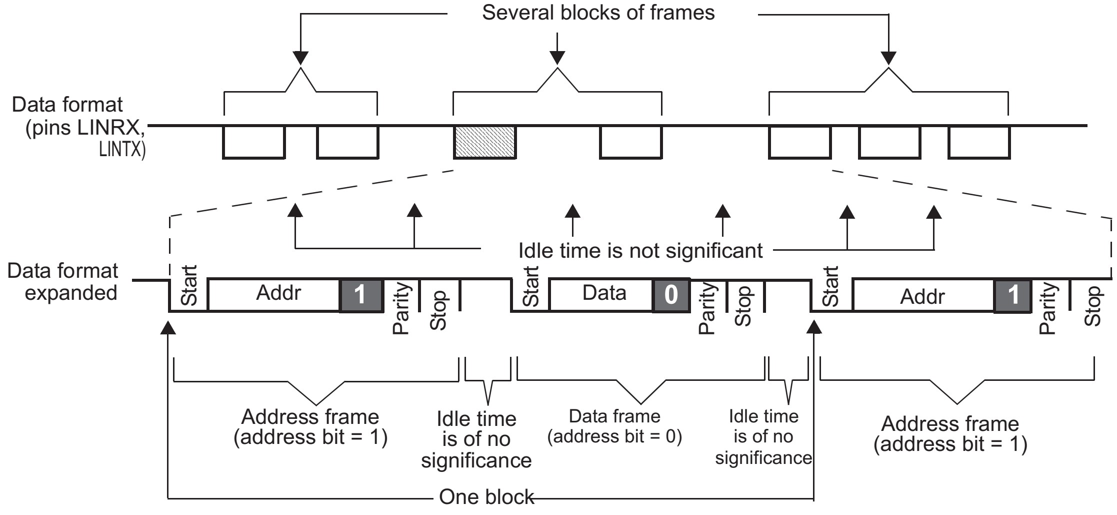
28.2.4.2 Address-Bit Multiprocessor Mode

In the address-bit protocol, each frame has an extra bit immediately following the data field called an address bit. A frame with the address bit set to 1 is an address frame; a frame with the address bit set to 0 is a data frame. The idle period timing is irrelevant in this mode. Figure 28-7 illustrates the format of several blocks and frames with the address-bit mode.

When address-bit mode is used, the value of the TXWAKE bit is the value sent as the address bit. To send an address frame, software must set the TXWAKE bit. This bit is cleared as the contents of the SCITD are shifted from the TXWAKE register so that all frames sent are data except when the TXWAKE bit is written as a 1.

No dummy write to SCITD is required before an address frame is sent in address-bit mode. The first byte written to SCITD after the TXWAKE bit is written to 1 is transmitted with the address bit set when address-bit mode is used.

Figure 28-7. Address-Bit Multiprocessor Communication Format



28.2.5 SCI Multi-Buffered Mode

To reduce CPU load when Receiving or Transmitting data in interrupt mode or DMA mode, the SCI/LIN module has eight separate Receive and transmit buffers. Multi buffered mode is enabled by setting the MBUF MODE bit in the SCIGCR1 register.

The multi-buffer 3-bit counter counts the data bytes transferred from the SCIRXSHF register to the RDy receive buffers and TDy transmit buffers register to SCITXSHF register. The 3-bit compare register contains the number of data bytes expected to be received or transmitted. the LENGTH value in SCIFORMAT register indicates the expected length and is used to load the 3-bit compare register.

A receive interrupt (RX interrupt; see the SCIINTVECT0 and SCIINTVECT1registers), and a receive ready RXRDY flag set in SCIFLR register, as well as a DMA request (RXDMA) could occur after receiving a response if there are no response receive errors for the frame (such as, there is, frame error, and overrun error).

A transmit interrupt (TX interrupt), and a transmit ready flag (TXRDY flag in SCIFLR register), and a DMA request (TXDMA) could occur after transmitting a response.

Figure 28-8 and Figure 28-9 shows the receive and transmit multi-buffer functional block diagram.

Figure 28-8. Receive Buffers

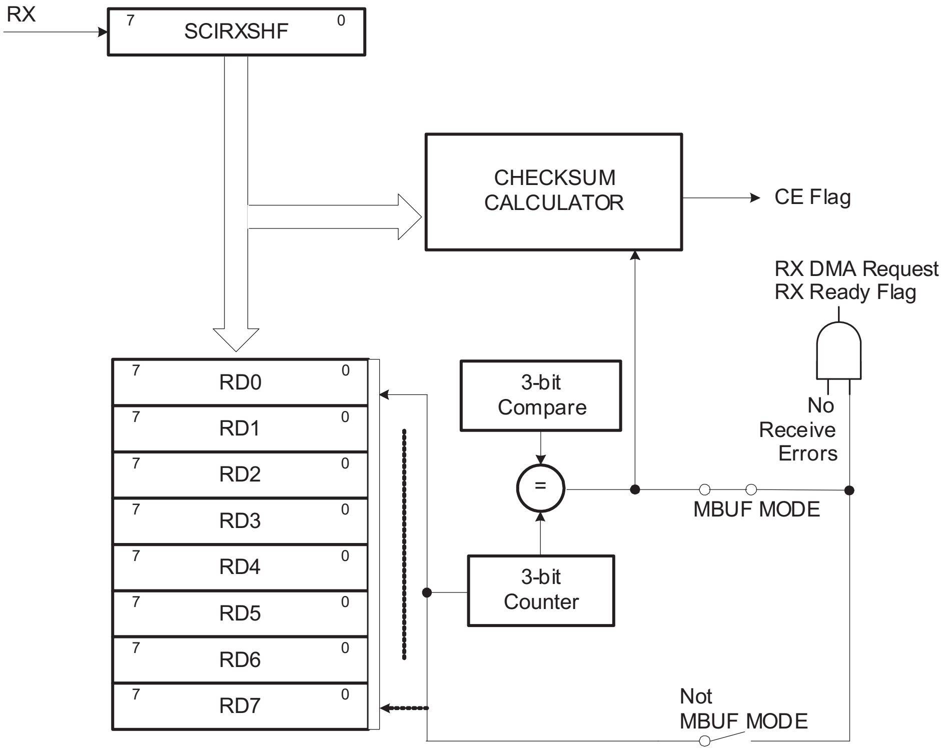
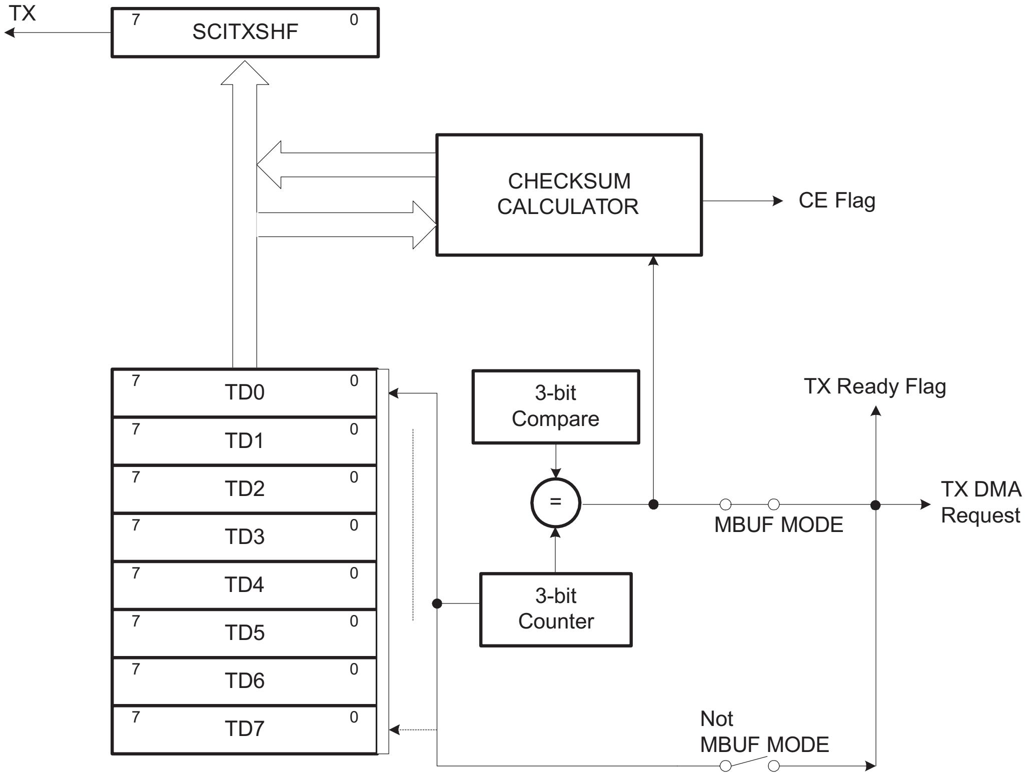


Figure 28-9. Transmit Buffers



28.3 SCI Interrupts

The SCI/LIN module has two interrupt lines, level 0 and level 1, to the vectored interrupt manager (VIM) module (see Figure 28-10). Two offset registers SCIINTVECT0 and SCIINTVECT1 determine which flag triggered the interrupt according to the respective priority encoders. Each interrupt condition has a bit to enable/disable the interrupt in the SCISETINT and SCICLRINT registers respectively.

Each interrupt also has a bit that can be set as interrupt level 0(INT0) or as interrupt level 1(INT1). By default, interrupts are in interrupt level 0. SCISETINTLVL sets a given interrupt to level1. SCICLEARINTLVL resets a given interrupt level to the default level 0.

The interrupt vector registers SCIINTVECT0 and SCIINTVECT1 return the vector of the pending interrupt line INT0 or INT1. If more than one interrupt is pending, the interrupt vector register holds the highest priority interrupt.

Figure 28-10. General Interrupt Scheme

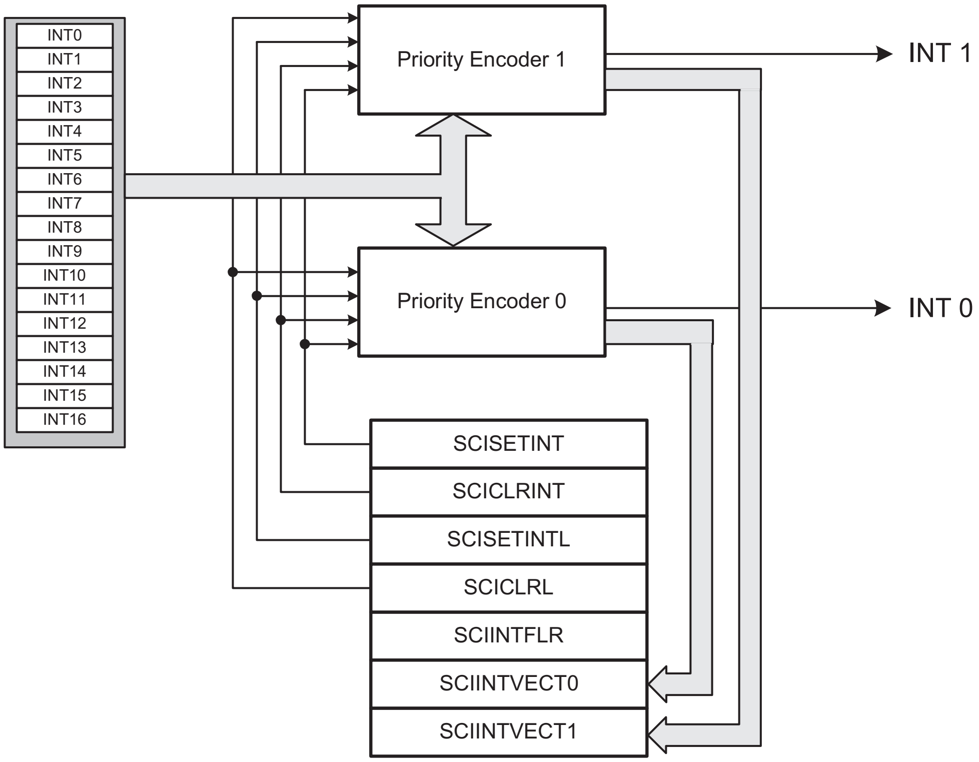
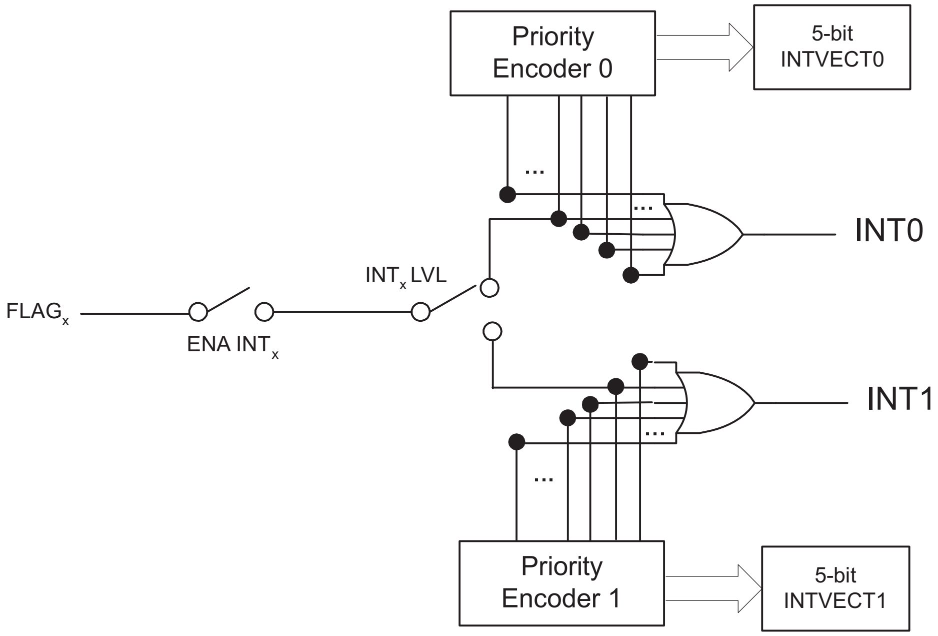


Figure 28-11. Interrupt Generation for Given Flags



28.3.1 Transmit Interrupt

To use transmit interrupt functionality, SET TX INT bit must be enabled and SET TX DMA bit must be cleared. The transmit ready (TXRDY) flag is set when the SCI transfers the contents of SCITD to the shift register, SCITXSHF. The TXRDY flag indicates that SCITD is ready to be loaded with more data. In addition, the SCI sets the TX EMPTY bit if both the SCITD and SCITXSHF registers are empty. If the SET TX INT bit is set, then a transmit interrupt is generated when the TXRDY flag goes high. Transmit Interrupt is not generated immediately after setting the SET TX INT bit unlike transmit DMA request. Transmit Interrupt is generated only after the first transfer from SCITD to SCITXSHF, that is first data has to be written to SCITD by the User before any interrupt gets generated. To transmit further data the user can write data to SCITD in the transmit Interrupt service routine.

Writing data to the SCITD register clears the TXRDY bit. When this data has been moved to the SCITXSHF register, the TXRDY bit is set again. The interrupt request can be suspended by setting the CLR TX INT bit; however, when the SET TX INT bit is again set to 1, the TXRDY interrupt is asserted again. The transmit interrupt request can be eliminated until the next series of values is written to SCITD, by disabling the transmitter via the TXENA bit, by a software reset SWnRST, or by a device hardware reset.

28.3.2 Receive Interrupt

The receive ready (RXRDY) flag is set when the SCI transfers newly received data from SCIRXSHF to SCIRD. The RXRDY flag therefore indicates that the SCI has new data to be read. Receive interrupts are enabled by the SET RX INT bit. If the SET RX INT is set when the SCI sets the RXRDY flag, then a receive interrupt is generated. The received data can be read in the Interrupt Service routine.

On a device with both SCI and a DMA controller, the bits SET RX DMA ALL and SET RX DMA must be cleared to select interrupt functionality.

28.3.3 WakeUp Interrupt

SCI sets the WAKEUP flag if bus activity on the RX line either prevents power-down mode from being entered, or RX line activity causes an exit from power-down mode. If enabled (SET WAKEUP INT), wakeup interrupt is triggered once WAKEUP flag is set.

28.3.4 Error Interrupts

The following error detection are supported with Interrupt by the SCI module:

- Parity errors (PE)
- Frame errors (FE)
- Break Detect errors (BRKDT)
- Overrun errors (OE)
- Bit errors (BE)

If all of these errors (PE, FE, BRKDT, OE, BE) are flagged, an interrupt for the flagged errors will be generated if enabled. A message is valid for both the transmitter and the receiver if there is no error detected until the end of the frame. Each of these flags is located in the receiver status (SCIFLR) register.

There are 16 interrupt sources in the SCI/LIN module, In SCI mode 8 interrupts are supported, as seen in Table 28-4.

Table 28-4. SCI/LIN Interrupts

Offset [1]	Interrupt	Applicable to SCI	Applicable to LIN
0	No interrupt		
1	Wakeup	Yes	Yes
2	Inconsistent-synch-field error	No	Yes
3	Parity error	Yes	Yes
4	ID	No	Yes
5	Physical bus error	No	Yes
6	Frame error	Yes	Yes
7	Break detect	Yes	No
8	Checksum error	No	Yes
9	Overrun error	Yes	Yes
10	Bit error	Yes	Yes
11	Receive	Yes	Yes
12	Transmit	Yes	Yes
13	No-response error	No	Yes
14	Timeout after wakeup signal (150 ms)	No	Yes
15	Timeout after three wakeup signals (1.5 s)	No	Yes
16	Timeout (Bus Idle, 4s)	No	Yes

[1] Offset 1 is the highest priority. Offset 16 is the lowest priority.

28.4 SCI DMA Interface

DMA requests for receive (RXDMA request) and transmit (TXDMA request) are available for the SCI/LIN module. The DMA transfers depending on whether multi-buffer mode bit (MBUF MODE in the SCIGCR1 register) is enabled or is not enabled. See Chapter 29 for additional information.

28.4.1 Receive DMA Requests

This DMA functionality is enabled/disabled by the CPU using the SET RX DMA/CLR RX DMA bits, respectively.

In Multi-Buffered SCI mode with DMA enabled, the receiver loads the RDy buffers for each received character. RX DMA request is triggered once the last character of the programmed number of characters (LENGTH) are received and copied to the corresponding RDy buffer successfully.

If the multi-buffer option is disabled, then DMA requests will be generated on a byte-per-byte basis.

In multiprocessor mode, the SCI can generate receiver interrupts for address frames and DMA requests for data frames or DMA requests for both. This is controlled by the SET RX DMA ALL bit.

In multiprocessor mode with the SLEEP bit set, no DMA is generated for received data frames. The software must clear the SLEEP bit before data frames can be received.

28.4.2 Transmit DMA Requests

DMA functionality is enabled/disabled by the CPU with SET TX DMA/CLR TX DMA bits, respectively.

In Multi-Buffered SCI mode, once the TXRDY bit is set or after a transmission of programmed number of characters (LENGTH) (up to eight data bytes stored in the transmit buffer(s) TDy in the LINTD0 and LINTD1 registers), a DMA request is generated in order to reload the transmit buffer for the next transmission. If the multi-buffer option is disabled, then DMA requests will be generated on a byte-per-byte basis.

28.5 SCI Configurations

Before the SCI sends or receives data, its registers should be properly configured. Upon power-up or a system-level reset, each bit in the SCI registers is set to a default state. The registers are writable only after the RESET bit in the SCIGCR0 register is set to 1. Of particular importance is the SWnRST bit in the SCIGCR1 register. The SWnRST is an active-low bit initialized to 0 and keeps the SCI in a reset state until it is programmed to 1. Therefore, all SCI configuration should be completed before a 1 is written to the SWnRST bit.

The following list details the configuration steps that software should perform prior to the transmission or reception of data. As long as the SWnRST bit is cleared to 0 the entire time that the SCI is being configured, the order in which the registers are programmed is not important.

- Enable SCI by setting the RESET bit to 1.
- Clear the SWnRST bit to 0 before SCI is configured.
- Select the desired frame format by programming the SCIGCR1 register.
- Set both the RX FUNC and TX FUNC bits in SCIPIO0 to 1 to configure the LINRX and LINTX pins for SCI functionality.
- Select the baud rate to be used for communication by programming the BRS register.
- Set the CLOCK bit in SCIGCR1 to 1 to select the internal clock.
- Set the CONT bit in SCIGCR1 to 1 to make SCI not halt for an emulation breakpoint until its current reception or transmission is complete (this bit is used only in an emulation environment).
- Set the LOOP BACK bit in SCIGCR1 to 1 to connect the transmitter to the receiver internally (this feature is used to perform a self-test).
- Set the RXENA bit in SCIGCR1 to 1, if data is to be received.
- Set the TXENA bitin SCIGCR1 to 1, if data is to be transmitted.
- Set the SWnRST bit to 1 after SCI is configured.
- Perform receiving or transmitting data (see Section 28.5.1 or Section 28.5.2).

28.5.1 Receiving Data

The SCI receiver is enabled to receive messages if both the RX FUNC bit and the RXENA bit are set to 1. If the RX FUNC bit is not set, the LINRX pin functions as a general-purpose I/O pin rather than as an SCI function pin.

SCI module can receive data in one of the following modes:

- Single-Buffer (Normal) Mode
- Multi-Buffer Mode

After a valid idle period is detected, data is automatically received as it arrives on the LINRX pin.

28.5.1.1 Receiving Data in Single-Buffer Mode

Single-buffer mode is selected when the MBUF MODE bit in SCIGCR1 is cleared to 0. In this mode, SCI sets the RXRDY bit when it transfers newly received data from SCIRXSHF to SCIRD. The SCI clears the RXRDY bit after the new data in SCIRD has been read. Also, as data is transferred from SCIRXSHF to SCIRD, the SCI sets the FE, OE, or PE flags if any of these error conditions were detected in the received data. These error conditions are supported with configurable interrupt capability. The wake-up and break-detect status bits are also set if one of these errors occurs, but they do not necessarily occur at the same time that new data is being loaded into SCIRD.

You can receive data by:

1. Polling Receive Ready Flag
2. Receive Interrupt
3. DMA

In polling method, software can poll for the RXRDY bit and read the data from the SCIRD register once the RXRDY bit is set high. The CPU is unnecessarily overloaded by selecting the polling method. To avoid this, you can use either the interrupt or DMA method. To use the interrupt method, the SET RX INT bit is set. To use the DMA method, the SET RX DMA bit is set. Either an interrupt or a DMA request is generated the moment the RXRDY bit is set.

28.5.1.2 Receiving Data in Multi-Buffer Mode

Multi-buffer mode is selected when the MBUF MODE bit in SCIGCR1 is set to 1. In this mode, SCI sets the RXRDY bit after receiving the programmed number of data in the receive buffer, the complete frame. The error condition detection logic is similar to the single-buffer mode, except that it monitors for the complete frame. Like single-buffer mode, you can use the polling, interrupt, or DMA method to read the data. The SCI clears the RXRDY bit after the new data in SCIRD has been read.

28.5.2 Transmitting Data

The SCI transmitter is enabled if both the TX FUNC bit and the TXENA bit are set to 1. If the TX FUNC bit is not set, the LINTX pin functions as a general-purpose I/O pin rather than as an SCI function pin. Any value written to the SCITD before TXENA is set to 1 is not transmitted. Both of these control bits allow for the SCI transmitter to be held inactive independently of the receiver.

SCI module can transmit data in one of the following modes:

- Single-Buffer (Normal) Mode
- Multi-Buffered or Buffered SCI Mode

28.5.2.1 Transmitting Data in Single-Buffer Mode

Single-buffer mode is selected when the MBUF MODE bit in SCIGCR1 is cleared to 0. In this mode, SCI waits for data to be written to SCITD, transfers it to SCITXSHF, and transmits the data. The TXRDY and TX EMPTY bits indicate the status of the transmit buffers. That is, when the transmitter is ready for data to be written to SCITD, the TXRDY bit is set. Additionally, if both SCITD and SCITXSHF are empty, then the TX EMPTY bit is also set.

You can transmit data by:

1. Polling Transmit Ready Flag
2. Transmit Interrupt
3. DMA

In polling method, software can poll for the TXRDY bit to go high before writing the data to the SCITD register. The CPU is unnecessarily overloaded by selecting the polling method. To avoid this, you can use either the interrupt or DMA method. To use the interrupt method, the SET TX INT bit is set. To use the DMA method, the SET TX DMA bit is set. Either an interrupt or a DMA request is generated the moment the TXRDY bit is set. When the SCI has completed transmission of all pending frames, the SCITXSHF register and SCITD are empty, the TXRDY bit is set, and an interrupt/DMA request is generated, if enabled. Because all data has been transmitted, the interrupt/DMA request should be halted. This can either be done by disabling the transmit interrupt (CLR TX INT) / DMA request (CLR TX DMA bit) or by disabling the transmitter (clear TXENA bit).

NOTE: The TXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVECT0 or SCIINTVECT1 register.

28.5.2.2 Transmitting Data in Multi-Buffer Mode

Multi-buffer mode is selected when the MBUF MODE bit in SCIGCR1 is set to 1. Like single-buffer mode, you can use the polling, interrupt, or DMA method to write the data to be transmitted. The transmitted data has to be written to the SCITD registers. SCI waits for data to be written to the SCITD register and transfers the programmed number of bytes to SCITXSHF to transmit one by one automatically.

28.6 SCI Low-Power Mode

The SCI/LIN 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 SCI/LIN. During global low-power mode, all clocks to the SCI/LIN are turned off so the module is completely inactive.

Local low-power mode is asserted by setting the POWERDOWN bit; setting this bit stops the clocks to the SCI/LIN internal logic and the module registers. Setting the POWERDOWN bit causes the SCI to enter local low-power mode and clearing the POWERDOWN bit causes SCI/LIN 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 SCI for that particular access alone.

The wake-up interrupt is used to allow the SCI to exit low-power mode automatically when a low level is detected on the LINRX pin and also this clears the POWERDOWN bit. If wake-up interrupt is disabled, then the SCI/LIN immediately enters low-power mode whenever it is requested and also any activity on the LINRX pin does not cause the SCI to exit low-power mode.

NOTE: Enabling Local Low-Power Mode During Receive and Transmit

If the wake-up interrupt is enabled and low-power mode is requested while the receiver is receiving data, then the SCI immediately generates a wake-up interrupt to clear the powerdown bit and prevents the SCI from entering low-power mode and thus completes the current reception. Otherwise, if the wake-up interrupt is disabled, then the SCI completes the current reception and then enters the low-power mode.

28.6.1 Sleep Mode for Multiprocessor Communication

When the SCI receives data and transfers that data from SCIRXSHF to SCIRD, the RXRDY bit is set and if RX INT ENA is set, the SCI also generates an interrupt. The interrupt triggers the CPU to read the newly received frame before another one is received. In multiprocessor communication modes, this default behavior may be enhanced to provide selective indication of new data. When SCI receives an address frame that does not match its address, the device can ignore the data following this non-matching address until the next address frame by using sleep mode. Sleep mode can be used with both idle-line and address-bit multiprocessor modes.

If sleep mode is enabled by the SLEEP bit, then the SCI transfers data from SCIRXSHF to SCIRD only for address frames. Therefore, in sleep mode, all data frames are assembled in the SCIRXSHF register without being shifted into the SCIRD and without initiating a receive interrupt or DMA request. Upon reception of an address frame, the contents of the SCIRXSHF are moved into SCIRD, and the software must read SCIRD and determine if the SCI is being addressed by comparing the received address against the address previously set in the software and stored somewhere in memory (the SCI does not have hardware available for address comparison). If the SCI is being addressed, the software must clear the SLEEP bit so that the SCI will load SCIRD with the data of the data frames that follow the address frame.

When the SCI has been addressed and sleep mode has been disabled (in software) to allow the receipt of data, the SCI should check the RXWAKE bit (SCIFLR.12) to determine when the next address has been received. This bit is set to 1 if the current value in SCIRD is an address and set to 0 if SCIRD contains data. If the RXWAKE bit is set, then software should check the address in SCIRD against its own address. If it is still being addressed, then sleep mode should remain disabled. Otherwise, the SLEEP bit should be set again.

Following is a sequence of events typical of sleep mode operation:

- The SCI is configured and both sleep mode and receive actions are enabled.
- An address frame is received and a receive interrupt is generated.
- Software compares the received address frame against that set by software and determines that the SCI is not being addressed, so the value of the SLEEP bit is not changed.
- Several data frames are shifted into SCIRXSHF, but no data is moved to SCIRD and no receive interrupts are generated.
- A new address frame is received and a receive interrupt is generated.
- Software compares the received address frame against that set by software and determines that the SCI is being addressed and clears the SLEEP bit.
- Data shifted into SCIRXSHF is transferred to SCIRD, and a receive interrupt is generated after each data frame is received.
- In each interrupt routine, software checks RXWAKE to determine if the current frame is an address frame.
- Another address frame is received, RXWAKE is set, software determines that the SCI is not being addressed and sets the SLEEP bit back to 1. No receive interrupts are generated for the data frames following this address frame.

By ignoring data frames that are not intended for the device, fewer interrupts are generated. These interrupts would otherwise require CPU intervention to read data that is of no significance to this specific device. Using sleep mode can help free some CPU resources.

Except for the RXRDY flag, the SCI continues to update the receiver status flags (see Table 28-12) while sleep mode is active. In this way, if an error occurs on the receive line, an application can immediately respond to the error and take the appropriate corrective action.

Because the RXRDY bit is not updated for data frames when sleep mode is enabled, the SCI can enable sleep mode and use a polling algorithm if desired. In this case, when RXRDY is set, software knows that a new address has been received. If the SCI is not being addressed, then the software should not change the value of the SLEEP bit and should continue to poll RXRDY.

28.7 LIN Communication Formats

The SCI/LIN module can be used in LIN mode or SCI mode. The enhancements for baud generation, DMA controls and additional receive/transmit buffers necessary for LIN mode operation are also part of the enhanced buffered SCI module. LIN mode is selected by enabling LIN MODE bit in SCIGCR1 register.

NOTE: The SCI/LIN is built around the SCI platform and uses a similar sampling scheme: 16 samples for each bit with majority vote on samples 8, 9, and 10.

The SCI/LIN control registers are located at the SCI/LIN base address. For a detailed description of each register, see Section 28.13.

28.7.1 LIN Standards

For compatibility with LIN2.0 standard the following additional features are implemented over LIN1.3:

i. Support for LIN 2.0 checksum

ii. Enhanced synchronizer FSM support for frame processing

iii. Enhanced handling of extended frames

iv. Enhanced baudrate generator

v. Update wakeup/go to sleep

The LIN module covers the CPU performance-consuming features, defined in the *LIN Specification Package* Revision 1.3 and 2.0 by hardware.

The Master Mode of LIN module is compatible with LIN 2.1 standard.

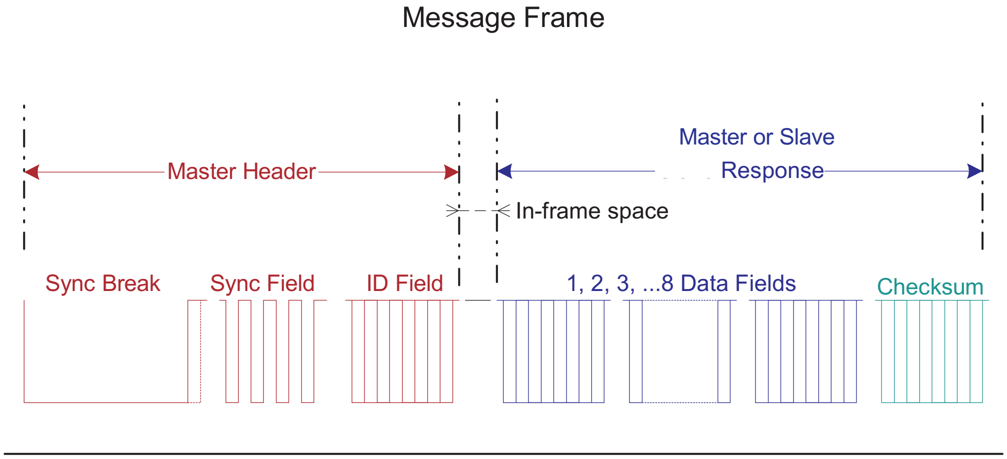
28.7.2 Message Frame

The LIN protocol defines a message frame format, illustrated in Figure 28-12. Each frame includes one master header, one response, one in-frame response space, and inter-byte spaces. In-frame-response and inter-byte spaces may be 0.

There is no arbitration in the definition of the LIN protocol; therefore, multiple slave nodes responding to a header might be detected as an error.

The LIN bus is a single channel wired-AND. The bus has a binary level: either dominant for a value of 0, or recessive for a value of 1.

Figure 28-12. LIN Protocol Message Frame Format: Master Header and Slave Response

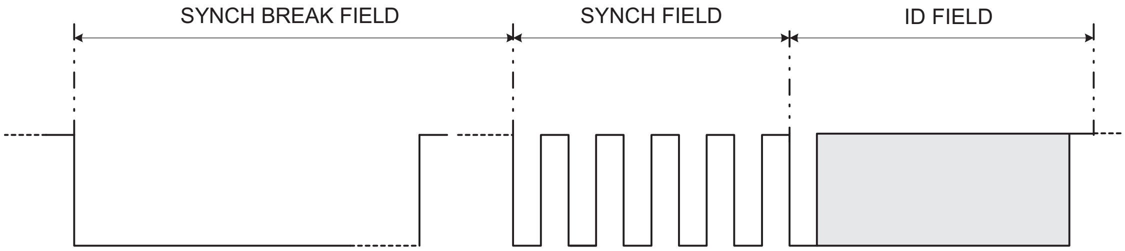


28.7.2.1 Message Header

The header of a message is initiated by a master (see Figure 28-13) and consists of a three field-sequence:

- The synch break field signaling the beginning of a message
- The synch field conveying bit rate information of the LIN bus
- The ID field denoting the content of a message

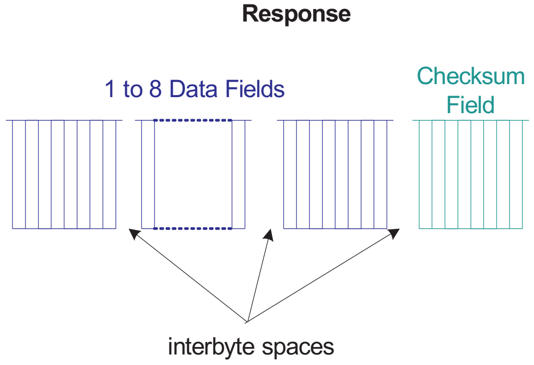
Figure 28-13. Header 3 Fields: Synch Break, Synch, and ID



28.7.2.2 Response

The format of the response is as illustrated in Figure 28-14. There are two types of fields in a response: data and checksum. The data field consists of exactly one data byte, one start bit, and one stop bit, for a total of 10 bits. The LSB is transmitted first. The checksum field consists of one checksum byte, one start bit and one stop bit. The checksum byte is the inverted modulo-256 sum over all data bytes in the data fields of the response.

Figure 28-14. Response Format of LIN Message Frame



The format of the response is a stream of N data fields and one checksum field. Typically N is from 1 to 8, with the exception of the extended command frames (Section 28.7.6). The length N of the response is indicated either with the optional length control bits of the ID Field (this is used in standards earlier than LIN 1.x); see Table 28-5, or by LENGTH value in SCIFORMAT[18:16] register; see Table 28-6. The SCI/LIN module supports response lengths from 1 to 8 bytes in compliance with LIN 2.0.

Table 28-5. Response Length Info Using IDBYTE Field Bits [5:4] for LIN Standards Earlier than 1.3

ID5	ID4	Number of Data bytes
0	0	2
0	1	2
1	0	4
1	1	8

Table 28-6. Response Length with SCIFORMAT[18:16] Programming

SCIFORMAT[18:16]	No. of Bytes
000	1
001	2
010	3
011	4
100	5
101	6
110	7
111	8

28.7.3 Synchronizer

The synchronizer has three major functions in the messaging between master and slave nodes. It generates the master header data stream, it synchronizes to the LIN bus for responding, and it locally detects timeouts. A bit rate is programmed using the prescalers in the BRS register to match the indicated LIN_speed value in the LIN description file.

The LIN synchronizer will perform the following functions: master header signal generation, slave detection and synchronization to message header with optional baud rate adjustment, response transmission timing and timeout control.

The LIN synchronizer is capable of detecting an incoming break and initializing communication at all times.

28.7.4 Baud Rate

The transmission baud rate of any node is configured by the CPU at the beginning; this defines the bit time T_{bit}. The bit time is derived from the fields P and M in the baud rate selection register (BRS). There is an additional 3-bit fractional divider value, field U in the BRS register, which further fine tunes the data field baud rate.

The ranges for the prescaler values in the BRS are:

$P = 0, 1, 2, 3, . . . , 2^{24} - 1$

$M = 0, 1, 2, . . . , 15$

$U = 0, 1, 2, 3, 4, 5, 6, 7$

The P, M, and U values in the BRS are user programmable. The P and M dividers could be used for both SCI mode and LIN mode to select a baud rate. The U value is an additional 3-bit value determining that “**a TVCLK**“ (with **a** = 0,1) is added to each T_{bit} as explained in Section 28.7.4.2. If the ADAPT bit is set and the LIN slave is in adaptive baud rate mode, then all these divider values are automatically obtained during header reception when the synchronization field is measured.

The LIN protocol defines baud rate boundaries as follows:

$1kHz \le F_{LINCLK} \le 20kHz$

All transmitted bits are shifted in and out at T_{bit} periods.

28.7.4.1 Fractional Divider

The M field in the BRS modifies the integer prescaler P for fine tuning of the baud rate. The M value adds in increments of 1/16 of the P value.

The bit time, T_{bit} is expressed in terms of the VCLK period T_{VCLK} as follows:

For all P other than 0, and all M,

$$T_{bit} = 16\left(P + 1 + \frac{M}{16}\right)T_{VCLK} \quad (45)$$

For P= 0 : $T_{bit} = 32T_{VCLK}$

Therefore, the LINCLK frequency is given by:

$$F_{LINCLK} = \frac{F_{VCLK}}{16(P + 1 + \frac{M}{16})} \quad \text{For all P other than zero}$$

$$F_{LINCLK} = \frac{F_{VCLK}}{32} \quad \text{For P = 0} \quad (46)$$

28.7.4.2 Superfractional Divider

The superfractional divider scheme applies to the following modes:

- LIN master mode (synch field + identifier field + response field + checksum field)
- LIN slave mode (response field + checksum field)

28.7.4.3 Superfractional Divider In LIN Mode

Building on the 4-bit fractional divider M (BRS[27:24], the super fractional divider uses an additional 3-bit modulating value, illustrated in Table 28-7. The sync field (0x55), the identifier field and the response field can all be seen as 8-bit data bytes flanked by a start bit and a stop bit. The bits with a 1 in the table will have an additional VCLK period added to their T_{bit}.

Table 28-7. Superfractional Bit Modulation for LIN Master Mode and Slave Mode[1]

BRS[30:28]	Start Bit	D[0]	D[1]	D[2]	D[3]	D[4]	D[5]	D[6]	D[7]	Stop Bit
0h	0	0	0	0	0	0	0	0	0	0
1h	1	0	0	0	0	0	0	0	1	0
2h	1	0	0	0	1	0	0	0	1	0
3h	1	0	1	0	1	0	0	0	1	0
4h	1	0	1	0	1	0	1	0	1	0
5h	1	1	1	0	1	0	1	0	1	1
6h	1	1	1	0	1	1	1	0	1	1
7h	1	1	1	1	1	1	1	0	1	1

[1]

1. In LIN master mode, bit modulation applies to synch field + identifier field + response field
2. In LIN slave mode, bit modulation applies to identifier field + response field

The baud rate will vary over a LIN data field to average according to the BRS[30:28] value by a *d* fraction of the peripheral internal clock: 0<d<1.

The instantaneous bit time is expressed in terms of T_{VCLK} as follows:

For all P other than 0, and all M and d (0 or 1),

$$T^{i}bit = \left[16\left(P+1+\frac{M}{16}\right)+d\right]T_{VCLK} \tag{47}$$

For P = 0, $T_{bit} = 32T_{VCLK}$

The averaged bit time is expressed in terms of T_{VCLK} as follows:

For all P other than 0, and all M and d (0<d<1),

$$T^{a}bit = \left[16\left(P+1+\frac{M}{16}\right)+d\right]T_{VCLK} \tag{48}$$

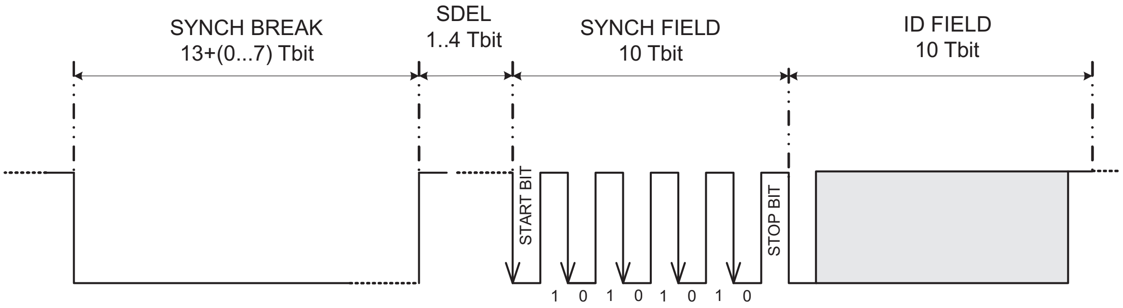
For P = 0, $T_{bit} = 32T_{VCLK}$

With the superfractional divider, a LIN baud rate of 20 kbps is achievable with an internal clock VCLK of 726 kHz. Furthermore, a rate of 400 kbps is achievable with an VCLK of 14.6 MHz.

28.7.5 Header Generation

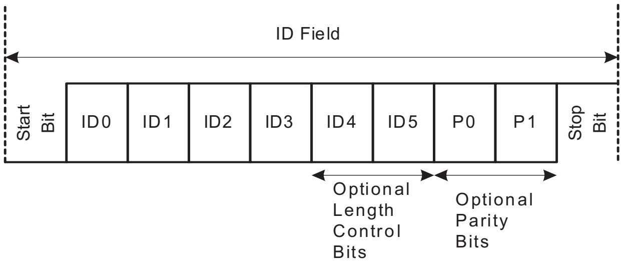
Automatic generation of the LIN protocol header data stream is supported without CPU interaction. The CPU pr the DMA will trigger a message header generation and the LIN state machine will handle the generation itself. A master node initiates header generation on CPU or DMA writes to the IDBYTE in the LINID register. The header is always sent by the master to initiate a LIN communication and consists of three fields: break field, synchronization field, and identification field, as seen in Figure 28-15.

Figure 28-15. Message Header in Terms of T_{bit}



- The break field consists of two components:
  - The synchronization break (SYNCH BREAK) consists of a minimum of 13 (dominant) low bits to a maximum of 20 dominant bits. The synch break length may be extended from the minimum with the 3-bit SBREAK value in the LINCOMP register.
  - The synchronization break delimiter (SDEL) consists of a minimum of 1 (recessive) high bit to a maximum of 4 recessive bits. The delimiter marks the end of the synchronization break field. The synch break delimiter length depends on the 2-bit SDEL value in the LINCOMP register.
- The synchronization field (SYNCH FIELD) consists of one start bit, byte 0x55, and a stop bit. It is used to convey T_{bit} information and resynchronize LIN bus nodes.
- The identifier field's ID byte may use six bits as an identifier, with optional length control (see *Note:Optional Control Length Bits*), and two optional bits as parity of the identifier. The identifier parity is used and checked if the PARITY ENA bit is set. If length control bits are not used, then there can be a total of 64 identifiers plus parity. If neither length control or parity are used there can be up to 256 identifiers. See Figure 28-16 for an illustration of the ID field.

Figure 28-16. ID Field



NOTE: Optional Control Length Bits

The control length bits only apply to LIN standards prior to LIN 1.3. IDBYTE field conveys response length information if compliant to standards earlier than LIN1.3. The SCIFORMAT register stores the length of the response for later versions of the LIN protocol.

NOTE: If the BLIN module, configured as Slave in multi-buffer mode, is in the process of transmitting data while a new header comes in, the module might end up in responding with the data from the previous interrupted response (not the data corresponding to the new ID). To avoid this scenario the following procedure could be used:

1. Check for the Bit Error (BE) during the response transmission. If the BE flag is set, this indicates that a collision has happened on the LIN bus (here because of the new Synch Break).
2. In the Bit Error ISR, configure the TD0 and TD1 registers with the next set of data to be transmitted on a TX Match for the incoming ID. Before writing to TD0/TD1 make sure that there was not already an update because of a Bit Error; otherwise TD0/TD1 might be written twice for one ID.
3. Once the complete ID is received, based on the match, the newly configured data will be transmitted by the node.

28.7.5.1 Event Triggered Frame Handling Proposal

The LIN 2.0 protocol uses event-triggered frames that may occasionally cause collisions. Event-triggered frames have to be handled in software.

If no slave answers to an event triggered frame header, the master node will set the NRE flag, and a NRE interrupt will occur if enabled. If a collision occurs, a frame error and checksum error may arise before the NRE error. Those errors are flagged and the appropriate interrupts will occur, if enabled.

Frame errors and checksum errors depend on the behavior and synchronization of the responding slaves. If the slaves are totally synchronized and stop transmission once the collision occurred, it is possible that only the NRE error is flagged despite the occurrence of a collision. To detect if there has been a reception of one byte before the NRE error is flagged, the BUS BUSY flag can be used as an indicator.

The bus busy flag is set on the reception of the first bit of the header and remains set until the header reception is complete, and again is set on the reception of the first bit of the response. In the case of a collision the flag is cleared in the same cycle as the NRE flag is set.

Software could implement the following sequence:

- Once the reception of the header is done (poll for RXID flag), wait for the bus busy flag to get set or NRE flag to get set.
- If bus busy flag is not set before NRE flag, then it is a true no response case (no data has been transmitted onto the bus).
- If bus busy flag gets set, then wait for NRE flag to get set or for successful reception. If NRE flag is set, then in this case a collision has occurred on the bus.

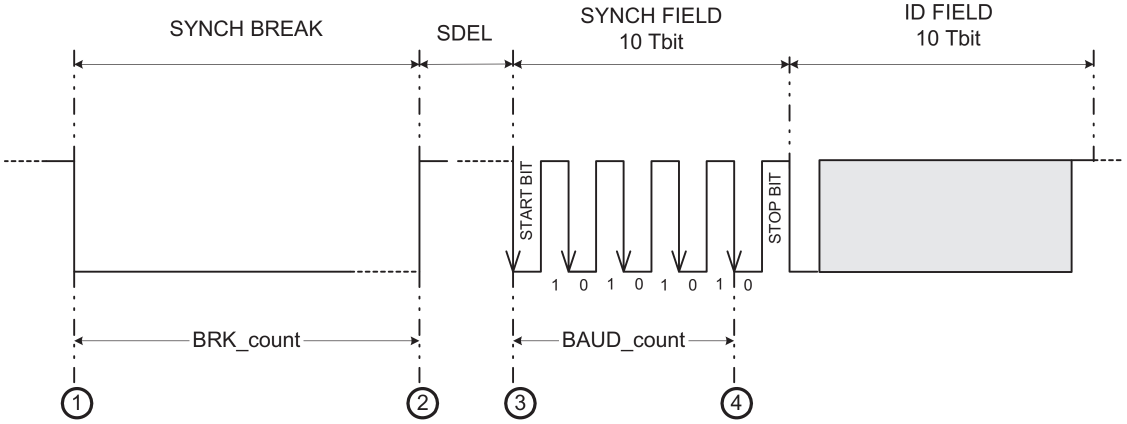
Even in the case of a collision, the received (corrupted) data is accessible in the RX buffers; registers LINRD0 and LINRD1.

28.7.5.2 Header Reception and Adaptive Baud Rate

A slave node baud rate can optionally be adjusted to the detected bit rate as an option to the LIN module. The adaptive baud rate option is enabled by setting the ADAPT bit. During header reception, a slave measures the baud rate during detection of the synch field. If ADAPT bit is set, then the measured baud rate is compared to the slave node's programmed baud rate and adjusted to the LIN bus baud rate if necessary.

The LIN synchronizer determines two measurements: BRK_count and BAUD_count (Figure 28-17). These values are always calculated during the Header reception for synch field validation (Figure 28-18).

Figure 28-17. Measurements for Synchronization



By measuring the values BRK_count and BAUD_count, a valid synch break sequence can be detected as described in Figure 28-18. The four numbered events in Figure 28-17 signal the start/stop of the synchronizer counter. The synchronizer counter uses VCLK as the time base.

The synchronizer counter is used to measure the synch break relative to the detecting node T_{bit}. For a slave node receiving the synch break, a threshold of 11 T_{bit} is used as required by the LIN protocol. For detection of the dominant data stream of the synch break, the synchronizer counter is started on a falling edge and stopped on a rising edge of the LINRX. On detection of the synch break delimiter, the synchronizer counter value is saved and then reset.

On detection of five consecutive falling edges, the BAUD_count is measured. Bit timing calculation and consistency to required accuracy is implemented following the recommendations of LIN revision 2.0. A slave node can calculate a single T_{bit} time by division of BAUD_count by 8. In addition, for consistency between the detected edges the following is evaluated:

BAUD_count + BAUD_count » 2 + BAUD_count » 3 ≤ BRK_count

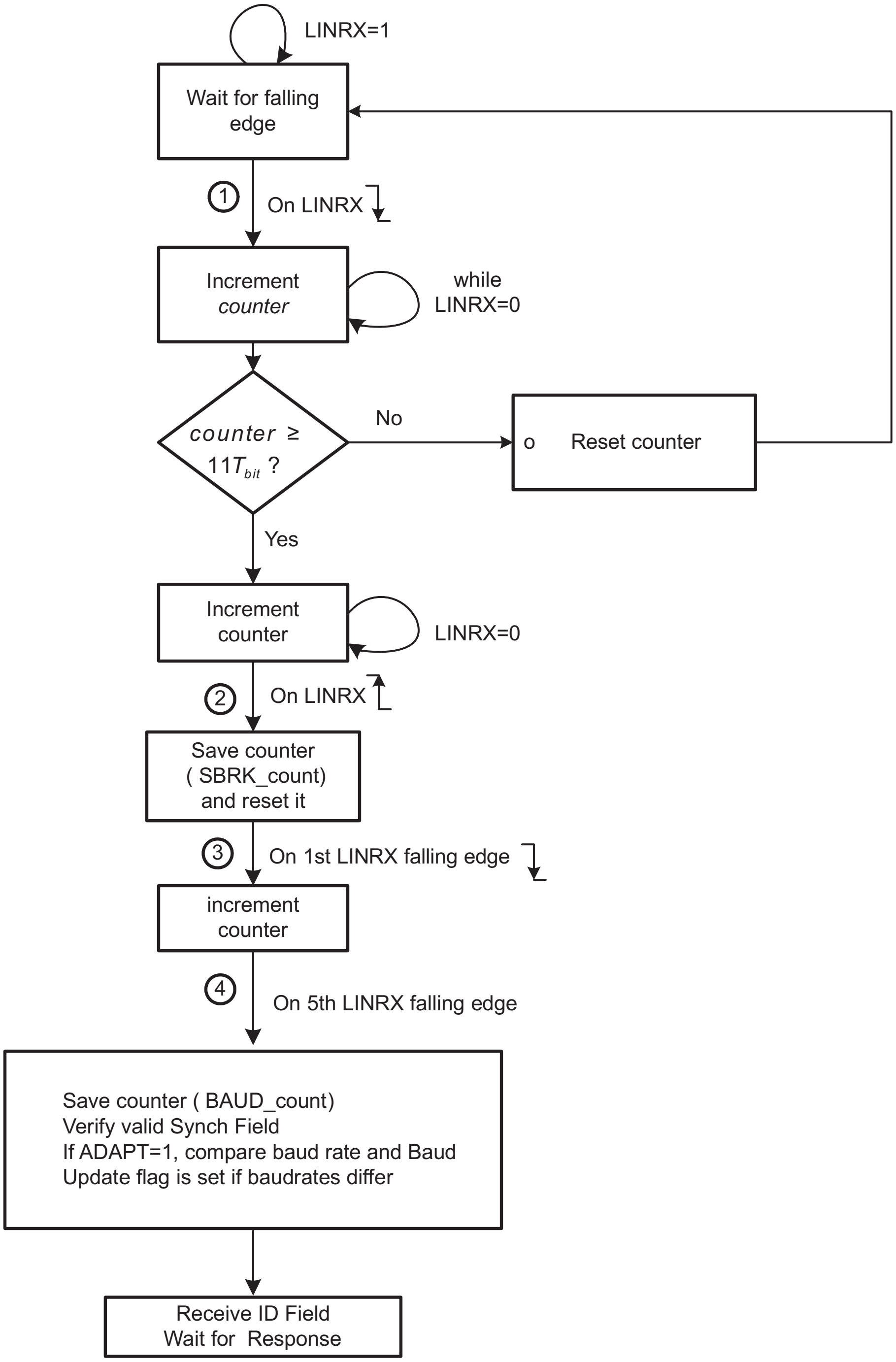
The BAUD_count value is shifted 3 times to the right and rounded using the first insignificant bit to obtain a T_{bit} unit. If the ADAPT bit is set, then the detected baud rate is compared to the programmed baud rate.

During the header reception processing as illustrated in Figure 28-18, if the measured BRK_count value is less than 11 T_{bit}, the synch break is not valid according to the protocol for a fixed rate. If the ADAPT bit is set, then the MBRS register is used for measuring BRK_count and BAUD_count values and automatically adjusts to any allowed LIN bus rate (refer to *LIN Specification Package 2.0*).

NOTE: In adaptive mode, the MBRS divider should be set to allow a maximum baud rate that is not more than 10% above the expected operating baud rate in the LIN network. Otherwise, a 0x00 data byte could mistakenly be detected as a synch break.

The break-threshold relative to the slave node is 11 T_{bit}. The break is 13 T_{bit} as specified in LIN version 1.3.

Figure 28-18. Synchronization Validation Process and Baud Rate Adjustment


LINRX=1
Wait for falling edge
1
On LINRX
Increment *counter*
while LINRX=0
counter ≥ 11T_{bit} ?
No
o Reset counter
Yes
Increment counter
LINRX=0
2
On LINRX
Save counter (SBRK_count) and reset it
3
On 1st LINRX falling edge
increment counter
4
On 5th LINRX falling edge
Save counter (BAUD_count)
Verify valid Synch Field
If ADAPT=1, compare baud rate and Baud Update flag is set if baudrates differ
Receive ID Field
Wait for Response

If the synch field is not detected within the given tolerances, the inconsistent-synch-field-error (ISFE) flag will be set. An ISFE interrupt will be generated, if enabled by its respective bit in the SCISETINT register. The ID byte should be received after the synch field validation was successful. Any time a valid break (larger than 11 T_{bit}) is detected, the receiver's state machine should reset to reception of this new frame. This reset condition is only valid during response state, not if an additional synch break occurs during header reception.

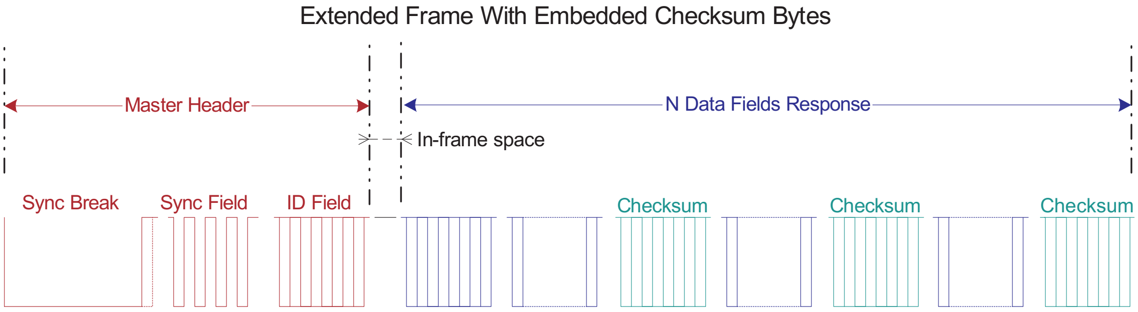
NOTE: When an inconsistent synch field (ISFE) error occurs, suggested action for the application is to reset the SWnRST bit and set the SWnRST bit to make sure that the internal state machines are back to their normal states

28.7.6 Extended Frames Handling

The LIN protocol 2.0 and prior includes two extended frames with identifiers 62 (user defined) and 63 (reserved extended). The response data length of the user-defined frame (ID 62, or 0x3E) is unlimited. The length for this identifier will be set at network configuration time to be shared with the LIN bus nodes.

Extended frame communication is triggered on reception of a header with identifier 0x3E; see Figure 28-19. Once the extended frame communication is triggered, unlike normal frames, this communication needs to be stopped before issuing another header. To stop the extended frame communication the STOP EXT FRAME bit must be set.

Figure 28-19. Optional Embedded Checksum in Response for Extended Frames



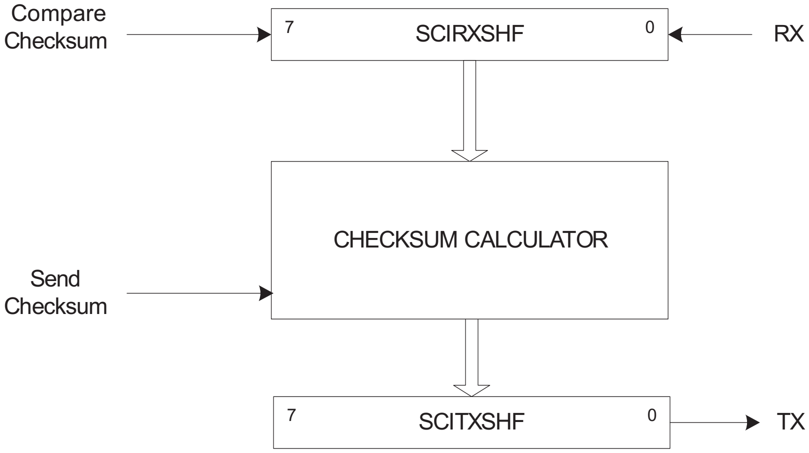
An ID interrupt will be generated (if enabled and there is a match) on reception of ID 62 (0x3E). This interrupt allows the CPU using a software counter to keep track of the bytes that are being sent out and decides when to calculate and insert a checksum byte (recommended at periodic rates). To handle this procedure, SC bit is used. A write to the send checksum bit SC will initiate an automatic send of the checksum byte. The last data field should always be a checksum in compliance with the LIN protocol.

The periodicity of the checksum insertion, defined at network configuration time, is used by the receiving node to evaluate the checksum of the ongoing message, and has the benefit of enhanced reliability.

For the sending node, the checksum is automatically embedded each time the send checksum bit SC is set. For the receiving node, the checksum is compared each time the compare checksum bit CC is set; see Figure 28-20.

NOTE: The LIN 2.0 enhanced checksum does not apply to the reserved identifiers. The reserved identifiers always use the classic checksum.

Figure 28-20. Checksum Compare and Send for Extended Frames



28.7.7 Timeout Control

Any LIN node listening to the bus and expecting a response initiated from a master node could flag a no-response error timeout event. The LIN protocol defines four types of timeout events, which are all handled by the hardware of the LIN module. The four LIN protocol events are:

- No-response timeout error
- Bus idle detection
- Timeout after wakeup signal
- Timeout after three wakeup signals

28.7.7.1 No-Response Error (NRE)

The no-response error will occur when any node expecting a response waits for T_{FRAME_MAX} time and the message frame is not fully completed within the maximum length allowed, T_{FRAME_MAX}. After this time a no-response error (NRE) is flagged in the NRE bit of the SCIFLR register. An interrupt is triggered if enabled.

As specified in the LIN 1.3 standard, the minimum time to transmit a frame is:

$T_{FRAME\_MIN} = T_{HEADER\_MIN} + T_{DATA\_FIELD} + T_{CHECKSUM\_FIELD}$

$= 44 + 10N$

where N = number of data fields.

And the maximum time frame is given by:

$T_{FRAME\_MAX} = T_{FRAME\_MIN} * 1.4$

$= (44 + 10N) * 1.4$

The timeout value T_{FRAME_MAX} is derived from the *N* number of data fields value. The *N* value is either embedded in the header's ID field for messages or is part of the description file. In the latter case, the 3-bit CHAR value in SCIFORMAT register, will indicate the value for *N*.

NOTE: The length coding of the ID field does not apply to two extended frame identifiers, ID fields of 0x3E (62) and 0x3F (63). In these cases, the ID field can be followed by an arbitrary number of data byte fields. Also, the LIN 2.0 protocol specification mentions that ID field 0x3F (63) cannot be used. For these two cases, the NRE will not be handled by the LIN controller hardware.

Table 28-8. Timeout Values in T_{bit} Units

N	T_{DATA_FIELD}	T_{FRAME_MIN}	T_{FRAME_MAX}
1	10	54	76
2	20	64	90
3	30	74	104
4	40	84	118
5	50	94	132
6	60	104	146
7	70	114	160
8	80	124	174

28.7.7.2 Bus Idle Detection

The second type of timeout can occur when a node detects an inactive LIN bus: no transitions between recessive and dominant values are detected on the bus. This happens after a minimum of 4 s (this is 80,000 F_{LINCLK} cycles with the fastest bus rate of 20 kbps). If a node detects no activity in the bus as the TIMEOUT bit is set, then it can be assumed that the LIN bus is in sleep mode. Application software can use the Timeout flag to determine when the LIN bus is inactive and put the LIN into sleep mode by writing the POWERDOWN bit.

NOTE: After the timeout was flagged, a software RESET should be asserted before entering Low-Power Mode. This is required to reset the receiver in case that an incomplete frame was on the bus before the idle period.

28.7.7.3 Timeout after Wakeup Signal and Timeout after Three Wakeup Signals

The third and fourth types of timeout are related to the wakeup signal. A node initiating a wakeup should expect a header from the master within a defined amount of time: timeout after wakeup signal. See Section 28.11.3 for more details.

28.7.8 TXRX Error Detector (TED)

The following sources of error are detected by the TXRX error detector logic (TED). The TED logic consists of a bit monitor, an ID parity checker, and a checksum error. The following errors are detected:

- Bit errors (BE)
- Physical bus errors (PBE)
- Identifier parity errors (PE)
- Checksum errors (CE)

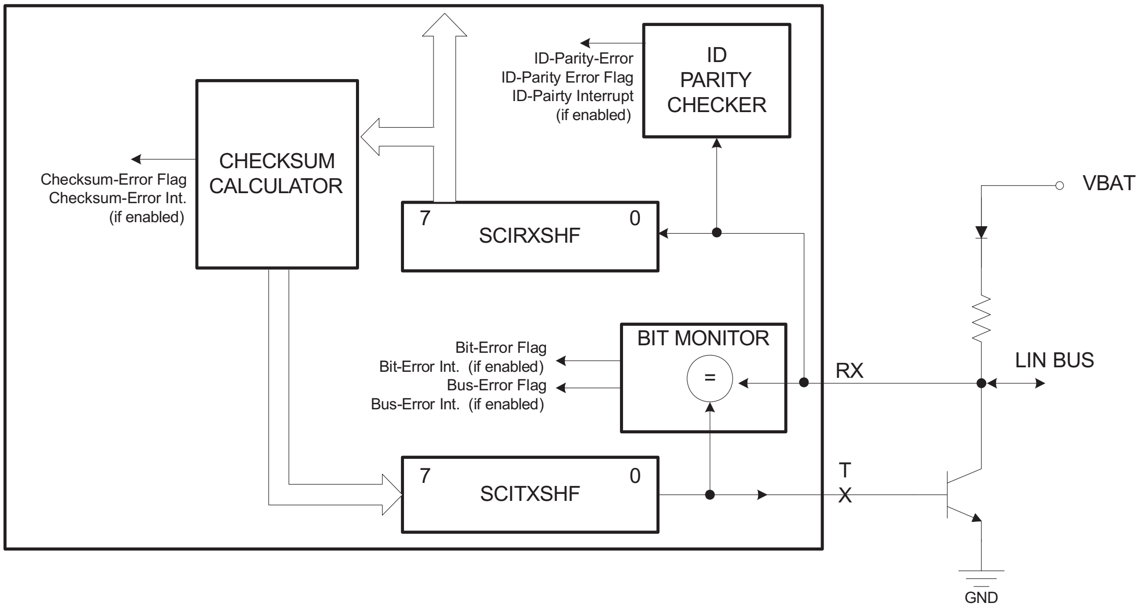
All of these errors (BE, PBE, PE, CE) are flagged. An interrupt for the flagged errors will be generated if enabled. A message is valid for both the transmitter and the receiver if there is no error detected until the end of the frame.

28.7.8.1 Bit Errors

A bit error (BE) is detected at the bit time when the bit value that is monitored is different from the bit value that is sent. A bit error is indicated by the BE flag in SCIFLR. After signaling a BE, the transmission is aborted no later than the next byte. The bit monitor ensures that the transmitted bit in LINTX is the correct value on the LIN bus by reading back on the LINRX pin as shown in Figure 28-21.

NOTE: If BE Occurs due to New Header reception during a Slave Response, NRE/TIMEOUT flag will not be set for the new Frame.

Figure 28-21. TXRX Error Detector



28.7.8.2 Physical Bus Errors

A Physical Bus Error (PBE) has to be detected by a master if no valid message can be generated on the bus (Bus shorted to GND or VBAT). The bit monitor detects a PBE during the header transmission, if no Synch Break can be generated (for example, because of a bus shortage to VBAT) or if no Synch break Delimiter can be generated (for example, because of a bus shortage to GND). Once the Synch Break Delimiter was validated, all other deviations between the monitored and the sent bit value are flagged as Bit Errors (BE) for this frame.

28.7.8.3 ID Parity Errors

If parity is enabled, an ID parity error (PE) is detected if any of the two parity bits of the sent ID byte are not equal to the calculated parity on the receiver node. The two parity bits are generated using the following mixed parity algorithm.

$$P0 = ID0 \oplus ID1 \oplus ID2 \oplus ID4 \ (even \ \ Parity)$$
$$P1 = ID1 \oplus ID3 \oplus ID4 \oplus ID5 \ (odd \ \ Parity) \quad (49)$$

If an ID-parity error is detected, the ID-parity error is flagged, and the received ID is not valid. See Section 28.7.9 for details.

28.7.8.4 Checksum Errors

A checksum error (CE) is detected and flagged at the receiving end if the calculated modulo-256 sum over all received data bytes (including the ID byte if it is the enhanced checksum type) plus the checksum byte does not result in 0xFF. The modulo-256 sum is calculated over each byte by adding with carry, where the carry bit of each addition is added to the LSB of its resulting sum.

For the transmitting node, the checksum byte sent at the end of a message is the inverted sum of all the data bytes (see Figure 28-22) for classic checksum implementation. The checksum byte is the inverted sum of the identifier byte and all the data bytes (see Figure 28-23) for the LIN 2.0 compliant enhanced checksum implementation. The classic checksum implementation should always be used for reserved identifiers 60 to 63; therefore, the CTYPE bit will be overridden in this case. For signal-carrying-frame identifiers (0 to 59) the type of checksum used depends on the CTYPE bit.

Figure 28-22. Classic Checksum Generation at Transmitting Node

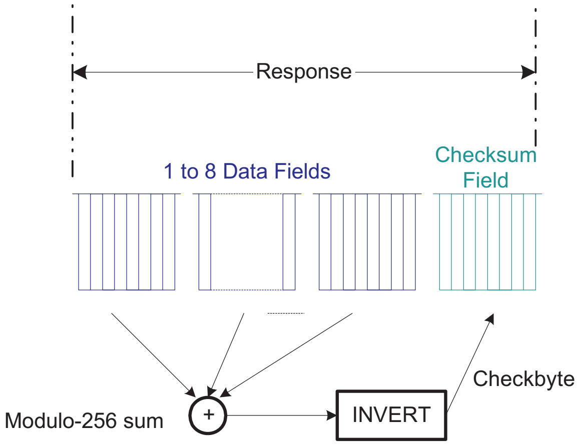
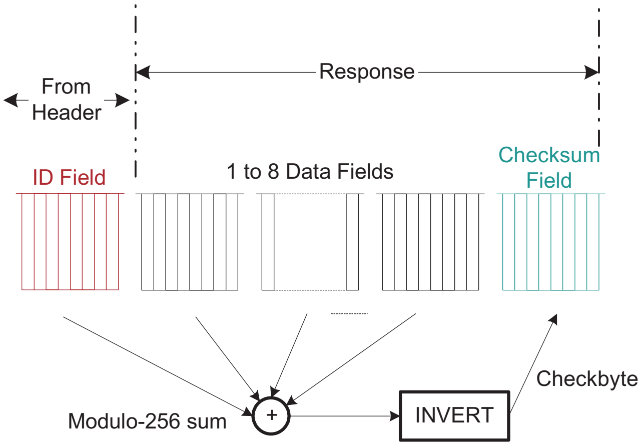


Figure 28-23. LIN 2.0-Compliant Checksum Generation at Transmitting Node



28.7.9 Message Filtering and Validation

Message filtering uses the entire identifier to determine which nodes will participate in a response, either receiving or transmitting a response. Therefore, two acceptance masks are used, as shown in Figure 28-24. During header reception, all nodes filter the ID-Field (ID-Field is the part of the header explained in Figure 28-16) to determine whether they transmit a response or receive a response for the current message. There are two masks for message ID filtering: one to accept a response reception, the other to initiate a response transmission. See Figure 28-24. All nodes compare the received ID to the identifier stored in the ID-SlaveTask BYTE of the LINID register and use the RX ID MASK and the TX ID MASK fields in the LINMASK register to filter the bits of the identifier that should not be compared.

If there is an RX match with no parity error and the RXENA bit is set, there will be an ID RX flag and an interrupt will be triggered if enabled. If there is a TX match with no parity error and the TXENA bit is set, there will be an ID TX flag and an interrupt will be triggered if enabled in the SCISETINT register.

The masked bits become don’t cares for the comparison. To build a mask for a set of identifiers, an XOR function could be used.

For example, to build a mask to accept IDs 0x26 and 0x25 using LINID[7:0] = 0x20; that is, compare 5 most significant bits (MSBs) and filter 3 least significant bits (LSBs), the acceptance mask could be:

$$(0x26 + 0x25) \oplus 0x20 = 0x07 \tag{50}$$

A mask of all zeros will compare all bits of the received identifier in the shift register with the ID-BYTE in LINID[7:0]. If HGEN CTRL is set to 1, a mask of 0xFF will always cause a match. A mask of all 1s will filter all bits of the received identifier, and thus there will be an ID match regardless of the content of the ID-SlaveTask BYTE field in the LINID register.

NOTE: When the HGEN CTRL bit = 0, the LIN nodes compare the received ID to the ID-BYTE field in the LINID register, and use the RX ID MASK and the TX ID MASK in the LINMASK register to filter the bits of the identifier that should not be compared.

If there is an RX match with no parity error and the RXENA bit is set, there will be an ID RX flag and an interrupt will be triggered if enabled. A mask of all 0s will compare all bits of the received identifier in the shift register with the ID-BYTE field in LINID[7:0]. A mask of all 1s will filter all bits of the received identifier and there will be no match.

During header reception, the received identifier is copied to the Received ID field LINID[23:16]. If there is no parity error and there is either a TX match or an RX match, then the corresponding TX or RX ID flag is set. If the ID interrupt is enabled, then an ID interrupt is generated.

After the ID interrupt is generated, the CPU may read the Received ID field LINID[23:16] and determine what response to load into the transmit buffers.

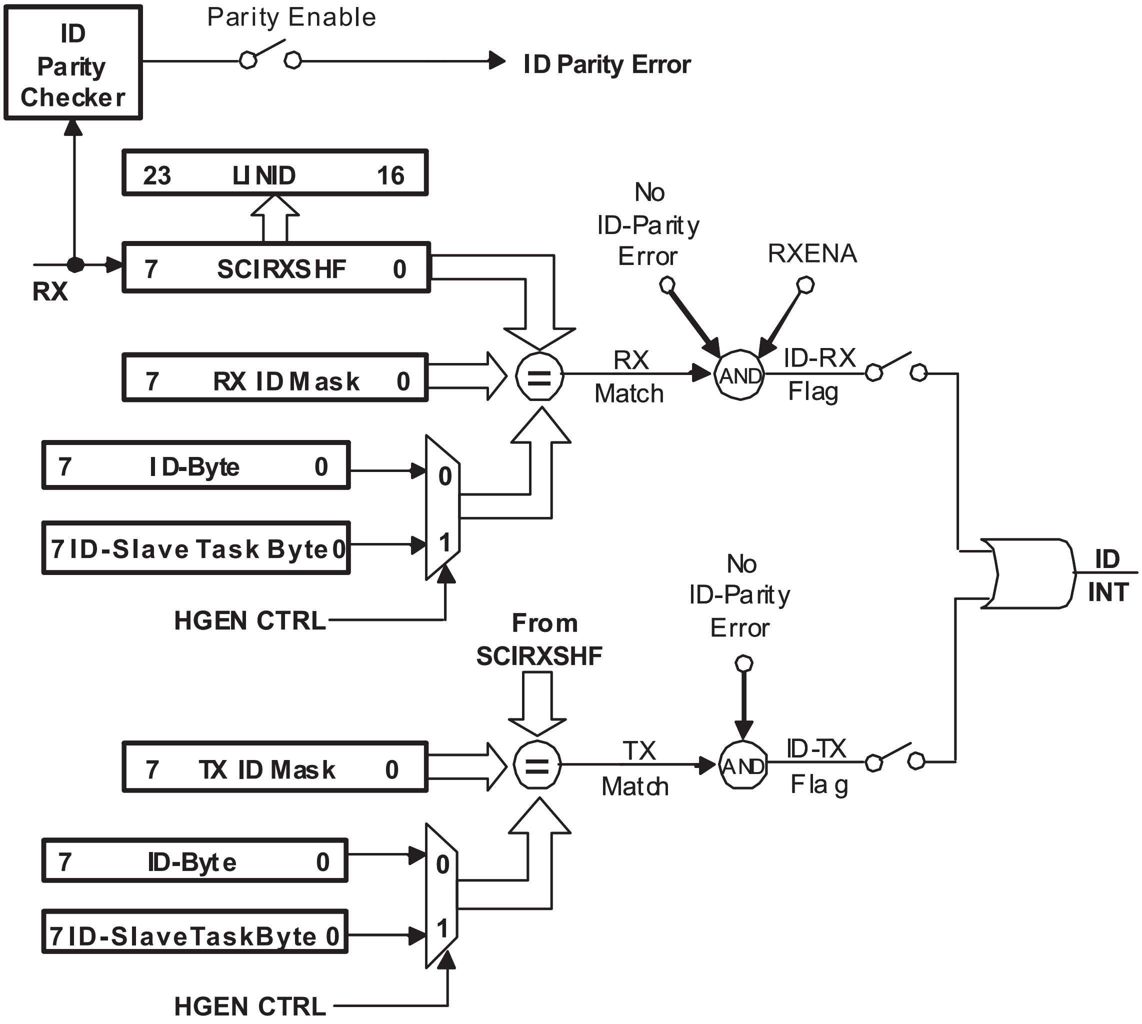
NOTE: When byte 0 is written to TD0 (LINTD0[31:24]), the response transmission is automatically generated.

In multi-buffer mode, the TXRDY flag will be set when all the response data bytes and checksum byte are copied to the shift register SCITXSHF. In non-multi-buffer mode, the TXRDY flag is set each time a byte is copied to the SCITXSHF register, and also for the last byte of the frame after the checksum byte is copied to the SCITXSHF register.

In multi-buffer mode, the TXEMPTY flag is set when both the transmit buffer(s) TDy and the SCITXSHF shift register are emptied and the checksum has been sent. In non multi-buffer mode, TXEMPTY is set each time TD0 and SCITXSHF are emptied, except for the last byte of the frame where the checksum byte must also be transmitted.

If parity is enabled, all slave receiving nodes will validate the identifier using all eight bits of the received ID byte. The SCI/LIN will flag a corrupted identifier if an ID-parity error is detected.

Figure 28-24. ID Reception, Filtering and Validation



28.7.10 Receive Buffers

To reduce CPU load when receiving a LIN N-byte (with N = 1–8) response in interrupt mode or DMA mode, the SCI/LIN module has eight receive buffers. These buffers can store an entire LIN response in the RDy receive buffers.

The checksum byte following the data bytes is validated by the internal checksum calculator. The checksum error (CE) flag indicates a checksum error and a CE interrupt will be generated if enabled in the SCISETINT register.

The multi-buffer 3-bit counter counts the data bytes transferred from the SCIRXSHF register to the RDy receive buffers if multi-buffer mode is enabled, or to RD0 if multi-buffer mode is disabled. The 3-bit compare register contains the number of data bytes expected to be received. In cases where the ID BYTE field does not convey message length (see *Note: Optional Control Length Bits* in Section 28.7.5), the LENGTH value, indicates the expected length and is used to load the 3-bit compare register. Whether the length control field or the LENGTH value is used is selectable with the COMM MODE bit.

Figure 28-8 illustrates the receive buffers.

A receive interrupt, and a receive ready RXRDY flag set as well as a DMA request (RXDMA) could occur after receiving a response if there are no response receive errors for the frame (such as, there is no checksum error, frame error, and overrun error). The checksum byte will be compared before acknowledging a reception. A DMA request can be generated for each received byte or for the entire response depending on whether the multi-buffer mode is enabled or not (MBUF MODE bit in the SCIGCR1 register).

NOTE: In multi-buffer mode, following are the scenarios associated with clearing the RXRDY flag bit:

1. The RXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register.
2. For LENGTH less than or equal to 4, Read to RD0 register will clear the RXRDY flag.
3. For LENGTH greater than 4, Read to RD1 register will clear the RXRDY flag.

28.7.11 Transmit Buffers

To reduce the CPU load when transmitting a LIN N-byte (with N = 1–8) response in interrupt mode or DMA mode, the SCI/LIN module has eight transmit buffers, TD0–TD7 in LINTD0 and LINTD1. With these transmit buffers, an entire LIN response field can be preloaded in the TXy transmit buffers. Optionally, a DMA transfer could be done on a byte-per-byte basis when multi-buffer mode is not enabled (MBUF MODE bit in the SCIGCR1 register).

The multi-buffer 3-bit counter counts the data bytes transferred from the TDy transmit buffers register if multi-buffer mode is enabled, or from TD0 to SCITXSHF if multi-buffer mode is disabled. The 3-bit compare register contains the number of data bytes expected to be transmitted. If the ID field is not used to convey message length (see *Note: Optional Control Length Bits* in Section 28.7.5), the LENGTH value indicates the expected length and is used instead to load the 3-bit compare register. Whether the length control field or the LENGTH value is used is selectable with the COMM MODE bit.

A transmit interrupt (TX interrupt), and a transmit ready flag (TXRDY flag), and a DMA request (TXDMA) could occur after transmitting a response. A DMA request can be generated for each transmitted byte or for the entire response depending on whether multi-buffer mode is enabled or not (MBUF MODE bit in the SCIGCR1 register).

Figure 28-9 illustrates the transmit buffers.

The checksum byte will be automatically generated by the checksum calculator and sent after the data-fields transmission is finished. The multi-buffer 3-bit counter counts the data bytes transferred from the TDy buffers into the SCITXSHF register.

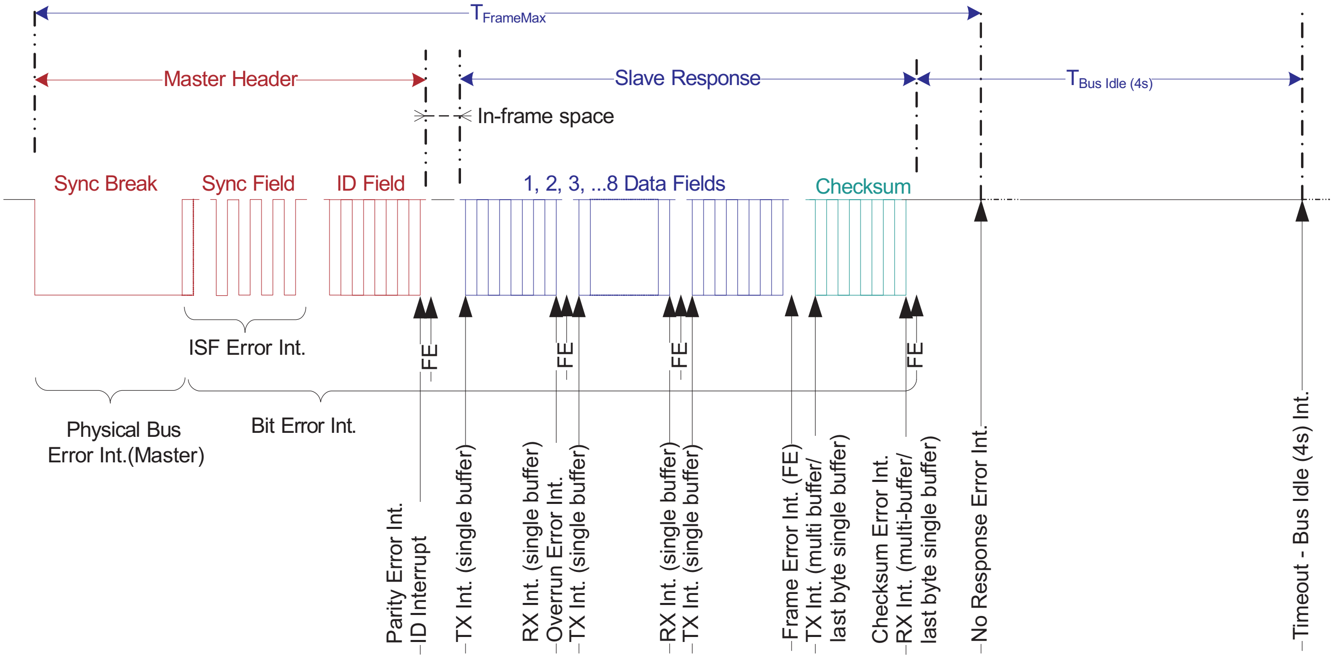
NOTE: The transmit interrupt request can be eliminated until the next series of data is written into the transmit buffers LINTD0 and LINTD1, by disabling the corresponding interrupt via the SCICLRINT register or by disabling the transmitter via the TXENA bit.

28.8 LIN Interrupts

LIN and SCI mode have a common Interrupt block as explained in Section 28.3. There are 16 interrupt sources in the SCI/LIN module, with 8 of them being LIN mode only, as seen in Table 28-4.

A LIN message frame indicating the timing and sequence of the LIN interrupts that could occur is shown in Figure 28-25.

Figure 28-25. LIN Message Frame Showing LIN Interrupt Timing and Sequence



28.9 LIN DMA Interface

LIN DMA Interface uses the SCI DMA interface logic. DMA requests for receive (RXDMA request) and transmit (TXDMA request) are available for the SCI/LIN module. There are two modes for DMA transfers depending on whether multi-buffer mode is enabled or not via the multi-buffer enable control bit (MBUF MODE bit in the SCIGCR1 register).

28.9.1 LIN Receive DMA Requests

In LIN mode, when the multi-buffer option is enabled, if a received response (up to eight data bytes) is transferred to the receive buffers (RDy), then a DMA request is generated. If the multi-buffer option is disabled, then DMA requests will be generated on a byte-per-byte basis until all the expected response data fields are received. This DMA functionality is enabled/disabled by the user using SET RX DMA / CLR RX DMA bits, respectively.

28.9.2 LIN Transmit DMA Requests

In LIN mode with the multi-buffer option enabled, after a transmission (up to eight data bytes stored in the transmit buffer(s) TDy in the LINTD0 and LINTD1 registers), a DMA request is generated in order to reload the transmit buffer for the next transmission. If the multi-buffer option is disabled, then DMA requests will be generated on a byte-per-byte basis until all bytes are transferred. This DMA functionality is enabled/disabled by the user using SET TX DMA / CLR TX DMA bits, respectively.

28.10 LIN Configurations

The following list details the configuration steps that software should perform prior to the transmission or reception of data in LIN mode. As long as the SWnRST bit in the SCIGCR1 register is cleared to 0 the entire time that the LIN is being configured, the order in which the registers are programmed is not important.

- Enable LIN by setting the RESET bit in SCIGCR0 to 1.
- Clear the SWnRST bit to 0 before LIN is configured.
- Select the desired frame format by programming the SCIGCR1 register.
- Set both the RX FUNC and TX FUNC bits in SCIPIO0 to 1 to configure the LINRX and LINTX pins for LIN functionality.
- Set the LIN MODE bit in SCIGCR1 to 1 to enable LIN mode.
- Select Master or Slave mode by programming the CLOCK bit in SCIGCR1.
- Set the MBUF MODE bit in SCIGCR1 to 1 to select multi-buffer mode.
- Select the baud rate to be used for communication by programming the BRS register.
- Set the maximum baud rate to be used for communication by programming the MBRS register.
- Set the CONT bit in SCIGCR1 to 1 to make LIN not halt for an emulation breakpoint until its current reception or transmission is complete (this bit is used only in an emulation environment).
- Set the LOOP BACK bit in SCIGCR1 to 1 to connect the transmitter to the receiver internally (this feature is used to perform a self-test).
- Set the RXENA bit in SCIGCR1 to 1, if data is to be received.
- Set the TXENA bit in SCIGCR1 to 1, if data is to be transmitted.
- Select the RX ID MASK and the TX ID MASK fields in the LINMASK register.
- Set the SWnRST bit to 1 after LIN is configured.
- Perform receiving or transmitting data (see Section 28.10.1 or Section 28.10.2).

28.10.1 Receiving Data

The LIN receiver is enabled to receive messages if both the RX FUNC bit and the RXENA bit are set to 1. If the RX FUNC bit is not set, the LINRX pin functions as a general-purpose I/O pin rather than as a LIN function pin.

The ID RX FLAG is set after a valid LIN ID is received with RX Match. An ID interrupt is generated, if enabled.

28.10.1.1 Receiving Data in Single-Buffer Mode

Single-buffer mode is selected when the MBUF MODE bit is cleared to 0. In this mode, LIN sets the RXRDY bit when it transfers newly received data from SCIRXSHF to RD0. The SCI clears the RXRDY bit after the new data in RD0 has been read. Also, as data is transferred from SCIRXSHF to RD0, the LIN sets the FE, OE, or PE flags if any of these error conditions were detected in the received data. These error conditions are supported with configurable interrupt capability.

You can receive data by:

1. Polling Receive Ready Flag
2. Receive Interrupt
3. DMA

In polling method, software can poll for the RXRDY bit and read the data from RD0 byte of the LINRD0 register once the RXRDY bit is set high. The CPU is unnecessarily overloaded by selecting the polling method. To avoid this, you can use either the interrupt or DMA method. To use the interrupt method, the SET RX INT bit is set. To use the DMA method, the SET RX DMA bit is set. Either an interrupt or a DMA request is generated the moment the RXRDY bit is set. If the checksum scheme is enabled by setting the Compare Checksum (CC) bit to 1, the checksum will be compared on the byte that is currently being received, which is expected to be the checksum byte. The CC bit will be cleared once the checksum is received. A CE will immediately be flagged if there is a checksum error.

28.10.1.2 Receiving Data in Multi-Buffer Mode

Multi-buffer mode is selected when the MBUF MODE bit is set to 1. In this mode, LIN sets the RXRDY bit after receiving the programmed number of data in the receive buffer and the checksum field, the complete frame. The error condition detection logic is similar to the single-buffer mode, except that it monitors for the complete frame. Like single-buffer mode, you can use the polling, interrupt, or DMA method to read the data. The received data has to be read from the LINRD0 and LINRD1 registers, based on the number of bytes. For a LENGTH less than or equal to 4, a read from the LINRD0 register clears the RXRDY flag. For a LENGTH greater than 4, a read from the LINRD1 register clears the RXRDY flag. If the checksum scheme is enabled by setting the Compare Checksum (CC) bit to 1 during the reception of the data, then the byte that is received after the reception of the programmed number of data bytes indicated by the LENGTH field is treated as a checksum byte. The CC bit will be cleared once the checksum is received and compared.

28.10.2 Transmitting Data

The LIN transmitter is enabled if both the TX FUNC bit and the TXENA bit are set to 1. If the TX FUNC bit is not set, the LINTX pin functions as a general-purpose I/O pin rather than as a LIN function pin. Any value written to the TD0 before the TXENA bit is set to 1 is not transmitted. Both of these control bits allow for the LIN transmitter to be held inactive independently of the receiver.

The ID TX flag is set after a valid LIN ID is received with TX Match. An ID interrupt is generated, if enabled.

28.10.2.1 Transmitting Data in Single-Buffer Mode

Single-buffer mode is selected when the MBUF MODE bit is cleared to 0. In this mode, LIN waits for data to be written to TD0, transfers it to SCITXSHF, and transmits the data. The TXRDY and TX EMPTY bits indicate the status of the transmit buffers. That is, when the transmitter is ready for data to be written to TD0, the TXRDY bit is set. Additionally, if both TD0 and SCITXSHF are empty, then the TX EMPTY bit is also set.

You can transmit data by:

1. Polling Transmit Ready Flag
2. Transmit Interrupt
3. DMA

In polling method, software can poll for the TXRDY bit to go high before writing the data to the TD0. The CPU is unnecessarily overloaded by selecting the polling method. To avoid this, you can use either the interrupt or DMA method. To use the interrupt method, the SET TX INT bit is set. To use the DMA method, the SET TX DMA bit is set. Either an interrupt or a DMA request is generated the moment the TXRDY bit is set. When the LIN has completed transmission of all pending frames, the SCITXSHF register and the TD0 are empty, the TXRDY bit is set, and an interrupt/DMA request is generated, if enabled. Because all data has been transmitted, the interrupt/DMA request should be halted. This can either be done by disabling the transmit interrupt (CLR TX INT) / DMA request (CLR TX DMA bit) or by disabling the transmitter (clear TXENA bit). If the checksum scheme is enabled by setting the Send Checksum (SC) bit to 1, the checksum byte will be sent after the current byte transmission. The SC bit will be cleared after the checksum byte has been transmitted.

NOTE: The TXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVECT0 or SCIINTVECT1 register.

28.10.2.2 Transmitting Data in Multi-Buffer Mode

Multi-buffer mode is selected when the MBUF MODE bit is set to 1. Like single-buffer mode, you can use the polling, interrupt, or DMA method to write the data to be transmitted. The transmitted data has to be written to the LINTD0 and LINTD1 registers, based on the number of bytes. LIN waits for data to be written to Byte 0 (TD0) of the LINTD0 register and transfers the programmed number of bytes to SCITXSHF to transmit one by one automatically. If the checksum scheme is enabled by setting the Send Checksum (SC) bit to 1, the checksum will be sent after transmission of the last byte of the programmed number of data bytes, indicated by the LENGTH field. The SC bit will be cleared after the checksum byte has been transmitted.

28.11 Low-Power Mode

The SCI/LIN 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 SCI/LIN module. During global low-power mode, all clocks to the SCI/LIN are turned off so the module is completely inactive. If global low-power mode is requested while the receiver is receiving data, then the SCI/BLIN completes the current reception and then enters the low-power mode, that is, module enters low-power mode only when Busy bit (SCIFLR.3) is cleared.

The BLIN module may enter low-power mode either when there was no activity on the LINRX pin for more than 4s (this can be either a constant recessive or dominant level) or when a Sleep Command frame was received. Once the Timeout flag (SCIFLR.4) was set or once a Sleep Command was received, the POWERDOWN bit (SCIGCR2.0) must be set by the application software to make the module enter local low-power mode. A wakeup signal will terminate the sleep mode of the LIN bus.

NOTE: Enabling Local Low-Power Mode During Receive and Transmit

If the wakeup interrupt is enabled and low-power mode is requested while the receiver is receiving data, then the SCI/LIN immediately generates a wake-up interrupt to clear the powerdown bit. Thus, the SCI/LIN is prevented from entering low-power mode and completes the current reception. Otherwise, if the wakeup interrupt is disabled, the SCI/LIN completes the current reception and then enters the low-power mode.

28.11.1 Entering Sleep Mode

In LIN protocol, a sleep command is used to broadcast the sleep mode to all nodes. The sleep command consists of a diagnostic master request frame with identifier 0x3C (60), with the first data field as 0x00. There should be no activity in the bus once all nodes receive the sleep command: the bus is in sleep mode.

Local low-power mode is asserted by setting the POWERDOWN bit; setting this bit stops the clocks to the SCI/LIN internal logic and registers. Clearing the POWERDOWN bit causes SCI/LIN to exit from local low-power mode. All the registers are accessible during local power-down mode. If a register is accessed in low-power mode, this access results in enabling the clock to the module for that particular access alone.

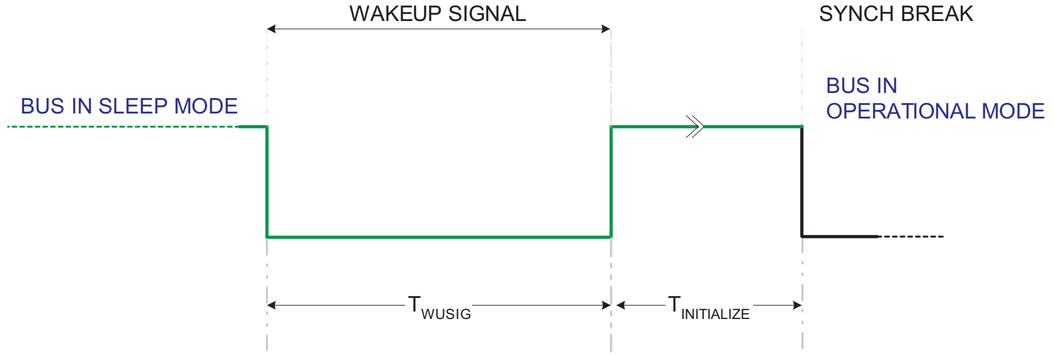
28.11.2 Wakeup

The wakeup interrupt is used to allow the SCI/LIN module to automatically exit low-power mode. A SCI/LIN wakeup is triggered when a low level is detected on the receive RX pin, and this clears the POWERDOWN bit.

NOTE: If the wakeup interrupt is disabled then the SCI/LIN enters low-power mode whenever it is requested to do so, but a low level on the receive RX pin does NOT cause the SCI/LIN to exit low-power mode.

In LIN mode, any node can terminate sleep mode by sending a wakeup signal; see Figure 28-26. A slave node that detects the bus in sleep mode, and with a wakeup request pending, will send a wakeup signal. The wakeup signal is a dominant value on the LIN bus for T_{WUSIG}; this is at least 5 T_{bits} for the LIN bus baud rates. The wakeup signal is generated by sending an 0xF0 byte containing 5 dominant T_{bits} and 5 recessive T_{bits}.

Figure 28-26. Wakeup Signal Generation



$$0.25\text{ms} \le T_{WUSIG} \le 5\text{ms} \qquad (51)$$

Assuming a perfect bus with no noise or loading effects, a write of 0xF0 to TD0 will load the transmitter to meet the wakeup signal timing requirement for T_{WUSIG}. Then, setting the GENWU bit will transmit the preloaded value in TD0 for a wakeup signal transmission.

NOTE: The GENWU bit can be set/reset only when SWnRST is set to’1’ and the node is in power down mode. The bit will be cleared on a valid synch break detection. A master sending a wakeup request, will exit power down mode upon reception of the wakeup pulse. The bit will be cleared on a SWnRST. This can be used to stop a master from sending further wakeup requests.

The TI TPIC1021 LIN transceiver, upon receiving a wakeup signal, will translate it to the microcontroller for wakeup with a dominant level on the RX pin, or a signal to the voltage regulator. While the POWERDOWN bit is set, if the LIN module detects a recessive-to-dominant edge (falling edge) on the RX pin, it will generate a wakeup interrupt if enabled in the SCISETINT register.

According to LIN protocol 2.0, the TI TPIC1021 LIN transceiver detecting a dominant level on the bus longer than 150 ms will detect it as a wakeup request. The LIN controller’s slave is ready to listen to the bus in less than 100 ms ($T_{INITIALIZE}$<100ms) after a dominant-to-recessive edge (end-of-wakeup signal).

28.11.3 Wakeup Timeouts

The LIN protocol defines the following timeouts for a wakeup sequence. After a wakeup signal has been sent to the bus, all nodes wait for the master to send a header. If no synch field is detected before 150 ms (3,000 cycles at 20 kHz) after wakeup signal is transmitted, a new wakeup is sent by the same node that requested the first wakeup. This sequence is not repeated more than two times. After three attempts to wake up the LIN bus, wakeup signal generation is suspended for a 1.5 s (30,000 cycles at 20 kHz) period after three breaks.

NOTE: To achieve compatibility to LIN1.3 timeout conditions, the MBRS register must be set to assure that the LIN 2.0 (real-time-based) timings meet the LIN 1.3 bit time base. A node triggering the wakeup should set the MBRS register accordingly to meet the targeted time as 128 Tbits x programmed prescaler.

The LIN controller handles the wakeup expiration times defined by the LIN protocol with a hardware implementation.

28.12 Emulation Mode

In emulation mode, the CONT bit determines how the SCI/LIN operates when the program is suspended. The SCI/LIN counters are affected by this bit during debug mode. when set, the counters are not stopped and when cleared, the counters are stopped debug mode.

Any reads in emulation mode to a SCI/LIN register will not have any effect on the flags in the SCIFLR register.

NOTE: When emulation mode is entered during the Frame transmission or reception of the frame and CONT bit is not set, Communication is not expected to be successful. The suggested usage is to set CONT bit during emulation mode for successful communication.

28.13 SCI/LIN Control Registers

The SCI/LIN module registers are based on the SCI registers, with added functionality registers enabled by the LIN MODE bit in the SCIGCR1 register.

These registers are accessible in 8-, 16-, and 32-bit reads or writes. The SCI/LIN is controlled and accessed through the registers listed in Table 28-9. Among the features that can be programmed are the LIN protocol mode, communication and timing modes, baud rate value, frame format, DMA requests, and interrupt configuration. The base address for the control registers is FFF7 E400h.

Table 28-9. SCI/LIN Control Registers

Offset	Acronym	Register Description	Section
00h	SCIGCR0	SCI Global Control Register 0	Section 28.13.1
04h	SCIGCR1	SCI Global Control Register 1	Section 28.13.2
08h	SCIGCR2	SCI Global Control Register 2	Section 28.13.3
0Ch	SCISETINT	SCI Set Interrupt Register	Section 28.13.4
10h	SCICLEARINT	SCI Clear Interrupt Register	Section 28.13.5
14h	SCISETINTLVL	SCI Set Interrupt Level Register	Section 28.13.6
18h	SCICLEARINTLVL	SCI Clear Interrupt Level Register	Section 28.13.7
1Ch	SCIFLR	SCI Flags Register	Section 28.13.8
20h	SCIINTVECT0	SCI Interrupt Vector Offset 0	Section 28.13.9
24h	SCIINTVECT1	SCI Interrupt Vector Offset 1	Section 28.13.10
28h	SCIFORMAT	SCI Format Control Register	Section 28.13.11
2Ch	BRS	Baud Rate Selection Register	Section 28.13.12
30h	SCIED	Receiver Emulation Data Buffer	Section 28.13.13.1
34h	SCIRD	Receiver Data Buffer	Section 28.13.13.2
38h	SCITD	Transmit Data Buffer	Section 28.13.13.3
3Ch	SCIPIO0	SCI Pin I/O Control Register 0	Section 28.13.14
40h	SCIPIO1	SCI Pin I/O Control Register 1	Section 28.13.15
44h	SCIPIO2	SCI Pin I/O Control Register 2	Section 28.13.16
48h	SCIPIO3	SCI Pin I/O Control Register 3	Section 28.13.17
4Ch	SCIPIO4	SCI Pin I/O Control Register 4	Section 28.13.18
50h	SCIPIO5	SCI Pin I/O Control Register 5	Section 28.13.19
54h	SCIPIO6	SCI Pin I/O Control Register 6	Section 28.13.20
58h	SCIPIO7	SCI Pin I/O Control Register 7	Section 28.13.21
5Ch	SCIPIO8	SCI Pin I/O Control Register 8	Section 28.13.22
60h	LINCOMPARE	LIN Compare Register	Section 28.13.23
64h	LINRD0	LIN Receive Buffer 0 Register	Section 28.13.24
68h	LINRD1	LIN Receive Buffer 1 Register	Section 28.13.25
6Ch	LINMASK	LIN Mask Register	Section 28.13.26
70h	LINID	LIN Identification Register	Section 28.13.27
74h	LINTD0	LIN Transmit Buffer 0	Section 28.13.28
78h	LINTD1	LIN Transmit Buffer 1	Section 28.13.29
7Ch	MBRS	Maximum Baud Rate Selection Register	Section 28.13.30
90h	IODFTCTRL	Input/Output Error Enable Register	Section 28.13.31

28.13.1 SCI Global Control Register 0 (SCIGCR0)

The SCIGCR0 register defines the module reset. Figure 28-27 and Table 28-10 illustrate this register.

Figure 28-27. SCI Global Control Register 0 (SCIGCR0) [offset = 00]

31	16
Reserved	
R-0	

15	1	0
Reserved		RESET
R-0		R/WP-0

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

Table 28-10. SCI Global Control Register 0 (SCIGCR0) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	RESET		This bit resets the SCI/LIN module. This bit is effective in SCI and LIN mode.
		0	SCI/LIN module is in reset.
		1	SCI/LIN module is out of reset.
			Note: Read/Write in privileged mode only.

28.13.2 SCI Global Control Register 1 (SCIGCR1)

The SCIGCR1 register defines the frame format, protocol, and communication mode used by the SCI. Figure 28-28 and Table 28-11 illustrate this register.

Figure 28-28. SCI Global Control Register 1 (SCIGCR1) [offset = 04h]

31	26	25	24
Reserved		TXENA	RXENA
R-0		R/W-0	R/W-0

23	18	17	16
Reserved		CONT	LOOP BACK
R-0		R/W-0	R/W-0

15	14	13	12	11	10	9	8
Reserved		STOP EXT FRAME	HGEN CTRL	CTYPE	MBUF MODE	ADAPT	SLEEP
R-0		R/WL-0	R/WL-0	R/WL-0	R/W-0	R/WL-0	R/W-0

7	6	5	4	3	2	1	0
SWnRST	LIN MODE	CLOCK	STOP	PARITY	PARITY ENA	TIMING MODE	COMM MODE
R/W-0	R/W-0	R/W-0	R/WC-0	R/WC-0	R/W-0	R/WC-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; WC = Write in SCI-compatible mode only; WL = Write in LIN mode only; -*n* = value after reset

Table 28-11. SCI Global Control Register 1 (SCIGCR1) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	Reads return 0. Writes have no effect.
25	TXENA		Transmit enable. This bit is effective in LIN and SCI modes. Data is transferred from SCITD, or the TDy (with y=0, 1,...7) buffers in LIN mode to the SCITXSHF shift out register only when the TXENA bit is set.
		0	Transfers from SCITD or TDy to SCITXSHF are disabled.
		1	Transfers from SCITD or TDy to SCITXSHF are enabled.
			Note: Data written to SCITD or the transmit multi-buffer before TXENA is set is not transmitted. If TXENA is cleared while transmission is ongoing, the data previously written to SCITD is sent (including the checksum byte in LIN mode).
24	RXENA		Receive enable. This bit is effective in LIN and SCI modes. RXENA allows or prevents the transfer of data from SCIRXSHF to SCIRD or the receive multi-buffers.
		0	The receiver will not transfer data from the shift buffer to the receive buffer or multi-buffers.
		1	The receiver will transfer data from the shift buffer to the receive buffer or multi-buffers.
			Note: Clearing RXENA stops received characters from being transferred into the receive buffer or multi-buffers, prevents the RX status flags (see Table 28-12) from being updated by receive data, and inhibits both receive and error interrupts. However, the shift register continues to assemble data regardless of the state of RXENA.
			Note: If RXENA is cleared before a frame is completely received, the data from the frame is not transferred into the receive buffer.
			Note: If RXENA is set before a frame is completely received, the data from the frame is transferred into the receive buffer. If RXENA is set while SCIRXSHF is in the process of assembling a frame, the status flags are not guaranteed to be accurate for that frame. To ensure that the status flags correctly reflect what was detected on the bus during a particular frame, RXENA should be set before the detection of that frame.
23-18	Reserved	0	Reads return 0. Writes have no effect.

Table 28-11. SCI Global Control Register 1 (SCIGCR1) Field Descriptions (continued)

Bit	Field	Value	Description
17	CONT		Continue on suspend. This bit is effective in LIN and SCI modes. This bit has an effect only when a program is being debugged with an emulator, and it determines how the SCI/LIN operates when the program is suspended. The SCI/LIN counters are affected by this bit: when the bit is set the counters are not stopped, when the bit is cleared the counters are stopped during debug mode.
		0	When debug mode is entered, the SCI/LIN state machine is frozen. Transmissions and LIN counters are halted and resume when debug mode is exited.
		1	When debug mode is entered, the SCI/LIN continues to operate until the current transmit and receive functions are complete.
16	LOOP BACK		Loopback bit. This bit is effective in LIN and SCI modes. The self-checking option for the SCI/LIN can be selected with this bit. If the LINITX and LINRX pins are configured with SCI/LIN functionality, then the LINTX pin is internally connected to the LINRX pin. Externally, during loop back operation, the LINTX pin outputs a high value and the LINRX pin is in a high-impedance state. If this bit value is changed while the SCI/LIN is transmitting or receiving data, errors may result.
		0	Loop back mode is disabled.
		1	Loop back mode is enabled.
15-14	Reserved	0	Reads return 0. Writes have no effect.
13	STOP EXT FRAME		Stop extended frame communication. This bit is effective in LIN mode only. This bit can be written only during extended frame communication. When the extended frame communication is stopped, this bit is cleared automatically.
		0	This bit has no effect.
		1	Extended frame communication will be stopped when current frame transmission/reception is completed.
12	HGEN CTRL		HGEN control. This bit is effective in LIN mode only. This bit controls the type of mask filtering comparison
		0	ID filtering using the ID-BYTE field in LIN Identification Register (LINID) occurs. Mask of FFh in LIN Mask Register (LINMASK) register will result in no match.
		1	ID filtering uses ID-SlaveTask BYTE (recommended). Mask of FFh in LIN Mask Register (LINMASK) register will result in ALWAYS match. **Note: For software compatibility with future LIN modules, the HGEN CTRL bit must be set to 1, the RX ID MASK must be set to FFh, and the TX ID MASK must be set to FFh.**
11	CTYPE		Checksum type. This bit is effective in LIN mode only. This bit controls the type of checksum to be used: classic or enhanced.
		0	Classic checksum is used.
		1	Enhanced checksum is used.
10	MBUF MODE		Multi-buffer mode. This bit is effective in LIN and SCI modes. This bit controls receive/transmit buffer usage, that is, whether the RX/TX multi-buffers are used or a single register, RD0/TD0, is used.
		0	Multi-buffer mode is disabled.
		1	Multi-buffer mode is enabled.
9	ADAPT		Adapt. This mode is effective in LIN mode only. This bit has an effect during the detection of the synch field. Two LIN protocol bit rate modes could be enabled with this bit according to the node capability file definition: automatic or select. The software and network configuration will decide which of these two modes are enabled. When this bit is cleared, the LIN 2.0 protocol fixed bit rate should be used. If the ADAPT bit is set, a SCI/LIN slave node detecting the baud rate will compare it to the prescalers in BRS register and update it if they are different. The BRS register will be updated with the new value. If this bit is not set there will be no adjustment to the BRS register.
		0	Automatic baud rate adjustment is disabled.
		1	Automatic baud rate adjustment is enabled.

Table 28-11. SCI Global Control Register 1 (SCIGCR1) Field Descriptions (continued)

Bit	Field	Value	Description
8	SLEEP		SCI sleep. This bit is effective in SCI mode only. In a multiprocessor configuration, this bit controls the receive sleep function. Clearing this bit brings the SCI/LIN out of sleep mode.
		0	Sleep mode is disabled.
		1	Sleep mode is enabled.
			Note: The receiver still operates when the SLEEP bit is set; however, RXRDY is updated and SCIRD is loaded with new data only when an address frame is detected. The remaining receiver status flags (see Table 28-12) are updated and an error interrupt is requested if the corresponding interrupt enable bit is set, regardless of the value of the SLEEP bit. In this way, if an error is detected on the receive data line while the SCI is asleep, software can promptly deal with the error condition.
			Note: The SLEEP bit is *not* automatically cleared when an address byte is detected.
			See Section 28.6.1 for more information on using the SLEEP bit for multiprocessor communication.
7	SWnRST		Software reset (active low). This bit is effective in LIN and SCI modes.
		0	The SCI/LIN is in its reset state; no data will be transmitted or received. Writing a 0 to this bit initializes the SCI/LIN state machines and operating flags as defined in Table 28-12 and Table 28-13. All affected logic is held in the reset state until a 1 is written to this bit.
		1	The SCI/LIN is in its ready state; transmission and reception can be done. After this bit is set to 1, the configuration of the module should not change.
			Note: The SCI/LIN should only be configured while SWnRST = 0.
6	LIN MODE		LIN mode. This bit is effective in LIN and SCI mode. This bit controls the module mode of operation.
		0	LIN mode is disabled; SCI mode is enabled.
		1	LIN mode is enabled; SCI mode is disabled.
5	CLOCK		SCI internal clock enable. The CLOCK bit determines the source of the module clock on the SCICLK pin. It also determines whether a LIN node is a slave or master.
			SCI mode
		0	The external SCICLK is the clock source.
			Note: If an external clock is selected, then the internal baud rate generator and baud rate registers are bypassed. The maximum frequency allowed for an externally sourced SCI clock is VCLK/16.
		1	The internal SCICLK is the clock source.
			LIN mode
		0	The node is in slave mode.
		1	The node is in master mode.
4	STOP		SCI number of stop bits per frame. This bit is effective in SCI mode only.
		0	One stop bit is used.
		1	Two stop bits are used.
			Note: The receiver checks for only one stop bit. However in idle-line mode, the receiver waits until the end of the second stop bit (if STOP = 1) to begin checking for an idle period.
3	PARITY		SCI parity odd/even selection. This bit is effective in SCI mode only. If the PARITY ENA bit is set, PARITY designates odd or even parity.
		0	Odd parity is used.
		1	Even parity is used.
			The parity bit is calculated based on the data bits in each frame and the address bit (in address-bit mode). The start and stop fields in the frame are not included in the parity calculation.
			For odd parity, the SCI transmits and expects to receive a value in the parity bit that makes odd the total number of bits in the frame with the value of 1.
			For even parity, the SCI transmits and expects to receive a value in the parity bit that makes even the total number of bits in the frame with the value of 1.

Table 28-11. SCI Global Control Register 1 (SCIGCR1) Field Descriptions (continued)

Bit	Field	Value	Description
2	PARITY ENA		Parity enable. This bit enables or disables the parity function.
			SCI or buffered SCI mode:
		0	Parity is disabled; no parity bit is generated during transmission or is expected during reception.
		1	Parity is enabled. A parity bit is generated during transmission and is expected during reception.
			LIN mode:
		0	ID field parity verification is disabled.
		1	ID field parity verification is enabled.
1	TIMING MODE		SCI timing mode bit. This bit is effective in SCI mode only. it selects the SCI timing mode.
		0	Synchronous timing is used.
		1	Asynchronous timing is used.
0	COMM MODE		SCI/LIN communication mode bit. In compatibility mode it selects the SCI communication mode. In LIN mode it selects length control option for ID-field bits ID4 and ID5.
			SCI mode:
		0	Idle-line mode is used.
		1	Address-bit mode is used.
			LIN mode:
		0	ID4 and ID5 are not used for length control.
		1	ID4 and ID5 are used for length control.

Table 28-12. SCI Receiver Status Flags

SCI Flag	Register	Bit	Value After Reset[1]
CE	SCIFLR	29	0
ISFE	SCIFLR	28	0
NRE	SCIFLR	27	0
FE	SCIFLR	26	0
OE	SCIFLR	25	0
PE	SCIFLR	24	0
RXWAKE	SCIFLR	12	0
RXRDY	SCIFLR	9	0
BUSY	SCIFLR	3	0
IDLE	SCIFLR	2	0
WAKE UP	SCIFLR	1	0
BRKDT	SCIFLR	0	0

[1] The flags are frozen with their reset value while SWnRST = 0.

Table 28-13. SCI Transmitter Status Flags

SCI Flag	Register	Bit	Value After Reset[1]
BE	SCIFLR	31	0
PBE	SCIFLR	30	0
TX WAKE	SCIFLR	10	0
TX EMPTY	SCIFLR	11	1
TXRDY	SCIFLR	8	1

[1] The flags are frozen with their reset value while SWnRST = 0.

28.13.3 SCI Global Control Register 2 (SCIGCR2)

The SCIGCR2 register is used to send or compare a checksum byte during extended frames, to generate a wakeup and for low-power mode control of the LIN module. Figure 28-29 and Table 28-14 illustrate this register.

Figure 28-29. SCI Global Control Register 2 (SCIGCR2) [offset = 08h]

31 18	17	16
Reserved	CC	SC
R-0	R/WL-0	R/WL-0

15 9	8	7 1	0
Reserved	GEN WU	Reserved	POWERDOWN
R-0	R/W-0	R-0	R/W-0

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

Table 28-14. SCI Global Control Register 2 (SCIGCR2) Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reads return 0. Writes have no effect.
17	CC		Compare checksum. This bit is effective in LIN mode only. This bit is used by the receiver for extended frames to trigger a checksum compare. The user will initiate this transaction by writing a one to this bit. CC bit has to be set only after RX_RDY flag is set for the last received data. In non-multi-buffer mode, when the CC bit is set, the checksum will be compared on the byte that is expected to be the checksum byte. During multi-buffer mode, the following scenarios are associated with the CC bit: a) If the CC bit is set during the reception of the data, then the byte that is received after the reception of the programmed number of data bytes as indicated by SCIFORMAT[18:16] is treated as a checksum byte. b) If the CC bit is set during the idle period (that is, during the inter-frame space), then the immediate next byte will be treated as a checksum byte. c) CC bit will be auto cleared after the checkbyte has been received and compared. Checksum reception is not guaranteed if CC bit is write cleared by software during the checksum reception. See Section 28.7.6 for more details.
		0	No checksum compare will occur.
		1	Compare checksum on expected checksum byte.
16	SC		Send checksum byte. This bit is effective in LIN mode only. This bit is used by the transmitter with extended frames to send a checksum byte. In non-multi-buffer mode, the checksum byte will be sent after the current byte transmission. In multi-buffer mode, the checksum byte will be sent after the last byte count, indicated by the SCIFORMAT[18:16]). See Section 28.7.6 for more details. This byte will be cleared after the checksum byte has been transmitted. In non-multi-buffer mode, the checksum byte will be sent after the current byte transmission. During multi-buffer mode, the following scenarios are associated with the SC bit: a) The checkbyte will be sent after the last byte count, indicated by the SCIFORMAT[18:16]). b) Checksum will not be sent if SC is set before transmitting the very first byte(that is, during interframe space). c) SC bit will be auto cleared after the checkbyte has been transmitted. Checksum transmission is not guaranteed if SC bit is write cleared by software during the checksum transmission. See Section 28.7.6 for more details.
		0	No checksum byte will be sent.
		1	A checksum byte will be sent.
15-9	Reserved	0	Reads return 0. Writes have no effect.
8	GEN WU		Generate wakeup signal. This bit is effective in LIN mode only. This bit controls the generation of a wakeup signal, by transmitting the TDO buffer value. The LIN protocol specifies that this signal should be a dominant for T_{WUSIG}. This bit is cleared on reception of a valid synch break.
		0	No wakeup signal will be generated.
		1	The TDO buffer value will be transmitted for a wakeup signal. The bit will be cleared on a SWnRST.
7-1	Reserved	0	Reads return 0. Writes have no effect.

Table 28-14. SCI Global Control Register 2 (SCIGCR2) Field Descriptions (continued)

Bit	Field	Value	Description
0	POWERDOWN		Power down. This bit is effective in LIN or SCI mode. When this bit is set, the SCI/LIN module attempts to enter local low-power mode. If the POWERDOWN bit is set while the receiver is actively receiving data and the wakeup interrupt is disabled, then the SCI/LIN will delay entering low-power mode until the reception is completed. In LIN mode, the user may set the POWERDOWN bit after receiving a sleep command or on idle bus detection (more than 4 seconds). See Section 28.11 for more information on low-power mode.
		0	The SCI/LIN module is in normal operation.
		1	The SCI/LIN module enters local low-power mode.

28.13.4 SCI Set Interrupt Register (SCISETINT)

Figure 28-30 and Table 28-15 illustrate this register. Refer Figure 28-30 for details on when different interrupt flags get set in a frame during LIN Mode.

Figure 28-30. SCI Set Interrupt Register (SCISETINT) [offset = 0Ch]

31	30	29	28	27	26	25	24
SET BE INT	SET PBE INT	SET CE INT	SET ISFE INT	SET NRE INT	SET FE INT	SET OE INT	SET PE INT
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/W-0	R/W-0	R/W-0

23				19	18	17	16
Reserved					SET RX DMA ALL	SET RX DMA	SET TX DMA
R-0					R/WC-0	R/W-0	R/W-0

15	14	13	12		10	9	8
Reserved		SET ID INT	Reserved			SET RX INT	SET TX INT
R-0		R/WL-0	R-0			R/W-0	R/W-0

7	6	5	4	3	2	1	0
SET TOA3WUS INT	SET TOAWUS INT	Reserved	SET TIMEOUT INT	Reserved		SET WAKEUP INT	SET BRKDT INT
R/WL-0	R/WL-0	R-0	R/WL-0	R-0		R/W-0	R/WC-0

LEGEND: R/W = Read/Write; R = Read only; WL = Write in LIN mode only; WC = Write in SCI-compatible mode only; -*n* = value after reset

Table 28-15. SCI Set Interrupt Register (SCISETINT) Field Descriptions

Bit	Field	Value	Description
31	SET BE INT		Set bit error interrupt. This bit is effective in LIN mode only. Setting this bit enables the SCI/LIN module to generate an interrupt when there is a bit error.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.
30	SET PBE INT		Set physical bus error interrupt. This bit is effective in LIN mode only. Setting this bit enables the SCI/LIN module to generate an interrupt when a physical bus error occurs.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.
29	SET CE INT		Set checksum-error interrupt. This bit is effective in LIN mode only. Setting this bit enables the SCI/LIN module to generate an interrupt when there is a checksum error.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.
28	SET ISFE INT		Set inconsistent-synch-field-error interrupt. This bit is effective in LIN mode only. Setting this bit enables the SCI/LIN module to generate an interrupt when there is an inconsistent synch field error.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.
27	SET NRE INT		Set no-response-error interrupt. This bit is effective in LIN mode only. Setting this bit enables the SCI/LIN module to generate an interrupt when a no-response error occurs.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.

Table 28-15. SCI Set Interrupt Register (SCISETINT) Field Descriptions (continued)

Bit	Field	Value	Description
26	SET FE INT	0	Set framing-error interrupt. This bit is effective in LIN or SCI-compatible mode. Setting this bit enables the SCI/LIN module to generate an interrupt when a framing error occurs. *Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.
25	SET OE INT	0	Set overrun-error interrupt. This bit is effective in LIN or SCI-compatible mode. Setting this bit enables the SCI/LIN module to generate an interrupt when an overrun error occurs. *Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.
24	SET PE INT	0	Set parity interrupt. This bit is effective in LIN or SCI-compatible mode. Setting this bit enables the SCI/LIN module to generate an interrupt when a parity error occurs. *Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.
23-19	Reserved	0	Reads return 0. Writes have no effect.
18	SET RX DMA ALL	0	Set receive DMA all. This bit is effective in SCI-compatible mode only. This bit determines if a separate interrupt is generated for the address frames sent in multiprocessor communications. When this bit is 0, RX interrupt requests are generated for address frames and DMA requests are generated for data frames. When this bit is 1, RX DMA requests are generated for both address and data frames. *Read:* The DMA request is disabled for address frames (the receive interrupt request is enabled for address frames). *Write:* Writing a 0 to this bit has no effect.
		1	*Read and write:* The DMA request is enabled for address and data frames.
17	SET RX DMA	0	Set receiver DMA. This bit is effective in LIN or SCI-compatible mode. To enable receiver DMA requests, this bit must be set. If it is cleared, interrupt requests are generated depending on the SET RX INT bit (SCISETINT). *Read:* The DMA request is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read and write:* The DMA request is enabled for address and data frames.
16	SET TX DMA	0	Set transmit DMA. This bit is effective in LIN or SCI-compatible mode. To enable DMA requests for the transmitter, this bit must be set. If it is cleared, interrupt requests are generated depending on the SET TX INT bit (SCISETINT). *Read:* Transmit DMA request is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read and write:* Transmit DMA request is enabled.
15-14	Reserved	0	Reads return 0. Writes have no effect.
13	SET ID INT	0	Set identification interrupt. This bit is effective in LIN mode only. This bit is set to enable an interrupt when a valid matching identifier is received. See Section 28.7.9 for more details. *Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.
12-10	Reserved	0	Reads return 0. Writes have no effect.
9	SET RX INT	0	Receiver interrupt enable. This bit is effective in LIN or SCI-compatible mode. Setting this bit enables the SCI/LIN to generate a receive interrupt after a frame has been completely received and the data is being transferred from SCIRXSHF to SCIRD. *Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.

Table 28-15. SCI Set Interrupt Register (SCISETINT) Field Descriptions (continued)

Bit	Field	Value	Description
8	SET TX INT		Set transmitter interrupt. This bit is effective in LIN or SCI-compatible mode. Setting this bit enables the SCI/LIN to generate a transmit interrupt as data is being transferred from SCITD to SCITXSHF and the TXRDY bit is being set.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.
7	SET TOA3WUS INT		Set timeout after three wakeup signals interrupt. This bit is effective in LIN mode only. Setting this bit enables the SCI/LIN to generate an interrupt when a timeout occurs after three wakeup signals have been sent.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.
6	SET TOAWUS INT		Set timeout after wakeup signal interrupt. This bit is effective in LIN mode only. Setting this bit enables the SCI/LIN to generate an interrupt when a timeout occurs after one wakeup signal has been sent.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.
5	Reserved	0	Reads return 0. Writes have no effect.
4	SET TIMEOUT INT		Set timeout interrupt. This bit is effective in LIN mode only. Setting this bit enables the SCI/LIN to generate an interrupt when there is no LIN bus activity (bus idle) for at least four seconds.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.
3-2	Reserved	0	Reads return 0. Writes have no effect.
1	SET WAKEUP INT		Set wake-up interrupt. This bit is effective in LIN or SCI-compatible mode. Setting this bit enables the SCI/LIN to generate a wakeup interrupt and thereby exit low-power mode. If enabled, the wakeup interrupt is asserted when local low-power mode is requested while the receiver is busy or if a low level is detected on the LINRX pin during low-power mode.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.
0	SET BRKDT INT		Set break-detect interrupt. This bit is effective in SCI-compatible mode only. Setting this bit enables the SCI/LIN to generate an error interrupt if a break condition is detected on the LINRX pin.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.

28.13.5 SCI Clear Interrupt Register (SCICLEARINT)

Figure 28-31 and Table 28-16 illustrate this register. SCICLEARINT register is used to clear the enabled interrupts without accessing SCISETINT register.

Figure 28-31. SCI Clear Interrupt Register (SCICLEARINT) [offset = 10h]

31	30	29	28	27	26	25	24
CLR BE INT	CLR PBE INT	CLR CE INT	CLR ISFE INT	CLR RE INT	CLR FE INT	CLR OE INT	CLR PE INT
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/W-0	R/W-0	R/W-0

23				19	18	17	16
Reserved					CLR RX DMA ALL	CLR RX DMA	CLR TX DMA
R-0					R/WC-0	R/W-0	R/W-0

15	14	13	12		10	9	8
Reserved		CLR ID INT	Reserved			CLR RX INT	CLR TX INT
R-0		R/WL-0	R-0			R/W-0	R/W-0

7	6	5	4	3	2	1	0
CLR TOA3WUS INT	CLR TOAWUS INT	Reserved	CLR TIMEOUT INT	Reserved		CLR WAKEUP INT	CLR BRKDT INT
R/WL-0	R/WL-0	R-0	R/WL-0	R-0		R/W-0	R/WC-0

LEGEND: R/W = Read/Write; R = Read only; WL = Write in LIN mode only; WC = Write in SCI-compatible mode only; -*n* = value after reset

Table 28-16. SCI Clear Interrupt Register (SCICLEARINT) Field Descriptions

Bit	Field	Value	Description
31	CLR BE INT		Clear bit error interrupt. This bit is effective in LIN mode only. This bit disables the bit error interrupt when set.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
30	CLR PBE INT		Clear physical bus error interrupt. This bit is effective in LIN mode only. This bit disables the physical-bus error interrupt when set.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
29	CLR CE INT		Clear checksum-error interrupt. This bit is effective in LIN mode only. This bit disables the checksum interrupt when set.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
28	CLR ISFE INT		Clear inconsistent-synch-field-error (ISFE) interrupt. This bit is effective in LIN mode only. This bit disables the ISFE interrupt when set.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.

Table 28-16. SCI Clear Interrupt Register (SCICLEARINT) Field Descriptions (continued)

Bit	Field	Value	Description
27	CLR NRE INT	0	Clear no-response-error interrupt. This bit is effective in LIN mode only. This bit disables the NRE interrupt when set. *Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
26	CLR FE INT	0	Clear framing-error interrupt. This bit is effective in LIN or SCI mode. This bit disables the framing-error interrupt when set. *Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
25	CLR OE INT	0	Clear overrun-error interrupt. This bit is effective in LIN or SCI mode. This bit disables the SCI/LIN overrun error interrupt when set. *Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
24	CLR PE INT	0	Clear parity interrupt. This bit is effective in LIN or SCI mode. This bit disables the parity error interrupt when set. *Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
23-19	Reserved	0	Reads return 0. Writes have no effect.
18	CLR RX DMA ALL	0	Clear receive DMA all. This bit is effective in SCI mode only. This bit clears the receive DMA request for address frames when set. Only receive data frames generate a DMA request. *Read:* Receive DMA request for address frames is disabled; Instead, RX interrupt requests are enabled for address frames. Receive DMA requests are still enabled for data frames. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* Receive DMA request for address and data frames is enabled. *Write:* Receive DMA request for address and data frames is disabled.
17	CLR RX DMA	0	Clear receive DMA request. This bit is effective in LIN or SCI mode. This bit disables the receive DMA request when set. *Read:* Receive DMA request is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* Receive DMA request is enabled. *Write:* Receive DMA request for is disabled.
16	CLR TX DMA	0	Clear transmit DMA request. This bit is effective in LIN or SCI mode. This bit disables the transmit DMA request when set. *Read:* Transmit DMA request is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* Transmit DMA request is enabled. *Write:* Transmit DMA request for is disabled.
15-14	Reserved	0	Reads return 0. Writes have no effect.

Table 28-16. SCI Clear Interrupt Register (SCICLEARINT) Field Descriptions (continued)

Bit	Field	Value	Description
13	CLR ID INT		Clear ID interrupt. This bit is effective in LIN mode only. This bit disables the ID interrupt when set.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
12-10	Reserved	0	Reads return 0. Writes have no effect.
9	CLR RX INT		Clear receiver interrupt. This bit is effective in LIN or SCI mode. This bit disables the receiver interrupt when set.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
8	CLR TX INT		Clear transmitter interrupt. This bit is effective in LIN or SCI mode. This bit disables the transmitter interrupt when set.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
7	CLR TOA3WUS INT		Clear timeout after three wakeup signals interrupt. This bit is effective in LIN mode only. This bit disables the timeout after three wakeup signals interrupt when set.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
6	CLR TOAWUS INT		Clear timeout after wakeup signal interrupt. This bit is effective in LIN mode only. This bit disables the timeout after one wakeup signal interrupt when set.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
5	Reserved	0	Reads return 0. Writes have no effect.
4	CLR TIMEOUT INT		Clear timeout interrupt. This bit is effective in LIN mode only. This bit disables the timeout (LIN bus idle) interrupt when set.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
3-2	Reserved	0	Reads return 0. Writes have no effect.
1	CLR WAKEUP INT		Clear wake-up interrupt. This bit is effective in LIN or SCI-compatible mode. This bit disables the wakeup interrupt when set.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.

Table 28-16. SCI Clear Interrupt Register (SCICLEARINT) Field Descriptions (continued)

Bit	Field	Value	Description
0	CLR BRKDT INT		Clear break-detect interrupt. This bit is effective in SCI-compatible mode only. This bit disables the break-detect interrupt when set.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.

28.13.6 SCI Set Interrupt Level Register (SCISETINTLVL)

Figure 28-32 and Table 28-17 illustrate this register.

Figure 28-32. SCI Set Interrupt Level Register (SCISETINTLVL) [offset = 14h]

31	30	29	28	27	26	25	24
SET BE INT LVL	SET PBE INT LVL	SET CE INT LVL	SET ISFE INT LVL	SET NRE INT LVL	SET FE INT LVL	SET OE INT LVL	SET PE INT LVL
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/W-0	R/W-0	R/W-0

23				19	18	17	16
Reserved					SET RX DMA ALL INT LVL	Reserved	
R-0					R/WC-0	R-0	

15	14	13	12		10	9	8
Reserved		SET ID INT LVL	Reserved			SET RX INT LVL	SET TX INT LVL
R-0		R/WL-0	R-0			R/W-0	R/W-0

7	6	5	4	3	2	1	0
SET TOA3WUS INT LVL	SET TOAWUS INT LVL	Reserved	SET TIMEOUT INT LVL	Reserved		SET WAKEUP INT LVL	SET BRKDT INT LVL
R/WL-0	R/WL-0	R-0	R/WL-0	R-0		R/W-0	R/WC-0

LEGEND: R/W = Read/Write; R = Read only; WL = Write in LIN mode only; WC = Write in SCI-compatible mode only; -*n* = value after reset

Table 28-17. SCI Set Interrupt Level Register (SCISETINTLVL) Field Descriptions

Bit	Field	Value	Description
31	SET BE INT LVL		Set bit error interrupt level. This bit is effective in LIN mode only.
		0	*Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt level is mapped to the INT1 line.
30	SET PBE INT LVL		Set physical bus error interrupt level. This bit is effective in LIN mode only.
		0	*Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt level is mapped to the INT1 line.
29	SET CE INT LVL		Set checksum-error interrupt level. This bit is effective in LIN mode only.
		0	*Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt level is mapped to the INT1 line.
28	SET ISFE INT LVL		Set inconsistent-synch-field-error interrupt level. This bit is effective in LIN mode only.
		0	*Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt level is mapped to the INT1 line.
27	SET NRE INT LVL		Set no-response-error interrupt level. This bit is effective in LIN mode only.
		0	*Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt level is mapped to the INT1 line.
26	SET FE INT LVL		Set framing-error interrupt level. This bit is effective in LIN or SCI-compatible mode.
		0	*Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt level is mapped to the INT1 line.

Table 28-17. SCI Set Interrupt Level Register (SCISETINTLVL) Field Descriptions (continued)

Bit	Field	Value	Description
25	SET OE INT LVL	0	Set overrun-error interrupt level. This bit is effective in LIN or SCI-compatible mode. *Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt level is mapped to the INT1 line.
24	SET PE INT LVL	0	Set parity error interrupt level. This bit is effective in LIN or SCI-compatible mode. *Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt level is mapped to the INT1 line.
23-19	Reserved	0	Reads return 0. Writes have no effect.
18	SET RX DMA ALL LVL	0	Set receive DMA all interrupt levels. This bit is effective in SCI mode only. *Read:*The receive interrupt request for address frames is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The receive interrupt request for address frames is mapped to the INT1 line.
17-14	Reserved	0	Reads return 0. Writes have no effect.
13	SET ID INT LVL	0	Set ID interrupt level. This bit is effective in LIN mode only. *Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt level is mapped to the INT1 line.
12-10	Reserved	0	Reads return 0. Writes have no effect.
9	SET RX INT LVL	0	Set receiver interrupt level. This bit is effective in LIN or SCI-compatible mode. *Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt level is mapped to the INT1 line.
8	SET TX INT LVL	0	Set transmitter interrupt level. This bit is effective in LIN or SCI-compatible mode. *Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt level is mapped to the INT1 line.
7	SET TOA3WUS INT LVL	0	Set timeout after three wakeup signals interrupt level. This bit is effective in LIN mode only. *Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt level is mapped to the INT1 line.
6	SET TOAWUS INT LVL	0	Set timeout after wakeup signal interrupt level. This bit is effective in LIN mode only. *Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt level is mapped to the INT1 line.
5	Reserved	0	Reads return 0. Writes have no effect.
4	SET TIMEOUT INT LVL	0	Set timeout interrupt level. This bit is effective in LIN mode only. *Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt level is mapped to the INT1 line.
3-2	Reserved	0	Reads return 0. Writes have no effect.
1	SET WAKEUP INT LVL	0	Set wake-up interrupt level. This bit is effective in LIN or SCI-compatible mode. *Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt level is mapped to the INT1 line.

Table 28-17. SCI Set Interrupt Level Register (SCISETINTLVL) Field Descriptions (continued)

Bit	Field	Value	Description
0	SET BRKDT INT LVL		Set break-detect interrupt level. This bit is effective in SCI-compatible mode only.
		0	*Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt level is mapped to the INT1 line.

28.13.7 SCI Clear Interrupt Level Register (SCICLEARINTLVL)

Figure 28-33 and Table 28-18 illustrate this register.

Figure 28-33. SCI Clear Interrupt Level Register (SCICLEARINTLVL) [offset = 18h]

31	30	29	28	27	26	25	24
CLR BE INT LVL	CLR PBE INT LVL	CLR CE INT LVL	CLR ISFE INT LVL	CLR NRE INT LVL	CLR FE INT LVL	CLR OE INT LVL	CLR PE INT LVL
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/W-0	R/W-0	R/W-0

23				19	18	17	16
Reserved					CLR RX DMA ALL INT LVL	Reserved	
R-0					R/WC-0	R-0	

15	14	13	12		10	9	8
Reserved		CLR ID INT LVL	Reserved			CLR RX INT LVL	CLR TX INT LVL
R-0		R/WL-0	R-0			R/W-0	R/W-0

7	6	5	4	3	2	1	0
CLR TOA3WUS INT LVL	CLR TOAWUS INT LVL	Reserved	CLR TIMEOUT INT LVL	Reserved		CLR WAKEUP INT LVL	CLR BRKDT INT LVL
R/WL-0	R/WL-0	R-0	R/WL-0	R-0		R/W-0	R/WC-0

LEGEND: R/W = Read/Write; R = Read only; WL = Write in LIN mode only; WC = Write in SCI-compatible mode only; -*n* = value after reset

Table 28-18. SCI Clear Interrupt Level Register (SCICLEARINTLVL) Field Descriptions

Bit	Field	Value	Description
31	CLR BE INT LVL		Clear bit error interrupt. This bit is effective in LIN mode only.
		0	*Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
30	CLR PBE INT LVL		Clear physical bus error interrupt. This bit is effective in LIN mode only.
		0	*Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
29	CLR CE INT LVL		Clear checksum-error interrupt. This bit is effective in LIN mode only.
		0	*Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
28	CLR ISFE INT LVL		Clear inconsistent-synch-field-error (ISFE) interrupt. This bit is effective in LIN mode only.
		0	*Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
27	CLR NRE INT LVL		Clear no-response-error interrupt. This bit is effective in LIN mode only.
		0	*Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.

Table 28-18. SCI Clear Interrupt Level Register (SCICLEARINTLVL) Field Descriptions (continued)

Bit	Field	Value	Description
26	CLR FE INT LVL	0	Clear framing-error interrupt. This bit is effective in LIN or SCI-compatible mode. *Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
25	CLR OE INT LVL	0	Clear overrun-error interrupt. This bit is effective in LIN or SCI-compatible mode. *Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
24	CLR PE INT LVL	0	Clear parity interrupt. This bit is effective in LIN or SCI-compatible mode. *Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
23-19	Reserved	0	Reads return 0. Writes have no effect.
18	CLR RX DMA ALL LVL	0	Clear receive DMA interrupt level. This bit is effective in SCI-compatible mode only. *Read:* The receive interrupt request for address frames is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The receive interrupt request for address frames is mapped to the INT1 line. *Write:* The receive interrupt request for address frames is mapped to the INT0 line.
17-14	Reserved	0	Reads return 0. Writes have no effect.
13	CLR ID INT LVL	0	Clear ID interrupt. This bit is effective in LIN mode only. *Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
12-10	Reserved	0	Reads return 0. Writes have no effect.
9	CLR RX INT LVL	0	Clear receiver interrupt. This bit is effective in LIN or SCI-compatible mode. *Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
8	CLR TX INT LVL	0	Clear transmitter interrupt. This bit is effective in LIN or SCI-compatible mode. *Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
7	CLR TOA3WUS INT LVL	0	Clear timeout after three wakeup signals interrupt. This bit is effective in LIN mode only. *Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.

Table 28-18. SCI Clear Interrupt Level Register (SCICLEARINTLVL) Field Descriptions (continued)

Bit	Field	Value	Description
6	CLR TOAWUS INT LVL		Clear timeout after wakeup signal interrupt. This bit is effective in LIN mode only.
		0	*Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
5	Reserved	0	Reads return 0. Writes have no effect.
4	CLR TIMEOUT INT LVL		Clear timeout interrupt. This bit is effective in LIN mode only.
		0	*Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
3-2	Reserved	0	Reads return 0. Writes have no effect.
1	CLR WAKEUP INT LVL		Clear wake-up interrupt. This bit is effective in LIN or SCI-compatible mode.
		0	*Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
0	CLR BRKDT INT LVL		Clear break-detect interrupt. This bit is effective in SCI-compatible mode only.
		0	*Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.

28.13.8 SCI Flags Register (SCIFLR)

Figure 28-34 and Table 28-19 illustrate this register.

Figure 28-34. SCI Flags Register (SCIFLR) [offset = 1Ch]

31	30	29	28	27	26	25	24
BE	PBE	CE	ISFE	NRE	FE	OE	PE
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/WL-0	R/W-0	R/W-0	R/W-0

23	16
Reserved	
R-0	

15	14	13	12	11	10	9	8
Reserved	ID RX	ID TX	RX WAKE	TX EMPTY	TX WAKE	RX RDY	TX RDY
R-0	R/WL-0	R/WL-0	R/WC-0	R/W-1	R/WC-0	R/W-0	R/W-1
7	6	5	4	3	2	1	0
TOA3WUS	TOAWUS	Reserved	TIMEOUT	BUSY	IDLE	WAKEUP	BRKDT
R/WL-0	R/WL-0	R-0	R/WL-0	R/W-0	R-0	R/WL-0	R/WC-0

LEGEND: R/W = Read/Write; R = Read only; WC = Write in SCI-compatible mode only; WL = Write in LIN mode only; -*n* = value after reset

Table 28-19. SCI Flags Register (SCIFLR) Field Descriptions

Bit	Field	Value	Description
31	BE		Bit error flag. This bit is effective in LIN mode only. This bit is set when a bit error has occurred. This is detected by the internal bit monitor. See Section 28.7.8 for more information. The bit error flag is cleared by any of the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • On reception of a new synch break • Reading the corresponding interrupt offset in SCIINTVECT0/1
		0	*Read:* No error has been detected since this bit was last cleared. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* An error has been detected since this bit was last cleared. *Write:* The bit is cleared to 0.
30	PBE		Physical bus error flag. This bit is effective in LIN mode only. This bit is set when a physical bus error has been detected by the bit monitor in TED. See Section 28.7.8 for more information. The physical bus error flag is cleared by the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • On reception of a new synch break • Reading the corresponding interrupt offset in SCIINTVECT0/1 **Note: The PBE will only be flagged, if no Synch Break can be generated (for example, because of a bus shortage to VBAT) or if no Synch Break Delimiter can be generated (for example, because of a bus shortage to GND).**
		0	*Read:* No error has been detected since this bit was last cleared. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* An error has been detected since this bit was last cleared. *Write:* The bit is cleared to 0.

Table 28-19. SCI Flags Register (SCIFLR) Field Descriptions (continued)

Bit	Field	Value	Description
29	CE		Checksum error flag. This bit is effective in LIN mode only. This bit is set when a checksum error has been detected by a receiving node. This error is detected by the TED logic. See Section 28.7.8 for more information. The type of checksum to be used depends on the CTYPE bit in SCIGCR1. The checksum error flag is cleared by the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reception of a new synch break • Reading the corresponding interrupt offset in SCIINTVECT0/1
		0	*Read:* No error has been detected since this bit was last cleared. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* An error has been detected since this bit was last cleared. *Write:* The bit is cleared to 0.
28	ISFE		Inconsistent synch field error flag. This bit is effective in LIN mode only. This bit is set when an inconsistent synch field error has been detected by the synchronizer during header reception. See Section 28.7.5.2 for more information. The inconsistent synch field error flag is cleared by the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reception of a new synch break • Reading the corresponding interrupt offset in SCIINTVECT0/1
		0	*Read:* No inconsistent synch field error has been detected. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* An inconsistent synch field error has been detected. *Write:* The bit is cleared to 0.
27	NRE		No-response error flag. This bit is effective in LIN mode only. This bit is set when there is no response to a master's header completed within TFRAME_MAX. This timeout period is applied for message frames of known length (identifiers 0 to 61). This error is detected by the synchronizer. See Section 28.7.7 for more information. The no-response error flag is cleared by the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reception of a new synch break • Reading the corresponding interrupt offset in SCIINTVECT0/1
		0	*Read:* No no-response error has been detected since the last clear. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* A no-response error has been detected. *Write:* The bit is cleared to 0.

Table 28-19. SCI Flags Register (SCIFLR) Field Descriptions (continued)

Bit	Field	Value	Description
26	FE		Framing error flag. This bit is effective in LIN or SCI-compatible mode. This bit is set when an expected stop bit is not found. In SCI compatibility mode, only the first stop bit is checked. The missing stop bit indicates that synchronization with the start bit has been lost and that the character is incorrectly framed. Detection of a framing error causes the SCI/LIN to generate an error interrupt if the SET FE INT bit is set in the register SCISETINT. The framing error flag is cleared by the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reading the corresponding interrupt offset in SCIINTVECT0/1 • Reception of a new character/frame, depending on whether the module is in SCI compatible or LIN mode In multi-buffer mode, the frame is defined in the SCIFORMAT register.
		0	*Read:* No framing error has been detected since the last clear. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* A framing error has been detected since the last clear. *Write:* The bit is cleared to 0.
25	OE		Overrun error flag. This bit is effective in LIN or SCI-compatible mode. This bit is set when the transfer of data from SCIRXSHF to SCIRD overwrites unread data already in SCIRD or the RDy buffers in LINRD0 and LINRD1. Detection of an overrun error causes the LIN to generate an error interrupt if the SET OE INT bit = 1. The OE flag is reset by the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reading the corresponding interrupt offset in SCIINTVECT0/1
		0	*Read:* No overrun error has been detected since the last clear. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* An overrun error has been detected. *Write:* The bit is cleared to 0.
24	PE		Parity error flag. This bit is effective in LIN or SCI-compatible mode. This bit is set when a parity error is detected in the received data. In SCI address-bit mode, the parity is calculated on the data and address bit fields of the received frame. In idle-line mode, only the data is used to calculate parity. An error is generated when a character is received with a mismatch between the number of 1s and its parity bit. If the parity function is disabled (SCIGCR[2] = 0), the PE flag is disabled and read as 0. Detection of a parity error causes the LIN to generate an error interrupt if the SET PE INT bit = 1. The PE bit is reset by the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reception of a new character or frame, depending on whether the module is in SCI compatible or LIN mode, respectively. • Reading the corresponding interrupt offset in SCIINTVECT0/1
		0	*Read:* No parity error has been detected since the last clear. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* A parity error has been detected. *Write:* The bit is cleared to 0.
23-15	Reserved	0	Reads return 0. Writes have no effect.

Table 28-19. SCI Flags Register (SCIFLR) Field Descriptions (continued)

Bit	Field	Value	Description
14	ID RX FLAG		Identifier on receive flag. This bit is effective in LIN mode only. This flag is set once an identifier is received with an receive match and no ID-parity error. See Section 28.7.9 for more details. This flag indicates that a new valid identifier has been received on an RX match. This bit is cleared by the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reading the LINID register • Reception of a new synch break • Reading the corresponding interrupt offset in SCIINTVECT0/1
		0	*Read:* No valid ID has been received since the last clear. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* A valid ID RX has been received in LINID[23:16] on an RX match. *Write:* The bit is cleared to 0.
13	ID TX FLAG		Identifier on transmit flag. This bit is effective in LIN mode only. This flag is set when an identifier is received with a transmit match and no ID-parity error. See Section 28.7.9 for more details. This flag indicates that a new valid identifier has been received on a TX match. This bit is cleared by the following: • Setting the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reading the LINID register • Receiving a new synch break • Reading the corresponding interrupt offset in SCIINTVECT0/1
		0	*Read:* No valid ID has been received since the last clear. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* A valid ID TX has been received in LINID[23:16] on an TX match. *Write:* The bit is cleared to 0.
12	RXWAKE		Receiver wakeup detect flag. This bit is effective in SCI-compatible mode only. The SCI sets this bit to indicate that the data currently in SCIRD is an address. RXWAKE is cleared by the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Upon receipt of a data frame.
		0	The data in SCIRD is not an address.
		1	The data in SCIRD is an address.
11	TX EMPTY		Transmitter empty flag. This flag indicates the transmitter's buffer register(s) (SCITD/TDy) and shift register (SCITXSHF) are empty. In multi-buffer mode, this flag indicates the TDx registers and shift register (SCITXSHF) are empty. In non-multi-buffer mode, this flag indicates the LINTD0 byte and the shift register (SCITXSHF) are empty. **Note: The RESET bit, an active SWnRST (SCIGCR1[7]) or a system reset sets this bit. This bit does not cause an interrupt request.** *SCI mode or LIN non-multi-buffer mode:*
		0	Transmitter buffer or shift register (or both) are loaded with data.
		1	Transmitter buffer and shift registers are both empty.
			In LIN mode using multi-buffer mode:
		0	Multi-buffer or shift register (or all) are loaded with data.
		1	Multi-buffer and shift registers are all empty.

Table 28-19. SCI Flags Register (SCIFLR) Field Descriptions (continued)

Bit	Field	Value	Description
10	TXWAKE		Transmitter wakeup method select. This bit is effective in SCI mode only. The TXWAKE bit controls whether the data in SCITD should be sent as an address or data frame using multiprocessor communication format. This bit is set to 1 or 0 by software before a byte is written to SCITD and is cleared by the SCI when data is transferred from SCITD to SCITXSHF or by a system reset. **Note: TXWAKE is not cleared by the SWnRST bit.** *Address-bit mode*:
		0	Frame to be transmitted will be data (address bit = 0).
		1	Frame to be transmitted will be an address (address bit = 1).
			Idle-line mode:
		0	The frame to be transmitted will be data.
		1	The following frame to be transmitted will be an address (writing a 1 to this bit followed by writing dummy data to the SCITD will result in a idle period of 11 bit periods before the next frame is transmitted).
9	RXRDY		Receiver ready flag. In SCI-compatible mode, the receiver sets this bit to indicate that the SCIRD contains new data and is ready to be read by the CPU. In *LIN mode,* RXRDY is set once a valid frame is received in multi-buffer mode, a valid frame being a message frame received with no errors. In *non-multi-buffer mode,* RXRDY is set for each received byte and will be set for the last byte of the frame if there are no errors. The SCI/LIN generates a receive interrupt when RXRDY flag bit is set if the SET RX INT bit is set (SCISETINT[9]); RXRDY is cleared by the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reading the SCIRD register in compatibility mode • Reading the last data byte RDy of the response in LIN mode **Note: The RXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register.**
		0	*Read:* No new data is in SCIRD. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* New data is ready to be read from SCIRD. *Write:* The bit is cleared to 0.
8	TXRDY		Transmitter buffer register ready flag. When set, this bit indicates that the transmit buffer(s) register(s) (SCITD in compatibility mode and LINTD0/LINTD1 in multi-buffer mode) are ready to get another character from a CPU write. *In SCI,* writing data to SCITD automatically clears this bit. In *LIN mode,* this bit is cleared once byte 0 (TD0) is written to LINTD0. This bit is set after the data of the TX buffer is shifted into the SCITXSHF register. This event can trigger a transmit interrupt after data is copied to the TX shift register SCITXSHF, if the SET TX INT bit is set. **Note:** **1) TXRDY is also set to 1 by setting of the RESET bit, enabling SWnRST, or by a system reset.** **2) The TXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register.** **3) The transmit interrupt request can be eliminated until the next series of data written into the transmit buffers LINTD0 and LINTD1, by disabling the corresponding interrupt via the SCICLEARINT register or by disabling the transmitter via the TXENA bit.** *SCI mode:*
		0	SCITD is full.
		1	SCITD is ready to receive the next character.
			LIN mode:
		0	Mult-ibuffers are full.
		1	Multi-buffers are ready to receive the next character. For more information on transmit interrupt handling, see the SCI document for compatibility mode and Section 28.7.9 for LIN mode.

Table 28-19. SCI Flags Register (SCIFLR) Field Descriptions (continued)

Bit	Field	Value	Description
7	TOA3WUS		Timeout after three wakeup signals flag. This bit is effective in LIN mode only. This flag is set if there is no synch break received after three wakeup signals and a period of 1.5 seconds has passed. Such expiration time is used before issuing another round of wakeup signals. This bit is cleared by the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reading the corresponding interrupt offset in SCIINTVECT0/1 See Section 28.11.3 for more information.
		0	*Read:* No timeout occurred after three wakeup signals. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* Timeout occurred after three wakeup signals and 1.5 seconds time. *Write:* The bit is cleared to 0.
6	TOAWUS		Timeout after wakeup signal flag. This bit is effective in LIN mode only. This bit is set if there is no synch break received after a wakeup signal has been sent. A minimum of 150 ms expiration time is used before issuing another wakeup signal. This bit is cleared by the following: • Setting the SWnRST bit • Setting of the RESET bit • A system reset occurring • Writing a 1 to this bit • Reading the corresponding interrupt offset in SCIINTVECT0/1 See Section 28.11.3 for more information.
		0	*Read:* No timeout occurred after one wakeup signal (150 ms). *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* Timeout occurred after one wakeup signal. *Write:* The bit is cleared to 0.
5	Reserved	0	Reads return 0. Writes have no effect.
4	TIMEOUT		LIN bus idle timeout flag. This bit is effective in LIN mode only. This bit is set earliest after at least four seconds of bus inactivity. Bus inactivity is defined as no transactions between recessive and dominant (and vice versa). This bit is cleared by the following: • Setting the SWnRST bit • Setting of the RESET bit • A system reset occurring • Writing a 1 to this bit • Reading the corresponding interrupt offset in SCIINTVECT0/1 See Section 28.7.7 for more information.
		0	*Read:* No bus idle has been detected since this bit was last cleared. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* A LIN bus idle has been detected. *Write:* The bit is cleared to 0.
3	BUSY		Bus busy flag. This bit is effective in LIN mode and SCI-compatible mode. This bit indicates whether the receiver is in the process of receiving a frame. As soon as the receiver detects the beginning of a start bit, the BUSY bit is set to 1. When the reception of a frame is complete, the SCI/LIN clears the BUSY bit. If SET WAKEUP INT is set and power down is requested while this bit is set, the SCI/LIN automatically prevents low-power mode from being entered and generates wakeup interrupt. The BUSY bit is controlled directly by the SCI/LIN receiver, but this bit can also be cleared by the following: • Setting the SWnRST bit • Setting of the RESET bit • A system reset occurring
		0	The receiver is not currently receiving a frame.
		1	The receiver is currently receiving a frame.

Table 28-19. SCI Flags Register (SCIFLR) Field Descriptions (continued)

Bit	Field	Value	Description
2	IDLE		SCI receiver in idle state. This bit is effective in SCI-compatible mode only. While this bit is set, the SCI looks for an idle period to resynchronize itself with the bit stream. The receiver does not receive any data while the bit is set. The bus must be idle for 11 bit periods to clear this bit. The SCI enters the idle state if one of the following events occurs: • A system reset • An SCI software reset • A power down • The RX pin is configured as a general I/O pin
		0	The idle period has been detected; the SCI is ready to receive.
		1	The idle period has not been detected; the SCI will not receive any data.
1	WAKEUP		Wakeup flag. This bit is effective in LIN mode only. This bit is set by the SCI/LIN when receiver or transmitter activity has taken the module out of power-down mode. An interrupt is generated if the SET WAKEUP INT bit (SCISETINT[2]) is set. It is cleared by the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reading the corresponding interrupt offset in SCIINTVECT0/1 For compatibility mode, see the SCI document for more information on low-power mode.
		0	*Read:* The module will not wake up from power-down mode. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* Wake up from power-down mode. *Write:* The bit is cleared to 0.
0	BRKDT		SCI break-detect flag. This bit is effective in SCI-compatible mode only. This bit is set when the SCI detects a break condition on the LINRX pin. A break condition occurs when the LINRX pin remains continuously low for at least 10 bits after a missing first stop bit, that is, after a framing error. Detection of a break condition causes the SCI to generate an error interrupt if the SET BRKDT INT bit is set. The BRKDT bit is reset by the following: • Setting of the SWnRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reading the corresponding interrupt offset in SCIINTVECT0/1
		0	*Read:* No break condition has been detected since the last clear. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* A break condition has been detected. *Write:* The bit is cleared to 0.

28.13.9 SCI Interrupt Vector Offset 0 (SCIINTVECT0)

Figure 28-35 and Table 28-20 illustrate this register.

Figure 28-35. SCI Interrupt Vector Offset 0 (SCIINTVECT0) [offset = 20h]

31	16
Reserved	
R-0	

15 5	4 0
Reserved	INTVECT0
R-0	R-0

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

Table 28-20. SCI Interrupt Vector Offset 0 (SCIINTVECT0) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reads return 0. Writes have no effect.
4-0	INVECT0	0-1Fh	Interrupt vector offset for INT0. This register indicates the offset for interrupt line INT0. A read to this register updates its value to the next highest priority pending interrupt in SCIFLR and clears the flag in SCIFLR corresponding to the offset that was read. See Table 28-4 for a list of the interrupts. **Note: The flags for the receive (SCIFLR[9]) and the transmit (SCIFLR[8]) interrupt cannot be cleared by reading the corresponding offset vector in this register (see detailed description in SCIFLR register).**

28.13.10 SCI Interrupt Vector Offset 1 (SCIINTVECT1)

Figure 28-36 and Table 28-21 illustrate this register.

Figure 28-36. SCI Interrupt Vector Offset 1 (SCIINTVECT1) [offset = 24h]

31	16
Reserved	
R-0	

15 5	4 0
Reserved	INTVECT1
R-0	R-0

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

Table 28-21. SCI Interrupt Vector Offset 1 (SCIINTVECT1) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reads return 0. Writes have no effect.
4-0	INVECT1	0-1Fh	Interrupt vector offset for INT1. This register indicates the offset for interrupt line INT1. A read to this register updates its value to the next highest priority pending interrupt in SCIFLR and clears the flag in SCIFLR corresponding to the offset that was read. See Table 28-4 for list of interrupts. **Note: The flags for the receive (SCIFLR[9]) and the transmit (SCIFLR[8]) interrupt cannot be cleared by reading the corresponding offset vector in this register (see detailed description in SCIFLR register).**

28.13.11 SCI Format Control Register (SCIFORMAT)

Figure 28-37 and Table 28-22 illustrate this register.

Figure 28-37. SCI Format Control Register (SCIFORMAT) [offset = 28h]

31 19	18 16
Reserved	LENGTH
R-0	R/W-0

15 3	2 0
Reserved	CHAR
R-0	R/WC-0

LEGEND: R/W = Read/Write; R = Read only; WC = Write in SCI-compatible mode only; -*n* = value after reset

Table 28-22. SCI Format Control Register (SCIFORMAT) Field Descriptions

Bit	Field	Value	Description
31-19	Reserved	0	Reads return 0. Writes have no effect.
18-16	LENGTH		Frame length control bits. In *LIN mode*, these bits indicate the number of bytes in the response field from 1 to 8 bytes. In *buffered SCI mode*, these bits indicate the number of characters, with the number of bits per character specified in CHAR (SCIFORMAT[2:0]). When these bits are used to indicate LIN response length (SCIGCR1[0] = 1), then when there is an ID RX match, this value should be updated with the expected length of the response. In buffered SCI mode, these bits indicate the transmitter/receiver format for the number of characters: 1 to 8. There can be up to eight characters with eight bits each.
		0	The response field has 1 byte/character.
		1h	The response field has 2 bytes/characters.
		2h	The response field has 3 bytes/characters.
		3h	The response field has 4 bytes/characters.
		4h	The response field has 5 bytes/characters.
		5h	The response field has 6 bytes/characters.
		6h	The response field has 7 bytes/characters.
		7h	The response field has 8 bytes/characters.
15-3	Reserved	0	Reads return 0. Writes have no effect.
2-0	CHAR		Character length control bits. These bits are effective in non-buffered SCI and buffered SCI modes only. These bits set the SCI character length from 1 to 8 bits. **In non-buffered SCI and buffered SCI mode, when data of fewer than eight bits in length is received, it is left justified in SCIRD/RDy and padded with trailing zeros. Data read from the SCIRD should be shifted by software to make the received data right justified.** **Data written to the SCITD should be right justified but does not need to be padded with leading zeros.**
		0	The character is 1 bit long.
		1h	The character is 2 bits long.
		2h	The character is 3 bits long.
		3h	The character is 4 bits long.
		4h	The character is 5 bits long.
		5h	The character is 6 bits long.
		6h	The character is 7 bits long.
		7h	The character is 8 bits long.

28.13.12 Baud Rate Selection Register (BRS)

This section describes the baud rate selection register. Figure 28-38 and Table 28-23 illustrate this register.

Figure 28-38. Baud Rate Selection Register (BRS) [offset = 2Ch]

31	30 - 28	27 - 24	23 - 16
Rsvd	U	M	PRESCALER P
R-0	R/W-0	R/W-0	R/W-0

15 - 0
PRESCALER P
R/W-0

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

Table 28-23. Baud Rate Selection Register (BRS) Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reads return 0. Writes have no effect.
30-28	U	0-2h	SCI/LIN super fractional divider selection. These bits are effective in LIN or SCI asynchronous mode. These bits are an additional fractional part for the baud rate specification. These bits allow a super-fine tuning of the fractional baud rate with seven more intermediate values for each of the M fractional divider values. See Section 28.7.4.1 for more details.
27-24	M	0-3h	SCI/LIN 4-bit fractional divider selection. These bits are effective in LIN or SCI asynchronous mode. These bits are used to select a baud rate for the SCI/LIN module, and they are a fractional part for the baud rate specification. The M divider allows fine-tuning of the baud rate over the P prescaler with 15 additional intermediate values for each of the P integer values. See Section 28.7.4.1 for more details.
23-0	PRESCALER P	0-FF FFFFh	These bits are used to select a baud rate for the SCI/LIN module. These bits are effective in LIN mode and SCI compatibility. The SCI/LIN has an internally generated serial clock determined by the VCLK and the prescalers P and M in this register. The LIN uses the 24-bit integer prescaler P value of this register to select one of over 16,700,000. The additional 4-bit fractional divider M refines the baud rate selection PRESCALER[27:24]. **NOTE: In LIN mode, ONLY the asynchronous mode and baud rate values are used.** The baud rate can be calculated using the following formulas: $\textit{Asynchronous baud value} = \left(\frac{\text{VCLK Frequency}}{16\left(\text{P} + 1 + \frac{\text{M}}{16}\right)}\right)$ (52) $\textit{Isosynchronous baud value} = \left(\frac{\text{VCLK Frequency}}{\text{P} + 1}\right)$ (53) For P = 0, $\textit{Asynchronous baud value} = \left(\frac{\text{VCLK Frequency}}{32}\right)$ (54) $\textit{Isosynchronous baud value} = \left(\frac{\text{VCLK Frequency}}{2}\right)$ (55) Table 28-24 contains comparative baud values for different P values, with VCLK = 50 MHz, for asynchronous mode.

Table 28-24. Comparative Baud Values for Different P Values, Asynchronous Mode[1][2]

24-Bit Register Value		Baud Selected		Percent Error
Decimal	Hex	Ideal	Actual	
26	00001A	115200	115740	0.47
53	000035	57600	57870	0.47
80	000050	38400	38580	0.47
162	0000A2	19200	19172	-0.15
299	00012B	10400	10417	0.16
325	000145	9600	9586	-0.15
399	00018F	7812.5	7812.5	0.00
650	00028A	4800	4800	0.00
15624	003BA0	200	200	0.00
624999	098967	5	5	0.00

[1] VCLK = 50 MHz

[2] Values are in decimal except for column 2.

28.13.13 SCI Data Buffers (SCIED, SCIRD, SCITD)

The SCI has three addressable registers in which transmit and receive data is stored. These three registers are available in SCI mode only.

28.13.13.1 Receiver Emulation Data Buffer (SCIED)

The SCIED register is addressed at a location different from SCIRD, but is physically the same register. Figure 28-39 and Table 28-25 illustrate this register.

Figure 28-39. Receiver Emulation Data Buffer (SCIED) [offset = 30h]

31-16
Reserved
R-0

15-8	7-0
Reserved	ED
R-0	R-0

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

Table 28-25. Receiver Emulation Data Buffer (SCIED) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	ED	0-FFh	Emulator data. This bit is effective in SCI-compatible mode only. Reading SCIED[7:0] does not clear the RXRDY flag, unlike reading SCIRD. This register should be used only by an emulator that must continually read the data buffer without affecting the RXRDY flag.

28.13.13.2 Receiver Data Buffer (SCIRD)

This register provides a location for the receiver data. Figure 28-40 and Table 28-26 illustrate this register.

Figure 28-40. Receiver Data Buffer (SCIRD) [offset = 34h]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		RD	
R-0		R-0	

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

Table 28-26. Receiver Data Buffer (SCIRD) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	RD	0-FFh	Receiver data. This bit is effective in SCI-compatible mode only. When a frame has been completely received, the data in the frame is transferred from the receiver shift register SCIRXSHF to this register. As this transfer occurs, the RXRDY flag is set and a receive interrupt is generated if SET RX INT is set. **Note: When the data is read from SCIRD, the RXRDY flag is automatically cleared.**

NOTE: When the SCI receives data that is fewer than eight bits in length, it loads the data into this register in a left justified format padded with trailing zeros. Therefore, your software should perform a logical shift on the data by the correct number of positions to make it right justified.

28.13.13.3 Transmit Data Buffer Register (SCITD)

Data to be transmitted is written to the SCITD register. Figure 28-41 and Table 28-27 illustrate this register.

Figure 28-41. Transmit Data Buffer Register (SCITD) [offset = 38h]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		TD	
R-0		R/W-0	

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

Table 28-27. Transmit Data Buffer Register (SCITD) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	TD	0-FFh	Transmit data. This bit is effective in SCI-compatible mode only. Data to be transmitted is written to the SCITD register. The transfer of data from this register to the transmit shift register SCITXSHF sets the TXRDY flag, which indicates that SCITD is ready to be loaded with another byte of data. **Note: If SET TX INT is set, this data transfer also causes an interrupt.**

NOTE: Data written to the SCITD register that is fewer than eight bits long must be right justified, but it does not need to be padded with leading zeros.

28.13.14 SCI Pin I/O Control Register 0 (SCIPIO0)

Figure 28-42 and Table 28-28 illustrate this register.

Figure 28-42. SCI Pin I/O Control Register 0 (SCIPIO0) [offset = 3Ch]

31	8
Reserved	
R-0	

7	3	2	1	0
Reserved		TX FUNC	RX FUNC	Reserved
R-0		R/W-0	R/W-0	R-0

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

Table 28-28. SCI Pin I/O Control Register 0 (SCIPIO0) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TX FUNC		Transfer function. This bit is effective in LIN or SCI mode. This bit defines the function of pin LINTX.
		0	LINTX is a general-purpose digital I/O pin.
		1	LINTX is the SCI/LIN transmit pin.
1	RX FUNC		Receive function. This bit is effective in LIN or SCI mode. This bit defines the function of pin LINRX.
		0	LINRX is a general-purpose digital I/O pin.
		1	LINRX is the SCI/LIN receive pin.
0	Reserved	0	Reads return 0. Writes have no effect.

28.13.15 SCI Pin I/O Control Register 1 (SCIPIO1)

Figure 28-43 and Table 28-29 illustrate this register.

Figure 28-43. SCI Pin I/O Control Register 1 (SCIPIO1) [offset = 40h]

31–8
Reserved
R-0

7–3	2	1	0
Reserved	TX DIR	RX DIR	Reserved
R-0	R/W-0	R/W-0	R-0

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

Table 28-29. SCI Pin I/O Control Register 1 (SCIPIO1) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TX DIR		Transmit pin direction. This bit is effective in LIN or SCI mode. This bit determines the data direction on the LINTX pin if it is configured with general-purpose I/O functionality (TX FUNC = 0). See Table 28-30 for the LINTX pin control with this bit and others.
		0	LINTX is a general-purpose input pin.
		1	LINTX is a general-purpose output pin.
1	RX DIR		Receive pin direction. This bit is effective in LIN or SCI mode. This bit determines the data direction on the LINRX pin if it is configured with general-purpose I/O functionality (RX FUNC = 0). See Table 28-31 for the LINRX pin control with this bit and others.
		0	LINRX is a general-purpose input pin.
		1	LINRX is a general-purpose output pin.
0	Reserved	0	Reads return 0. Writes have no effect.

Table 28-30. LINTX Pin Control

Function	TX IN [(1)]	TX OUT	TX FUNC	TX DIR
LINTX	X	X	1	X
General purpose input	X	X	0	0
General purpose output, high	X	1	0	1
General purpose output, low	X	0	0	1

[(1)] TX IN is a read-only bit. Its value always reflects the level of the SCITX pin.

Table 28-31. LINRX Pin Control

Function	RX IN [(1)]	RX OUT	RX FUNC	RX DIR
LINRX	X	X	1	X
General purpose input	X	X	0	0
General purpose output, high	X	1	0	1
General purpose output, low	X	0	0	1

[(1)] RX IN is a read-only bit. Its value always reflects the level of the SCIRX pin.

28.13.16 SCI Pin I/O Control Register 2 (SCIPIO2)

Figure 28-44 and Table 28-32 illustrate this register.

Figure 28-44. SCI Pin I/O Control Register 2 (SCIPIO2) [offset = 44h]

31-8
Reserved
R-0

7-3	2	1	0
Reserved	TX IN	RX IN	Reserved
R-0	R-x	R-x	R-x

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

Table 28-32. SCI Pin I/O Control Register 2 (SCIPIO2) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TX IN		Transmit pin in. This bit is effective in LIN or SCI mode. This bit contains the current value on the LINTX pin.
		0	The LINTX pin is at logic low (0).
		1	The LINTX pin is at logic high (1).
1	RX IN		Receive pin in. This bit is effective in LIN or SCI mode. This bit contains the current value on the LINRX pin.
		0	The LINRX pin is at logic low (0).
		1	The LINRX pin is at logic high (1).
0	Reserved		Writes have no effect.

28.13.17 SCI Pin I/O Control Register 3 (SCIPIO3)

Figure 28-45 and Table 28-33 illustrate this register.

Figure 28-45. SCI Pin I/O Control Register 3 (SCIPIO3) [offset = 48h]

31	8
Reserved	
R-0	

7	3	2	1	0
Reserved		TX OUT	RX OUT	Reserved
R-0		R/W-0	R/W-0	R-0

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

Table 28-33. SCI Pin I/O Control Register 3 (SCIPIO3) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TX OUT		Transmit pin out. This bit is effective in LIN or SCI mode. This bit specifies the logic to be output on pin LINTX if the following conditions are met: • TX FUNC = 0 (LINTX pin is a general-purpose I/O.) • TX DIR = 1 (LINTX pin is a general-purpose output.) See Table 28-30 for an explanation of this bit's effect in combination with other bits.
		0	The output on the LINTX is at logic low (0).
		1	The output on the LINTX pin is at logic high (1). (Output voltage is V_{OH} or higher if TXPDR = 0 and output is in high impedance state if TXPDR = 1.)
1	RX OUT		Receive pin out. This bit is effective in LIN or SCI mode. This bit specifies the logic to be output on pin LINRX if the following conditions are met: • RX FUNC = 0 (LINRX pin is a general-purpose I/O.) • RX DIR = 1 (LINRX pin is a general-purpose output.) See Table 28-31 for an explanation of this bit's effect in combination with the other bits
		0	The output on the LINRX pin is at logic low (0).
		1	The output on the LINRX pin is at logic high (1). (Output voltage is V_{OH} or higher if RXPDR = 0, and output is in high impedance state if RXPDR = 1.)
0	Reserved	0	Reads return 0. Writes have no effect.

28.13.18 SCI Pin I/O Control Register 4 (SCIPIO4)

Figure 28-46 and Table 28-34 illustrate this register.

Figure 28-46. SCI Pin I/O Control Register 4 (SCIPIO4) [offset = 4Ch]

31 – 8
Reserved
R-0

7 – 3	2	1	0
Reserved	TX SET	RX SET	Reserved
R-0	R/W-0	R/W-0	R-0

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

Table 28-34. SCI Pin I/O Control Register 4 (SCIPIO4) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TX SET		Transmit pin set. This bit is effective in LIN or SCI mode. This bit sets the logic to be output on pin LINTX if the following conditions are met: • TX FUNC = 0 (LINTX pin is a general-purpose I/O.) • TX DIR = 1 (LINTX pin is a general-purpose output.) See Table 28-30 for an explanation of this bit’s effect in combination with other bits.
		0	*Read:* The output on LINTX is at logic low (0). *Write:* Writing a 0 to this bit has no effect.
		1	*Read and write:* The output on LINTX is at logic high (1).
1	RX SET		Receive pin set. This bit is effective in LIN or SCI mode. This bit sets the logic to be output on pin LINRX if the following conditions are met: • RX FUNC = 0 (LINRX pin is a general-purpose I/O.) • RX DIR = 1 (LINRX pin is a general-purpose output.) See Table 28-31 for an explanation of this bit’s effect in combination with the other bits.
		0	*Read:* The output on LINRX is at logic low (0). *Write:* Writing a 0 to this bit has no effect.
		1	*Read and write:* The output on LINRX is at logic high (1).
0	Reserved	0	Reads return 0. Writes have no effect.

28.13.19 SCI Pin I/O Control Register 5 (SCIPIO5)

Figure 28-47 and Table 28-35 illustrate this register.

Figure 28-47. SCI Pin I/O Control Register 5 (SCIPIO5) [offset = 50h]

31–8
Reserved
R-0

7–3	2	1	0
Reserved	TX CLR	RX CLR	Reserved
R-0	R/W-0	R/W-0	R-0

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

Table 28-35. SCI Pin I/O Control Register 5 (SCIPIO5) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TX CLR		Transmit pin clear. This bit is effective in LIN or SCI mode. This bit clears the logic to be output on pin LINTX if the following conditions are met: • TX FUNC = 0 (LINTX pin is a general-purpose I/O.) • TX DIR = 1 (LINTX pin is a general-purpose output.)
		0	*Read:* The output on LINTX is at logic low (0). *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The output on LINTX is at logic high (1). *Write:* The output on LINTX is at logic low (0).
1	RX CLR		Receive pin clear. This bit is effective in LIN or SCI mode. This bit clears the logic to be output on pin LINRX if the following conditions are met: • RX FUNC = 0 (LINRX pin is a general-purpose I/O.) • RX DIR = 1 (LINRX pin is a general-purpose output.)
		0	*Read:* The output on LINRX is at logic low (0). *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The output on LINRX is at logic high (1). *Write:* The output on LINRX is at logic low (0).
0	Reserved	0	Reads return 0. Writes have no effect.

28.13.20 SCI Pin I/O Control Register 6 (SCIPIO6)

Figure 28-48 and Table 28-36 illustrate this register.

Figure 28-48. SCI Pin I/O Control Register 6 (SCIPIO6) [offset = 54h]

31 8
Reserved
R-0

3	2	1	0
Reserved	TX PDR	RX PDR	Reserved
R-0	R/W-0	R/W-0	R-0

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

Table 28-36. SCI Pin I/O Control Register 6 (SCIPIO6) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TX PDR		Transmit pin open drain enable. This bit is effective in LIN or SCI mode. This bit enables open-drain capability in the output pin LINTX, if the following conditions are met: • TX FUNC = 0 (LINTX pin is a general-purpose I/O.) • TX DIR = 1 (LINTX pin is a general-purpose output.)
		0	Open-drain functionality is disabled; the output voltage is V_{OL} or lower if TXOUT = 0, and is V_{OH} or higher if TXOUT = 1.
		1	Open-drain functionality is enabled; the output voltage is V_{OL} or lower if TXOUT = 0, and is high-impedance if TXOUT = 1.
1	RX PDR		Receive pin open drain enable. This bit is effective in LIN or SCI mode. This bit enables open-drain capability in the output pin LINRX, if the following conditions are met: • RX FUNC = 0 (LINRX pin is a general-purpose I/O.) • RX DIR = 1 (LINRX pin is a general-purpose output.)
		0	Open-drain functionality is disabled; the output voltage is V_{OL} or lower if RXOUT = 0, and is V_{OH} or higher if RXOUT = 1.
		1	Open-drain functionality is enabled; the output voltage is V_{OL} or lower if RXOUT = 0, and is high-impedance if RXOUT = 1.
0	Reserved	0	Reads return 0. Writes have no effect.

28.13.21 SCI Pin I/O Control Register 7 (SCIPIO7)

Figure 28-49 and Table 28-37 illustrate this register.

Figure 28-49. SCI Pin I/O Control Register 7 (SCIPIO7) [offset = 58h]

31	8
Reserved	
R-0	

7	3	2	1	0
Reserved		TX PD	RX PD	Reserved
R-0		R/W-n	R/W-n	R-n

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset, Refer to the Terminal Functions in the device datasheet for default pin settings.

Table 28-37. SCI Pin I/O Control Register 7 (SCIPIO7) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TX PD		Transmit pin pull control disable. This bit is effective in LIN or SCI mode. This bit disables pull control capability on the input pin LINTX.
		0	The pull control on the LINTX pin is enabled.
		1	The pull control on the LINTX pin is disabled.
1	RX PD		Receive pin pull control disable. This bit is effective in LIN or SCI mode. This bit disables pull control capability on the input pin LINRX.
		0	Pull control on the LINRX pin is enabled.
		1	Pull control on the LINRX pin is disabled.
0	Reserved		Writes have no effect.

28.13.22 SCI Pin I/O Control Register 8 (SCIPIO8)

Figure 28-50 and Table 28-38 illustrate this register.

Figure 28-50. SCI Pin I/O Control Register 8 (SCIPIO8) [offset = 5Ch]

31	8
Reserved	
R-0	

7	3	2	1	0
Reserved		TX PSL	RX PSL	Reserved
R-0		R/W-n	R/W-n	R-n

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset, Refer to the Terminal Functions in the device datasheet for default pin settings.

Table 28-38. SCI Pin I/O Control Register 8 (SCIPIO8) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TX PSL		Transmit pin pull select. This bit is effective in LIN or SCI mode. This bit selects pull type in the input pin LINTX.
		0	The LINTX pin is a pull down.
		1	The LINTX pin is a pull up.
1	RX PSL		Receive pin pull select. This bit is effective in LIN or SCI mode. This bit selects pull type in the input pin LINRX.
		0	The LINRX pin is a pull down.
		1	The LINRX pin is a pull up.
0	Reserved		Writes have no effect.

28.13.23 LIN Compare Register (LINCOMPARE)

Figure 28-51 and Table 28-39 illustrate this register.

Figure 28-51. LIN Compare Register (LINCOMPARE) [offset = 60h]

31 16
Reserved
R-0

15 10	9 8	7 3	2 0
Reserved	SDEL	Reserved	SBREAK
R-0	R/WP-0	R-0	R/WP-0

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

Table 28-39. LIN Compare Register (LINCOMPARE) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return 0. Writes have no effect.
9-8	SDEL		2-bit synch delimiter compare. These bits are effective in LIN mode only. These bits are used to configure the number of T_{bit} for the synch delimiter in the synch field. The default value is 0.
			The formula to program the value (in T_{bits}) for the synchronization delimiter is:
			$T_{SDEL} = (SDEL + 1)\ T_{bit}$
		0	The synch delimiter has 1 T_{bit}.
		1h	The synch delimiter has 2 T_{bit}.
		2h	The synch delimiter has 3 T_{bit}.
		3h	The synch delimiter has 4 T_{bit}.
7-3	Reserved	0	Reads return 0. Writes have no effect.
2-0	SBREAK		Synch break extend. These bits are effective in LIN mode only. These bits are used to configure the number of T_{bit} for the synch break to extend the minimum 13 T_{bit} break field to a maximum of 20 T_{bit} long.
			Note: The default value is 0, which adds nothing to the automatically generated SYNCH BREAK.
			The formula to program the value (in T_{bits}) for the SYNCH BREAK is:
			$T_{SYNBRK} = 13T_{bit} + (SBREAK \times T_{bit})$
		0	The synch break has no additional T_{bit}.
		1h	The synch break has 1 additional T_{bit}.
		2h	The synch break has 2 additional T_{bit}.
		3h	The synch break has 3 additional T_{bit}.
		4h	The synch break has 4 additional T_{bit}.
		5h	The synch break has 5 additional T_{bit}.
		6h	The synch break has 6 additional T_{bit}.
		7h	The synch break has 7 additional T_{bit}.

28.13.24 LIN Receive Buffer 0 Register (LINRD0)

Figure 28-52 and Table 28-40 illustrate this register.

Figure 28-52. LIN Receive Buffer 0 Register (LINRD0) [offset = 64h]

31 … 24	23 … 16
RD0	RD1
R-0	R-0

15 … 8	7 … 0
RD2	RD3
R-0	R-0

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

Table 28-40. LIN Receive Buffer 0 Register (LINRD0) Field Descriptions

Bit	Field	Value	Description
31-24	RD0	0-FFh	Receive buffer 0. Byte 0 of the response data byte. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding RDy bit field according to the number of bytes received. A read of this byte clears the RXDY byte. **Note: RD<x-1> is equivalent to data byte <x> of the LIN frame.**
23-16	RD1	0-FFh	Receive buffer 1. Byte 1 of the response data byte. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding RDy register according to the number of bytes received.
15-8	RD2	0-FFh	Receive buffer 2. Byte 2 of the response data byte. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding RDy register according to the number of bytes received.
7-0	RD3	0-FFh	Receive buffer 3. Byte 3 of the response data byte. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding RDy register according to the number of bytes received.

28.13.25 LIN Receive Buffer 1 Register (LINRD1)

Figure 28-53 and Table 28-41 illustrate this register.

Figure 28-53. LIN Receive Buffer 1 Register (RD1) [offset = 68h]

31 … 24	23 … 16
RD4	RD5
R-0	R-0

15 … 8	7 … 0
RD6	RD7
R-0	R-0

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

Table 28-41. LIN Receive Buffer 1 Register (RD1) Field Descriptions

Bit	Field	Value	Description
31-24	RD4	0-FFh	Receive buffer 4. Byte 4 of the response data byte. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding register according to the number of bytes received. **Note: RD<x-1> is equivalent to data byte <x> of the LIN frame.**
23-16	RD5	0-FFh	Receive buffer 5. Byte 5 of the response data byte. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding register according to the number of bytes received.
15-8	RD6	0-FFh	Receive buffer 6. Byte 6 of the response data byte. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding register according to the number of bytes received.
7-0	RD7	0-FFh	Receive buffer 7. Byte 7 of the response data byte. Each response data-byte that is received in the SCIRXSHFT register is transferred to the corresponding register according to the number of bytes received.

28.13.26 LIN Mask Register (LINMASK)

Figure 28-54 and Table 28-42 illustrate this register.

Figure 28-54. LIN Mask Register (LINMASK) [offset = 6Ch]

31 – 24	23 – 16
Reserved	RX ID MASK
R-0	R/WL-0

15 – 8	7 – 0
Reserved	TX ID MASK
R-0	R/WL-0

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

Table 28-42. LIN Mask Register (LINMASK) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23-16	RX ID MASK	0-FFh	Receive ID mask. These bits are effective in LIN mode only. This 8-bit mask is used for filtering an incoming ID message and comparing it to the ID-byte. A compare match of the received ID with the RX ID MASK will set the ID RX flag and trigger an ID interrupt if enabled (SET ID INT in SCISETINT). A 0 bit in the mask indicates that bit is compared to the ID-byte. A 1 bit in the mask indicates that bit is filtered and therefore is not used in the compare.
15-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	TX ID MASK	0-FFh	Transmit ID mask. These bits are effective in LIN mode only. This 8-bit mask is used for filtering an incoming ID message and comparing it to the ID-byte. A compare match of the received ID with the TX ID MASK will set the ID TX flag and trigger an ID interrupt if enabled (SET ID INT in SCISETINT). A 0 bit in the mask indicates that bit is compared to the ID-byte. A 1 bit in the mask indicates that bit is filtered and therefore is not used for the compare.

28.13.27 LIN Identification Register (LINID)

Figure 28-55 and Table 28-43 illustrate this register.

Figure 28-55. LIN Identification Register (LINID) [offset = 70h]

31 24	23 16
Reserved	RECEIVED ID
R-0	R-0

15 8	7 0
ID-SLAVETASK BYTE	ID BYTE
R/WL-0	R/WL-0

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

Table 28-43. LIN Identification Register (LINID) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23-16	RECEIVED ID	0-FFh	Received identification. These bits are effective in LIN mode only. This byte contains the current message identifier. During header reception the received ID is copied from the SCIRXSHF register to this byte if there is no ID-parity error and there has been an RX/TX match.
15-8	ID-SLAVETASK BYTE	0-FFh	ID-SlaveTask Byte. These bits are effective in LIN mode only. This field contains the identifier to which the received ID of an incoming header will be compared to decide whether a receive response, a transmit response, or no action needs to be performed by the LIN node when a header with that particular ID is received.
7-0	ID BYTE	0-FFh	ID byte. This field is effective in LIN mode only. This byte is the LIN mode message ID. On a master node, a write to this register by the CPU initiates a header transmission. For a slave task, this byte is used for message filtering when HGEN CTRL = 0.

NOTE: For software compatibility with future LIN modules, the HGEN CTRL bit must be set to 1, the RX ID MASK field must be set to FFh, and the TX ID MASK field must be set to FFh.

28.13.28 LIN Transmit Buffer 0 Register (LINTD0)

Figure 28-56 and Table 28-44 illustrate this register.

Figure 28-56. LIN Transmit Buffer 0 Register (LINTD0) [offset = 74h]

31 24	23 16
TD0	TD1
R/W-0	R/W-0
15 8	7 0
TD2	TD3
R/W-0	R/W-0

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

Table 28-44. LIN Transmit Buffer 0 Register (LINTD0) Field Descriptions

Bit	Field	Value	Description
31-24	TD0	0-FFh	8-Bit transmit buffer 0. Byte 0 to be transmitted is written into this register and then copied to SCITXSHF for transmission. Once byte 0 is written in TD0 buffer, transmission will be initiated. **Note: TD<x-1> is equivalent to data byte <x> of the LIN frame.**
23-16	TD1	0-FFh	8-Bit transmit buffer 1. Byte 1 to be transmitted is written into this register and then copied to SCITXSHF for transmission.
15-8	TD2	0-FFh	8-Bit transmit buffer 2. Byte 2 to be transmitted is written into this register and then copied to SCITXSHF for transmission.
7-0	TD3	0-FFh	8-Bit transmit buffer 3. Byte 3 to be transmitted is written into this register and then copied to SCITXSHF for transmission.

28.13.29 LIN Transmit Buffer 1 Register (LINTD1)

Figure 28-57 and Table 28-45 illustrate this register.

Figure 28-57. LIN Transmit Buffer 1 Register (LINTD1) [offset = 78h]

31 24	23 16
TD4	TD5
R/W-0	R/W-0
15 8	7 0
TD6	TD7
R/W-0	R/W-0

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

Table 28-45. LIN Transmit Buffer 1 Register (LINTD1) Field Descriptions

Bit	Field	Value	Description
31-24	TD4	0-FFh	8-Bit transmit buffer 4. Byte 4 to be transmitted is written into this register and then copied to SCITXSHF for transmission. **Note: TD<x-1> is equivalent to data byte <x> of the LIN frame.**
23-16	TD5	0-FFh	8-Bit transmit buffer 5. Byte 5 to be transmitted is written into this register and then copied to SCITXSHF for transmission.
15-8	TD6	0-FFh	8-Bit transmit buffer 6. Byte 6 to be transmitted is written into this register and then copied to SCITXSHF for transmission.
7-0	TD7	0-FFh	8-Bit transmit buffer 7. Byte 7 to be transmitted is written into this register and then copied to SCITXSHF for transmission.

28.13.30 Maximum Baud Rate Selection Register (MBRS)

Figure 28-58 and Table 28-46 illustrate this register.

Figure 28-58. Maximum Baud Rate Selection Register (MBRS) [offset = 7Ch]

31	16
Reserved	
R-0	

15 13	12 0
Reserved	MBR
R-0	R/WL-0DACh

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

Table 28-46. Maximum Baud Rate Selection Register (MBRS) Field Descriptions

Bit	Field	Value	Description
31-13	Reserved	0	Reads return 0. Writes have no effect.
12-0	MBR	0-1FFFh	Maximum baud rate prescaler. This bit is effective in LIN mode only. This 13-bit prescaler is used during the synchronization phase (see Section 28.7.5.2) of a slave module if the ADAPT bit is set. In this way, a SCI/LIN slave using an automatic or select bit rate modes detects any LIN bus legal rate automatically. The MBR value should be programmed to allow a maximum baud rate that is not more than 10% above the expected operating baud rate in the LIN network. Otherwise, a 0x00 data byte could mistakenly be detected as a sync break. The default value for a 70Mhz VCLK is 0xDAC. This MBR prescaler is used by the wake-up and idle time counters for a constant expiration time relative to a 20 kHz rate. $\text{MBR} = \frac{0.9\text{x VCLK}}{\text{maxbaudrate}}$ (56)

28.13.31 Input/Output Error Enable (IODFTCTRL) Register

All the bits in the IODFTCTRL register are used in IODFT (I/O design for test) mode only. Figure 28-59 and Table 28-47 illustrate this register. After the basic SCI/LIN module configuration, enable the required Error mode to be created followed by IODFT Key enable.

NOTE: 1) All the bits are used in IODFT mode only.

2) Each IODFT are expected to be checked individually.

3) ISFE Error will not be Flagged during IODFT mode.

Figure 28-59. Input/Output Error Enable Register (IODFTCTRL) [offset = 90h]

31	30	29	28	27	26	25	24
BEN	PBEN	CEN	ISFE	Reserved	FEN	PEN	BRKDT ENA
R/WL-0	R/WL-0	R/WL-0	R/WL-0	R-0	R/W-0	R/WC-0	R/WC-0

23		21	20	19	18		16
Reserved			PIN SAMPLE MASK		TX SHIFT		
R-0			R/W-0		R/W-0		

15			12	11			8
Reserved				IODFTENA			
R-0				R/WP-5h			

7					2	1	0
Reserved						LPB ENA	RXP ENA
R-0						R/WP-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WL = Write in LIN mode only; WC = Write in SCI-compatible mode only; WP = Write in privilege mode only; -*n* = value after reset

Table 28-47. Input/Output Error Enable Register (IODFTCTRL) Field Descriptions

Bit	Field	Value	Description
31	BEN		Bit error enable. This bit is effective in LIN mode only. This bit is used to create a bit error.
		0	No bit error is created.
		1	The bit received is ORed with 1 and passed to the bit monitor circuitry.
30	PBEN		Physical bus error enable. This bit is effective in LIN mode only. This bit is used to create a physical bus error.
		0	No error is created.
		1	The bit received during synch break field transmission is ORed with 1 and passed to the bit monitor circuitry.
29	CEN		Checksum error enable. This bit is effective in LIN mode only. This bit is used to create a checksum error.
		0	No error is created.
		1	The polarity of the CTYPE (checksum type) in the receive checksum calculator is changed so that a checksum error is occurred.
28	ISFE		Inconsistent synch field (ISF) error enable. This bit is effective in LIN mode only. This bit is used to create an ISF error.
		0	No error is created.
		1	The bit widths in the synch field are varied so that the ISF check fails and the error flag is set.
27	Reserved	0	Reads return 0. Writes have no effect.
26	FEN		Frame error enable. This bit is used to create a frame error.
		0	No error is created.
		1	The stop bit received is ANDed with 0 and passed to the stop bit check circuitry.

Table 28-47. Input/Output Error Enable Register (IODFTCTRL) Field Descriptions (continued)

Bit	Field	Value	Description
25	PEN		Parity error enable. This bit is effective in SCI-compatible mode only. This bit is used to create a parity error.
		0	No parity error occurs.
		1	The parity bit received is toggled so that a parity error occurs.
24	BRKDT ENA		Break detect error enable. This bit is effective in SCI-compatible mode only. This bit is used to create a BRKDT error.
		0	No error is created.
		1	The stop bit of the frame is ANDed with 0 and passed to the RSM so that a frame error occurs. Then the RX pin is forced to continuous low for 10 T_{bit} so that a BRKDT error occurs.
32-21	Reserved	0	Reads return 0. Writes have no effect.
20-19	PIN SAMPLE MASK		Pin sample mask. These bits define the sample number at which the TX pin value that is being transmitted will be inverted to verify the receive pin samples majority detection circuitry. **Note: In IODFT mode testing for pin_sample mask must be done with prescalar P programmed greater than 2 (P > 2).**
		0	No mask is used.
		1h	Invert the TX Pin value at TBIT_CENTER.
		2h	Invert the TX Pin value at TBIT_CENTER + SCLK.
		3h	Invert the TX Pin value at TBIT_CENTER + 2 SCLK.
18-16	TX SHIFT		Transmit shift. These bits define the amount by which the value on TX pin is delayed so that the value on the RX pin is asynchronous. This feature is not applicable to the start bit.
		0	No delay occurs.
		1h	The value is delayed by 1 SCLK.
		2h	The value is delayed by 2 SCLK.
		3h	The value is delayed by 3 SCLK.
		4h	The value is delayed by 4 SCLK.
		5h	The value is delayed by 5 SCLK.
		6h	The value is delayed by 6 SCLK.
		7h	The value is delayed by 7 SCLK.
15-12	Reserved	0	Reads return 0. Writes have no effect.
11-8	IODFTENA		IODFT enable key. Write access permitted in Privilege mode only.
		Ah	IODFT is enabled.
		All Others	IODFT is disabled.
7-2	Reserved	0	Reads return 0. Writes have no effect.
1	LPB ENA		Module loopback enable. Write access permitted in Privilege mode only. **Note: In analog loopback mode the complete communication path through the I/Os can be tested, whereas in digital loopback mode the I/O buffers are excluded from this path.**
		0	Digital loopback is enabled.
		1	Analog loopback is enabled in module I/O DFT mode when IODFTENA = 1010.
0	RXP ENA		Module analog loopback through receive pin enable. Write access permitted in Privilege mode only. This bit defines whether the I/O buffers for the transmit or the receive pin are included in the communication path (in analog loopback mode).
		0	Analog loopback through the transmit pin is enabled.
		1	Analog loopback through the receive pin is enabled.

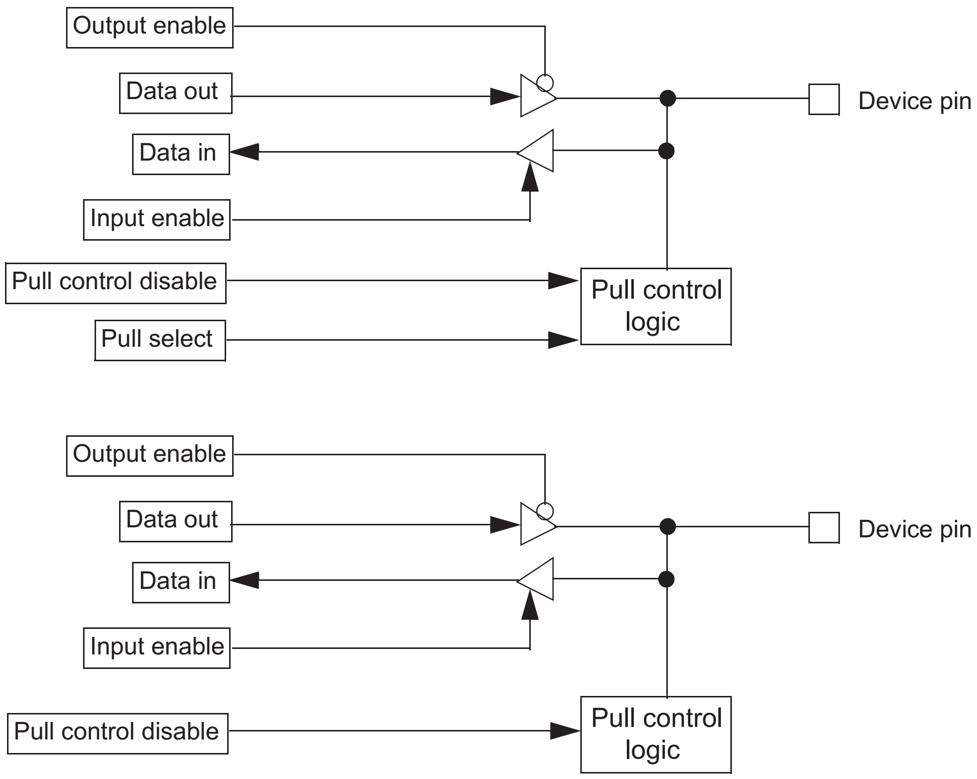
28.14 GPIO Functionality

The following sections apply to all device pins that can be configured as functional or general-purpose I/O pins.

28.14.1 GPIO Functionality

Figure 28-60 illustrates the GPIO functionality.

Figure 28-60. GPIO Functionality



28.14.2 Under Reset

The following apply if a device is under reset:

- Pull control. The reset pull control on the pins is enabled.
- Input buffer. The input buffer is enabled.
- Output buffer. The output buffer is disabled.

28.14.3 Out of Reset

The following apply if the device is out of reset:

- Pull control. The pull control is enabled by clearing the PD (pull control disable) bit in the SCIPIO7 register (Section 28.13.21). In this case, if the PSL (pull select) bit in the SCIPIO8 register (Section 28.13.22) is set, the pin will have a pull-up. If the PSL bit is cleared, the pin will have a pull-down. If the PD bit is set in the control register, there is no pull-up or pull-down on the pin.
- Input buffer. The input buffer is permanently enabled in this device.

NOTE: The pull-disable logic depends on the pin direction. It is independent of whether the device is in I/O or functional mode. If the pin is configured as output or transmit, then the pulls are disabled automatically.

- Output buffer. A pin can be driven as an output pin if the TX DIR bit is set in the pin direction control register (SCIPIO1; Section 28.13.15) AND the open-drain feature is not enabled in the SCIPIO6 register (Section 28.13.20).

28.14.4 Open-Drain Feature Enabled on a Pin

The following apply if the open-drain feature is enabled on a pin:

- The output buffer is enabled if a low signal is being driven on to the pin.
- The output buffer is disabled (the direction control signal DIR is internally forced low) if a high signal is being driven on to the pin.

NOTE: The open-drain feature is available only in I/O mode (SCIPIO0; Section 28.13.14).

28.14.5 Summary

The behavior of the input buffer, output buffer, and the pull control is summarized in Table 28-48.

Table 28-48. Input Buffer, Output Buffer, and Pull Control Behavior as GPIO Pins[1]

Device under Reset?	Pin Direction (DIR) [2]	Pull Disable (PULDIS) [3]	Pull Select (PULSEL) [4]	Pull Control	Output Buffer	Input Buffer
Yes	X	X	X	Enabled	Disabled	Enabled
No	0	0	0	Pull down	Disabled	Enabled
No	0	0	1	Pull up	Disabled	Enabled
No	0	1	0	Disabled	Disabled	Enabled
No	0	1	1	Disabled	Disabled	Enabled
No	1	X	X	Disabled	Enabled	Enabled

[1] X = Don't care
[2] DIR = 0 for input, = 1 for output
[3] PULDIS = 0 for enabling pull control, = 1 for disabling pull control
[4] PULSEL= 0 for pull-down functionality, = 1 for pull-up functionality

Chapter 29
SPNU514C–March 2018

Serial Communication Interface (SCI) Module

This chapter contains the description of the serial communication interface (SCI) module.

Topic **Page**

29.1 Introduction 1609
29.2 SCI Communication Formats 1611
29.3 SCI Interrupts 1616
29.4 SCI DMA Interface 1619
29.5 SCI Configurations 1620
29.6 SCI Low-Power Mode 1621
29.7 SCI Control Registers 1623
29.8 GPIO Functionality 1653

29.1 Introduction

The SCI module is a universal asynchronous receiver-transmitter that implements the standard nonreturn to zero format. The SCI can be used to communicate, for example, through an RS-232 port or over a K-line.

29.1.1 SCI Features

The following are the features of the SCI module:

- Standard universal asynchronous receiver-transmitter (UART) communication
- Supports full- or half-duplex operation
- Standard nonreturn to zero (NRZ) format
- Double-buffered receive and transmit functions
- Supports two individually enabled interrupt lines: level 0 and level 1
- Configurable frame format of 3 to 13 bits per character based on the following:
  - Data word length programmable from one to eight bits
  - Additional address bit in address-bit mode
  - Parity programmable for zero or one parity bit, odd or even parity
  - Stop programmable for one or two stop bits
- Asynchronous communication mode with no CLK pin
- Two multiprocessor communication formats allow communication between more than two devices
- Sleep mode is available to free CPU resources during multiprocessor communication and then wake up to receive an incoming message
- The 24-bit programmable baud rate supports 2^{24} different baud rates provide high accuracy baud rate selection
- Capability to use Direct Memory Access (DMA) for transmit and receive data
- Four error flags and Five status flags provide detailed information regarding SCI events
- Two external pins: SCIRX and SCITX

NOTE: SCI module does not support UART Hardware Flow Control. This feature can be implemented in Software using a General Purpose I/O pin.

29.1.2 Block Diagram

Three Major components of the SCI Module are:

- Transmitter
- Baud Clock Generator
- Receiver

Transmitter (TX) contains two major registers to perform double buffering:

- The transmitter data buffer register (SCITD) contains data loaded by the CPU to be transferred to the shift register for transmission.
- The transmitter shift register (SCITXSHF) loads data from the data buffer (SCITD) and shifts data onto the SCITX pin, one bit at a time.

Baud Clock Generator

- A programmable baud generator produces a baud clock scaled from VCLK.

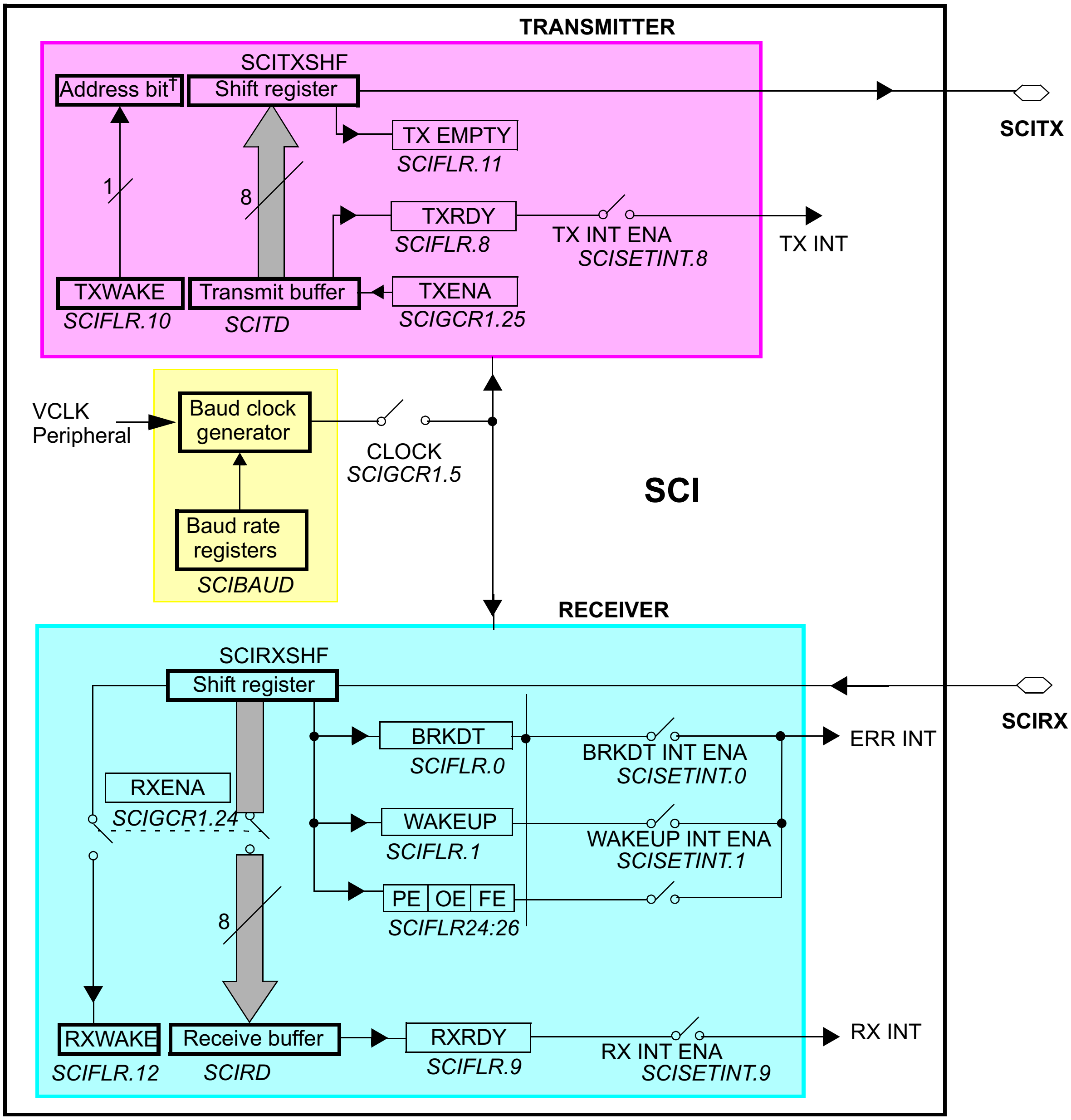
Receiver (RX) contains two major registers to perform double buffering:

- The receiver shift register (SCIRXSHF) shifts data in from the SCIRX pin one bit at a time and transfers completed data into the receive data buffer.
- The receiver data buffer register (SCIRD) contains received data transferred from the receiver shift register

The SCI receiver and transmitter are double-buffered, and each has its own separate enable and interrupt bits. The receiver and transmitter may each be operated independently or simultaneously in full duplex mode.

To ensure data integrity, the SCI checks the data it receives for breaks, parity, overrun, and framing errors. The bit rate (baud) is programmable to over 16 million different rates through a 24-bit baud-select register. Figure 29-1 shows the detailed SCI block diagram.

Figure 29-1. Detailed SCI Block Diagram



29.2 SCI Communication Formats

The SCI module can be configured to meet the requirements of many applications. Because communication formats vary depending on the specific application, many attributes of the SCI are user configurable. The list below describes these configuration options:

- SCI Frame format
- SCI Timing modes
- SCI Baud rate
- SCI Multiprocessor modes

29.2.1 SCI Frame Formats

The SCI uses a programmable frame format. All frames consist of the following:

- One start bit
- One to eight data bits
- Zero or one address bit
- Zero or one parity bit
- One or two stop bits

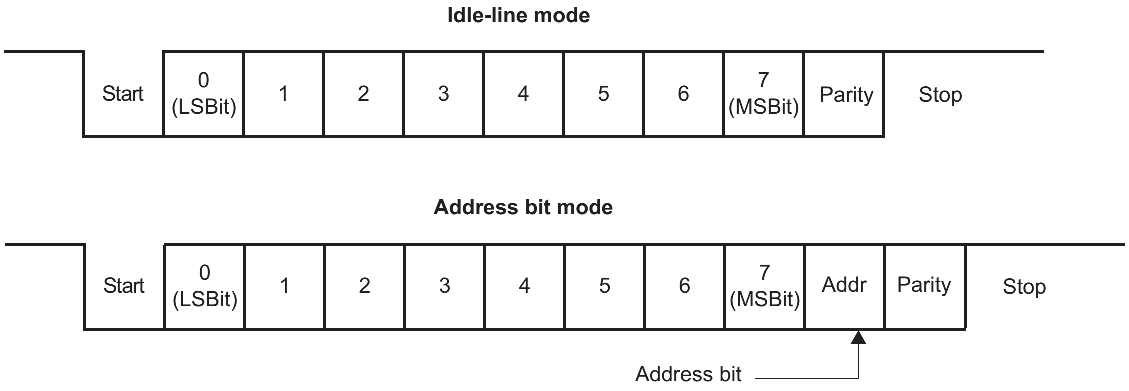
The frame format for both the transmitter and receiver is programmable through the bits in the SCIGCR1 register. Both receive and transmit data is in nonreturn to zero (NRZ) format, which means that the transmit and receive lines are at logic high when idle. Each frame transmission begins with a start bit, in which the transmitter pulls the SCI line low (logic low). Following the start bit, the frame data is sent and received least significant bit first (LSB).

An address bit is present in each frame if the SCI is configured to be in address-bit mode but is not present in any frame if the SCI is configured for idle-line mode. The format of frames with and without the address bit is illustrated in Figure 29-2.

A parity bit is present in every frame when the PARITY ENA bit is set. The value of the parity bit depends on the number of one bits in the frame and whether odd or even parity has been selected via the PARITY ENA bit. Both examples in Figure 29-2 have parity enabled.

All frames include one stop bit, which is always a high level. This high level at the end of each frame is used to indicate the end of a frame to ensure synchronization between communicating devices. Two stop bits are transmitted if the STOP bit in SCIGCR1 register is set. The examples shown in Figure 29-2 use one stop bit per frame.

Figure 29-2. Typical SCI Data Frame Formats



29.2.2 SCI Timing Mode

The SCI can be configured to use asynchronous or isosynchronous timing using TIMING MODE bit in SCIGCR1 register.

29.2.2.1 Asynchronous Timing Mode

The asynchronous timing mode uses only the receive and transmit data lines to interface with devices using the standard universal asynchronous receiver- transmitter (UART) protocol.

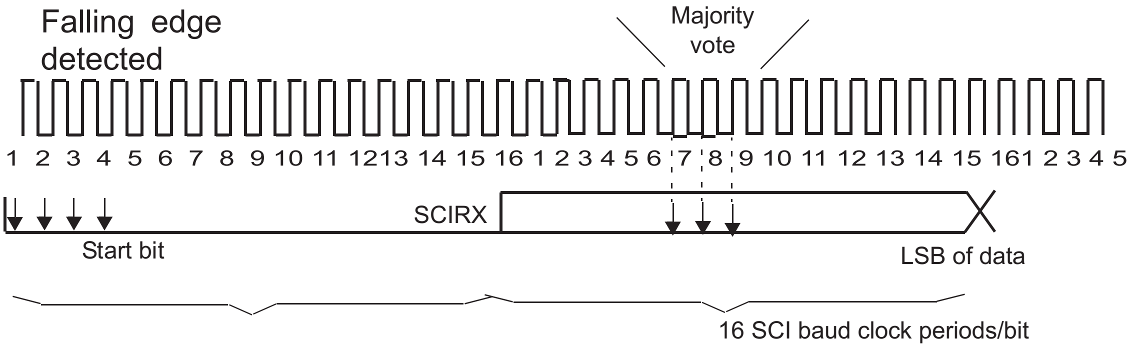
In the asynchronous timing mode, each bit in a frame has a duration of 16 SCI baud clock periods. Each bit therefore consists of 16 samples (one for each clock period). When the SCI is using asynchronous mode, the baud rates of all communicating devices must match as closely as possible. Receive errors result from devices communicating at different baud rates.

With the receiver in the asynchronous timing mode, the SCI detects a valid start bit if the first four samples after a falling edge on the SCIRX pin are of logic level 0. As soon as a falling edge is detected on SCIRX, the SCI assumes that a frame is being received and synchronizes itself to the bus.

To prevent interpreting noise as Start bit SCI expects SCIRX line to be low for at least four contiguous SCI baud clock periods to detect a valid start bit. The bus is considered idle if this condition is not met. When a valid start bit is detected, the SCI determines the value of each bit by sampling the SCIRX line value during the seventh, eighth, and ninth SCI baud clock periods. A majority vote of these three samples is used to determine the value stored in the SCI receiver shift register. By sampling in the middle of the bit, the SCI reduces errors caused by propagation delays and rise and fall times and data line noises. Figure 29-3 illustrates how the receiver samples a start bit and a data bit in asynchronous timing mode.

The transmitter transmits each bit for a duration of 16 SCI baud clock periods. During the first clock period for a bit, the transmitter shifts the value of that bit onto the SCITX pin. The transmitter then holds the current bit value on SCITX for 16 SCI baud clock periods.

Figure 29-3. Asynchronous Communication Bit Timing



29.2.2.2 Isosynchronous Timing Mode

In isosynchronous timing mode, each bit in a frame has a duration of exactly 1 baud clock period and therefore consists of a single sample. With this timing configuration, the transmitter and receiver are required to make use of the SCICLK pin to synchronize communication with other SCI. **This mode is not fully supported on this device because SCICLK pin is not available.**

29.2.3 SCI Baud Rate

The SCI has an internally generated serial clock determined by the peripheral VCLK and the prescalers BAUD. The SCI uses the 24-bit integer prescaler BAUD value of the BRS register to select the required baud rates.

In asynchronous timing mode, the SCI generates a baud clock according to the following formula:

$$\textit{Asynchronous baud value} = \frac{\text{VCLK Frequency}}{16 * (\text{BAUD} + 1)}$$

For BAUD = 0,

$$\textit{Asynchronous baud value} = \frac{\text{VCLK Frequency}}{32} \tag{57}$$

In isosynchronous timing mode, the SCI generates a baud clock according to the following formula:

$$\textit{Isosynchronous baud value} = \frac{\text{VCLK Frequency}}{\text{BAUD} + 1}$$

For BAUD = 0,

$$\textit{Isosynchronous baud value} = \frac{\text{VCLK Frequency}}{32} \tag{58}$$

29.2.4 SCI Multiprocessor Communication Modes

In some applications, the SCI may be connected to more than one serial communication device. In such a multiprocessor configuration, several frames of data may be sent to all connected devices or to an individual device. In the case of data sent to an individual device, the receiving devices must determine when they are being addressed. When a message is not intended for them, the devices can ignore the following data. When only two devices make up the SCI network, addressing is not needed, so multiprocessor communication schemes are not required.

SCI supports two multiprocessor Communication Modes which can be selected using COMM MODE bit:

- Idle-Line Mode
- Address Bit Mode

When the SCI is not used in a multiprocessor environment, software can consider all frames as data frames. In this case, the only distinction between the idle-line and address-bit modes is the presence of an extra bit (the address bit) in each frame sent with the address-bit protocol.

The SCI allows full-duplex communication where data can be sent and received via the transmit and receive pins simultaneously. However, the protocol used by the SCI assumes that only one device transmits data on the same bus line at any one time. No arbitration is done by the SCI.

NOTE: Avoid Transmitting Simultaneously on the Same Serial Bus

The system designer must ensure that devices connected to the same serial bus line do not attempt to transmit simultaneously. If two devices are transmitting different data, the resulting bus conflict could damage the device..

29.2.4.1 Idle-Line Multiprocessor Modes

In idle-line multiprocessor mode, a frame that is preceded by an idle period (10 or more idle bits) is an address frame. A frame that is preceded by fewer than 10 idle bits is a data frame. Figure 29-4 illustrates the format of several blocks and frames with idle-line mode.

There are two ways to transmit an address frame using idle-line mode:

Method 1: In software, deliberately leave an idle period between the transmission of the last data frame of the previous block and the address frame of the new block.

Method 2: Configure the SCI to automatically send an idle period between the last data frame of the previous block and the address frame of the new block.

Although Method 1 is only accomplished by a delay loop in software, Method 2 can be implemented by using the transmit buffer and the TXWAKE bit in the following manner:

Step 1 : Write a 1 to the TXWAKE bit.

Step 2 : Write a dummy data value to the SCITD register. This triggers the SCI to begin the idle period as soon as the transmitter shift register is empty.

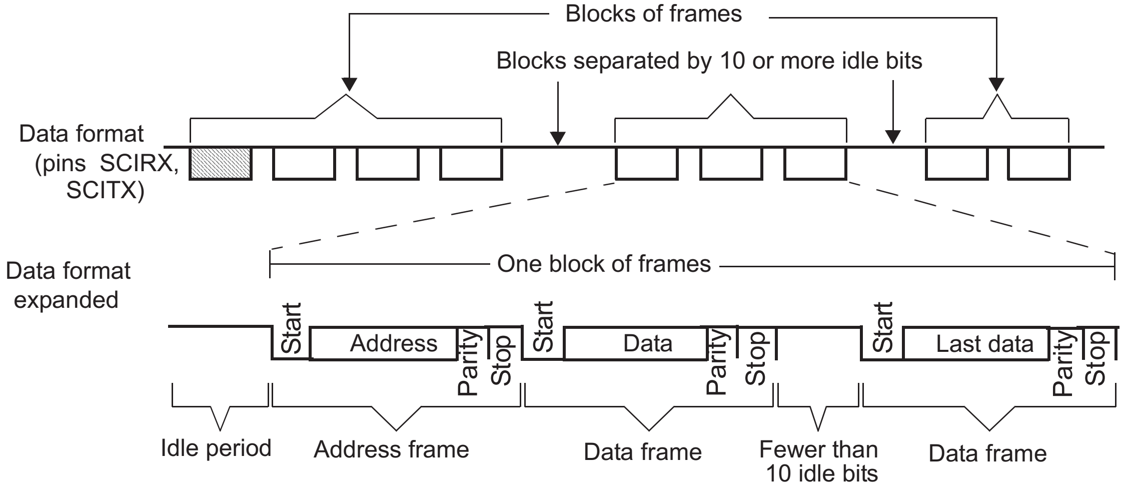
Step 3 : Wait for the SCI to clear the TXWAKE flag.

Step 4 : Write the address value to SCITD.

As indicated by Step 3, software should wait for the SCI to clear the TXWAKE bit. However, the SCI clears the TXWAKE bit at the same time it sets TXRDY (that is, transfers data from SCITD into SCITXSHF). Therefore, if the TX INT ENA bit is set, the transfer of data from SCITD to SCITXSHF causes an interrupt to be generated at the same time that the SCI clears the TXWAKE bit. If this interrupt method is used, software is not required to poll the TXWAKE bit waiting for the SCI to clear it.

When idle-line multiprocessor communications are used, software must ensure that the idle time exceeds 10 bit periods before addresses (using one of the methods mentioned above), and software must also ensure that data frames are written to the transmitter quickly enough to be sent without a delay of 10 bit periods between frames. Failure to comply with these conditions will result in data interpretation errors by other devices receiving the transmission.

Figure 29-4. Idle-Line Multiprocessor Communication Format



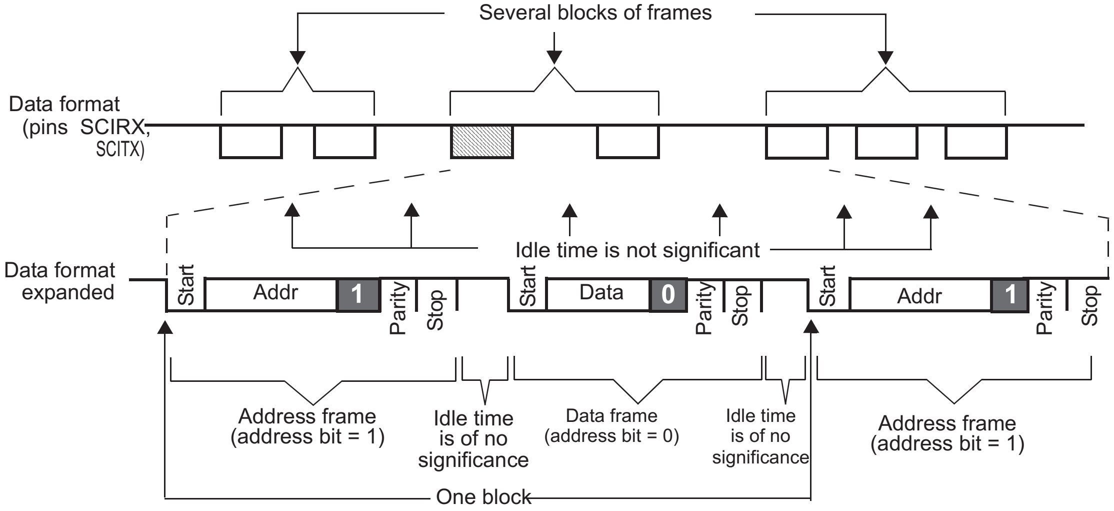
29.2.4.2 Address-Bit Multiprocessor Mode

In the address-bit protocol, each frame has an extra bit immediately following the data field called an address bit. A frame with the address bit set to 1 is an address frame; a frame with the address bit set to 0 is a data frame. The idle period timing is irrelevant in this mode. Figure 29-5 illustrates the format of several blocks and frames with the address-bit mode.

When address-bit mode is used, the value of the TXWAKE bit is the value sent as the address bit. To send an address frame, software must set the TXWAKE bit. This bit is cleared as the contents of the SCITD are shifted from the TXWAKE register so that all frames sent are data except when the TXWAKE bit is written as a 1.

No dummy write to SCITD is required before an address frame is sent in address-bit mode. The first byte written to SCITD after the TXWAKE bit is written to 1 is transmitted with the address bit set when address-bit mode is used.

Figure 29-5. Address-Bit Multiprocessor Communication Format



29.3 SCI Interrupts

The SCI module has two interrupt lines, level 0 and level 1, to the vectored interrupt manager (VIM) module (see Figure 29-6). Two offset registers SCIINTVECT0 and SCIINTVECT1 determine which flag triggered the interrupt according to the respective priority encoders. Each interrupt condition has a bit to enable and disable the interrupt in the SCISETINT and SCICLRINT registers, respectively.

Each interrupt also has a bit that can be set as interrupt level 0 (INT0) or as interrupt level 1 (INT1). By default, interrupts are in interrupt level 0. SCISETINTLVL sets a given interrupt to level1. SCICLEARINTLVL resets a given interrupt level to the default level 0.

The interrupt vector registers SCIINTVECT0 and SCIINTVECT1 return the vector of the pending interrupt line INT0 or INT1. If more than one interrupt is pending, the interrupt vector register holds the highest priority interrupt.

Figure 29-6. General Interrupt Scheme

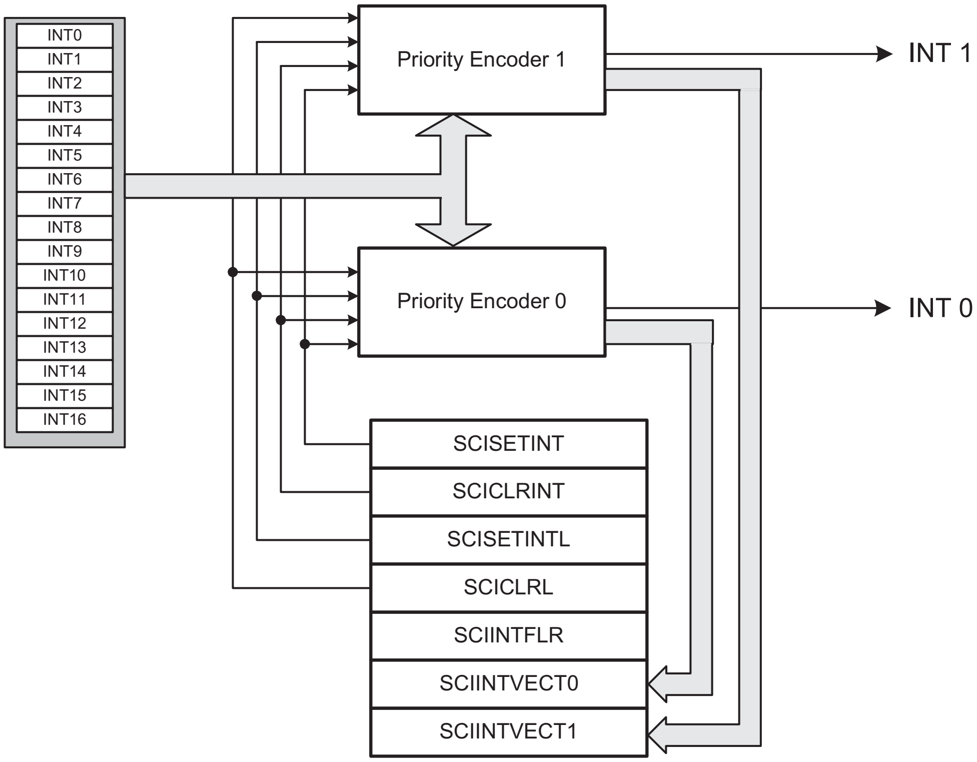
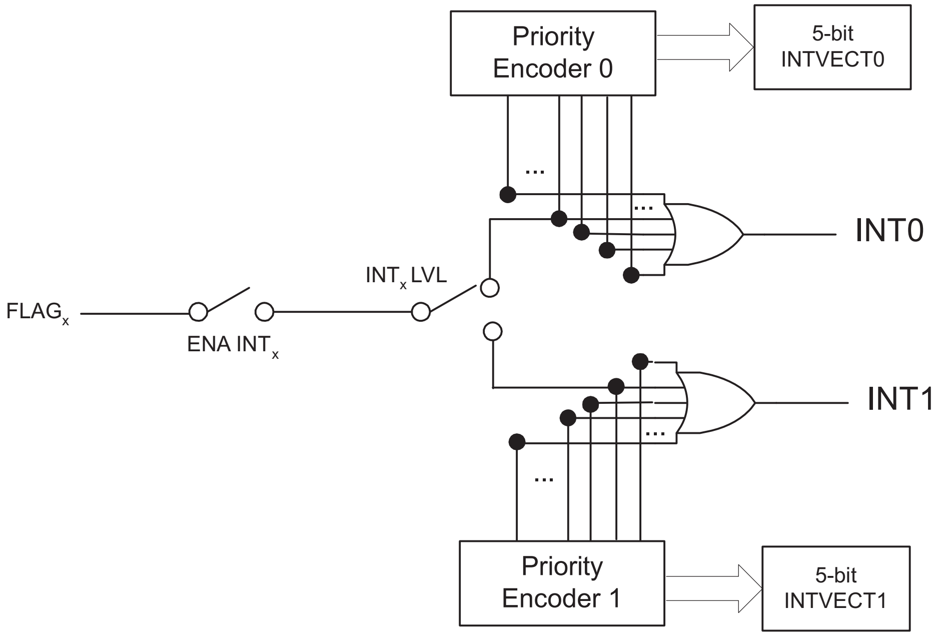


Figure 29-7. Interrupt Generation for Given Flags



29.3.1 Transmit Interrupt

To use transmit interrupt functionality, SET TX INT bit must be enabled and SET TX DMA bit must be cleared. The transmit ready (TXRDY) flag is set when the SCI transfers the contents of SCITD to the shift register, SCITXSHF. The TXRDY flag indicates that SCITD is ready to be loaded with more data. In addition, the SCI sets the TX EMPTY bit if both the SCITD and SCITXSHF registers are empty. If the SET TX INT bit is set, then a transmit interrupt is generated when the TXRDY flag goes high. Transmit Interrupt is not generated immediately after setting the SET TX INT bit unlike transmit DMA request. Transmit Interrupt is generated only after the first transfer from SCITD to SCITXSHF, that is first data has to be written to SCITD by the User before any interrupt gets generated. To transmit further data the user can write data to SCITD in the transmit Interrupt service routine.

Writing data to the SCITD register clears the TXRDY bit. When this data has been moved to the SCITXSHF register, the TXRDY bit is set again. The interrupt request can be suspended by setting the CLR TX INT bit; however, when the SET TX INT bit is again set to 1, the TXRDY interrupt is asserted again. The transmit interrupt request can be eliminated until the next series of values is written to SCITD, by disabling the transmitter via the TXENA bit, by a software reset SWnRST, or by a device hardware reset.

29.3.2 Receive Interrupt

The receive ready (RXRDY) flag is set when the SCI transfers newly received data from SCIRXSHF to SCIRD. The RXRDY flag therefore indicates that the SCI has new data to be read. Receive interrupts are enabled by the SET RX INT bit. If the SET RX INT is set when the SCI sets the RXRDY flag, then a receive interrupt is generated. The received data can be read in the Interrupt Service routine.

On a device with both SCI and a DMA controller, the bits SET RX DMA ALL and SET RX DMA must be cleared to select interrupt functionality.

29.3.3 WakeUp Interrupt

SCI sets the WAKEUP flag if bus activity on the RX line either prevents power-down mode from being entered, or RX line activity causes an exit from power-down mode. If enabled (SET WAKEUP INT), wakeup interrupt is triggered once WAKEUP flag is set.

29.3.4 Error Interrupts

The following error detection features are supported with Interrupt by the SCI module:

- Parity errors (PE)
- Frame errors (FE)
- Break Detect errors (BRKDT)
- Overrun errors (OE)

If any of these errors (PE, FE, BRKDT, OE) is flagged, an interrupt for the flagged errors will be generated if enabled. A message is valid for both the transmitter and the receiver if there is no error detected until the end of the frame. Each of these flags is located in the receiver status (SCIFLR) register. Further details on these flags are explained in SCIFLR register description.

The SCI module supports the following 7 interrupts as listed in Table 29-1.

Table 29-1. SCI Interrupts

Offset [1]	Interrupt
0	Reserved
1	Wakeup
2	Reserved
3	Parity error
4	Reserved
5	Reserved
6	Frame error
7	Break detect error
8	Reserved
9	Overrun error
10	Reserved
11	Receive
12	Transmit
13-15	Reserved

[1] Offset 1 is the highest priority. Offset 16 is the lowest priority.

29.4 SCI DMA Interface

DMA requests for receive (RXDMA request) and transmit (TXDMA request) are available for the SCI module. Refer to the DMA module chapter for DMA module configurations.

29.4.1 Receive DMA Requests

This DMA functionality is enabled/disabled by the CPU using the SET RX DMA/CLR RX DMA bits, respectively.

The receiver DMA request is set when a frame is received successfully and DMA functionality has been previously enabled. The RXRDY flag is set when the SCI transfers newly received data from the SCIRXSHF register to the SCIRD buffer. The RXRDY flag therefore indicates that the SCI has new data to be read. Receive DMA requests are enabled by the SET RX INT bit.

Parity, overrun, break detect, wake-up, and framing errors generate an error interrupt request immediately upon detection, if enabled, even if the device is in the process of a DMA data transfer. The DMA transfer is postponed until the error interrupt is served. The error interrupt can delete this particular DMA request by reading the receive buffer.

In multiprocessor mode, the SCI can generate receiver interrupts for address frames and DMA requests for data frames. This is controlled by an extra select bit SET RX DMA ALL.

If the SET RX DMA ALL bit is set and the SET RX DMA bit is set when the SCI sets the RXRDY flag, then a receive DMA request is generated for address and data frames.

If the SET RX DMA ALL bit is cleared and the SET RX DMA bit is set when the SCI sets the RXRDY flag upon receipt of a data frame, then a receive DMA request is generated. Receive interrupt requests are generated for address frames.

In multiprocessor mode with the SLEEP bit set, no DMA is generated for received data frames. The software must clear the SLEEP bit before data frames can be received. Table 29-2 specifies the bit values for DMA requests in multiprocessor modes.

Table 29-2. DMA and Interrupt Requests in Multiprocessor Modes

SET RX INT	SET RX DMA	SET RX DMA ALL	ADDR FRAME INT	ADDR FRAME DMA	DATA FRAME INT	DATA FRAME DMA
0	0	x	N	N	N	N
0	1	0	Y	N	N	Y
0	1	1	N	Y	N	Y
1	0	x	Y	N	Y	N
1	1	0	Y	N	Y	Y
1	1	1	Y	Y	Y	Y

In multiprocessor mode, the SCI can generate receiver interrupts for address frames and DMA requests for data frames or DMA requests for both. This is controlled by the SET RX DMA ALL bit.

In multiprocessor mode with the SLEEP bit set, no DMA is generated for received data frames. The software must clear the SLEEP bit before data frames can be received.

29.4.2 Transmit DMA Requests

DMA functionality is enabled and disabled by the CPU with the SET TX DMA and CLR TX DMA bits, respectively.

The TXRDY flag is set when the SCI transfers the contents of SCITD to SCITXSHF. The TXRDY flag indicates that SCITD is ready to be loaded with more data. In addition, the SCI sets the TX EMPTY bit if both the SCITD and SCITXSHF registers are empty.

Transmit DMA requests are enabled by the setting SET TX DMA and SET TX INT bits. If the SET TX DMA bit is set, then a TX DMA request is sent to the DMA when data is written to SCITD and TXRDY is set. The DMA will write the first byte to the transmit buffer.

29.5 SCI Configurations

Before the SCI sends or receives data, its registers should be properly configured. Upon power-up or a system-level reset, each bit in the SCI registers is set to a default state. The registers are writable only after the RESET bit in the SCIGCR0 register is set to 1. Of particular importance is the SWnRST bit in the SCIGCR1 register. The SWnRST is an active-low bit initialized to 0 and keeps the SCI in a reset state until it is programmed to 1. Therefore, all SCI configuration should be completed before a 1 is written to the SWnRST bit.

The following list details the configuration steps that software should perform prior to the transmission or reception of data. As long as the SWnRST bit is cleared to 0 the entire time that the SCI is being configured, the order in which the registers are programmed is not important.

- Enable SCI by setting the RESET bit to 1.
- Clear the SWnRST bit to 0 before SCI is configured.
- Select the desired frame format by programming the SCIGCR1 register.
- Set both the RX FUNC and TX FUNC bits in SCIPIO0 to 1 to configure the SCIRX and SCITX pins for SCI functionality.
- Select the baud rate to be used for communication by programming the BRS register.
- Set the CLOCK bit in SCIGCR1 to 1 to select the internal clock.
- Set the CONT bit in SCIGCR1 to 1 to make SCI not halt for an emulation breakpoint until its current reception or transmission is complete (this bit is used only in an emulation environment).
- Set LOOP BACK bit in SCIGCR1 to 1 to connect the transmitter to the receiver internally (this feature is used to perform a self-test).
- Set the RXENA bit in SCIGCR1 to 1, if data is to be received.
- Set the TXENA bitin SCIGCR1 to 1, if data is to be transmitted.
- Set the SWnRST bit to 1 after SCI is configured.
- Perform receiving or transmitting data (see Section 29.5.1 and Section 29.5.2).

29.5.1 Receiving Data

The SCI receiver is enabled to receive messages if both the RX FUNC bit and the RXENA bit are set to 1. If the RX FUNC bit is not set, the SCIRX pin functions as a general-purpose I/O pin rather than as an SCI function pin. After a valid idle period is detected, data is automatically received as it arrives on the SCIRX pin.

SCI sets the RXRDY bit when it transfers newly received data from SCIRXSHF to SCIRD. The SCI clears the RXRDY bit after the new data in SCIRD has been read. Also, as data is transferred from SCIRXSHF to SCIRD, the SCI sets the FE, OE, or PE flags if any of these error conditions were detected in the received data. These error conditions are supported with configurable interrupt capability. The wake-up and break-detect status bits are also set if one of these errors occurs, but they do not necessarily occur at the same time that new data is being loaded into SCIRD.

You can receive data by:

1. Polling Receive Ready Flag
2. Receive Interrupt
3. DMA

In polling method, software can poll for the RXRDY bit and read the data from SCIRD register once RXRDY is set high. The CPU is unnecessarily overloaded by selecting the polling method. To avoid this, you can use either the interrupt or DMA method. To use the interrupt method, the SET RX INT bit is set. To use the DMA method, the SET RX DMA bit is set. Either an interrupt or a DMA request is generated the moment the RXRDY bit is set.

29.5.2 Transmitting Data

The SCI transmitter is enabled if both the TX FUNC bit and the TXENA bit are set to 1. If the TX FUNC bit is not set, the SCITX pin functions as a general-purpose I/O pin rather than as an SCI function pin. Any value written to the SCITD before TXENA is set to 1 is not transmitted. Both of these control bits allow for the SCI transmitter to be held inactive independently of the receiver.

SCI waits for data to be written to SCITD, transfers it to SCITXSHF, and transmits the data. The TXRDY and TX EMPTY bits indicate the status of the transmit buffers. That is, when the transmitter is ready for data to be written to SCITD, the TXRDY bit is set. Additionally, if both SCITD and SCITXSHF are empty, then the TX EMPTY bit is also set.

You can transmit data by:

1. Polling Transmit Ready Flag
2. Transmit Interrupt
3. DMA

In polling method, software can poll for the TXRDY bit to go high before writing the data to the SCITD register. The CPU is unnecessarily overloaded by selecting the polling method. To avoid this, you can use either the interrupt or DMA method. To use the interrupt method, the SET TX INT bit is set. To use the DMA method, the SET TX DMA bit is set. Either an interrupt or a DMA request is generated the moment the TXRDY bit is set. When the SCI has completed transmission of all pending frames, the SCITXSHF register and SCITD are empty, the TXRDY bit is set, and an interrupt/DMA request is generated, if enabled. Because all data has been transmitted, the interrupt/DMA request should be halted. This can either be done by disabling the transmit interrupt (CLR TX INT) / DMA request (CLR TX DMA bit) or by disabling the transmitter (clear TXENA bit).

NOTE: The TXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVECT0 or SCIINTVECT1 register.

29.6 SCI Low-Power Mode

The SCI 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 SCI. During global low-power mode, all clocks to the SCI are turned off so the module is completely inactive.

Local low-power mode is asserted by setting the POWERDOWN bit; setting this bit stops the clocks to the SCI internal logic and the module registers. Setting the POWERDOWN bit causes the SCI to enter local low-power mode and clearing the POWERDOWN bit causes SCI 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 SCI for that particular access alone.

The wake-up interrupt is used to allow the SCI to exit low-power mode automatically when a low level is detected on the SCIRX pin and also this clears the POWERDOWN bit. If wake-up interrupt is disabled, then the SCI immediately enters low-power mode whenever it is requested and also any activity on the SCIRX pin does not cause the SCI to exit low-power mode.

NOTE: **Enabling Local Low-Power Mode During Receive and Transmit**

If the wake-up interrupt is enabled and low-power mode is requested while the receiver is receiving data, then the SCI immediately generates a wake-up interrupt to clear the power-down bit and prevents the SCI from entering low-power mode and thus completes the current reception. Otherwise, if the wake-up interrupt is disabled, then the SCI completes the current reception and then enters the low-power mode.

29.6.1 Sleep Mode for Multiprocessor Communication

When the SCI receives data and transfers that data from SCIRXSHF to SCIRD, the RXRDY bit is set and if RX INT ENA is set, the SCI also generates an interrupt. The interrupt triggers the CPU to read the newly received frame before another one is received. In multiprocessor communication modes, this default behavior may be enhanced to provide selective indication of new data. When SCI receives an address frame that does not match its address, the device can ignore the data following this non-matching address until the next address frame by using sleep mode. Sleep mode can be used with both idle-line and address-bit multiprocessor modes.

If sleep mode is enabled by the SLEEP bit, then the SCI transfers data from SCIRXSHF to SCIRD only for address frames. Therefore, in sleep mode, all data frames are assembled in the SCIRXSHF register without being shifted into the SCIRD and without initiating a receive interrupt or DMA request. Upon reception of an address frame, the contents of the SCIRXSHF are moved into SCIRD, and the software must read SCIRD and determine if the SCI is being addressed by comparing the received address against the address previously set in the software and stored somewhere in memory (the SCI does not have hardware available for address comparison). If the SCI is being addressed, the software must clear the SLEEP bit so that the SCI will load SCIRD with the data of the data frames that follow the address frame.

When the SCI has been addressed and sleep mode has been disabled (in software) to allow the receipt of data, the SCI should check the RXWAKE bit (SCIFLR.12) to determine when the next address has been received. This bit is set to 1 if the current value in SCIRD is an address and set to 0 if SCIRD contains data. If the RXWAKE bit is set, then software should check the address in SCIRD against its own address. If it is still being addressed, then sleep mode should remain disabled. Otherwise, the SLEEP bit should be set again.

Following is a sequence of events typical of sleep mode operation:

- The SCI is configured and both sleep mode and receive actions are enabled.
- An address frame is received and a receive interrupt is generated.
- Software compares the received address frame against that set by software and determines that the SCI is not being addressed, so the value of the SLEEP bit is not changed.
- Several data frames are shifted into SCIRXSHF, but no data is moved to SCIRD and no receive interrupts are generated.
- A new address frame is received and a receive interrupt is generated.
- Software compares the received address frame against that set by software and determines that the SCI is being addressed and clears the SLEEP bit.
- Data shifted into SCIRXSHF is transferred to SCIRD, and a receive interrupt is generated after each data frame is received.
- In each interrupt routine, software checks RXWAKE to determine if the current frame is an address frame.
- Another address frame is received, RXWAKE is set, software determines that the SCI is not being addressed and sets the SLEEP bit back to 1. No receive interrupts are generated for the data frames following this address frame.

By ignoring data frames that are not intended for the device, fewer interrupts are generated. These interrupts would otherwise require CPU intervention to read data that is of no significance to this specific device. Using sleep mode can help free some CPU resources.

Except for the RXRDY flag, the SCI continues to update the receiver status flags (see Table 29-11) while sleep mode is active. In this way, if an error occurs on the receive line, an application can immediately respond to the error and take the appropriate corrective action.

Because the RXRDY bit is not updated for data frames when sleep mode is enabled, the SCI can enable sleep mode and use a polling algorithm if desired. In this case, when RXRDY is set, software knows that a new address has been received. If the SCI is not being addressed, then the software should not change the value of the SLEEP bit and should continue to poll RXRDY.

29.7 SCI Control Registers

These registers are accessible in 8-, 16-, and 32-bit reads or writes. The SCI is controlled and accessed through the registers listed in Table 29-3. Among the features that can be programmed are the SCI communication and timing modes, baud rate value, frame format, DMA requests, and interrupt configuration. The base address for the control registers is FFF7 E500h.

Table 29-3. SCI Control Registers Summary

Offset	Acronym	Register Description	Section
00h	SCIGCR0	SCI Global Control Register 0	Section 29.7.1
04h	SCIGCR1	SCI Global Control Register 1	Section 29.7.2
0Ch	SCISETINT	SCI Set Interrupt Register	Section 29.7.3
10h	SCICLEARINT	SCI Clear Interrupt Register	Section 29.7.4
14h	SCISETINTLVL	SCI Set Interrupt Level Register	Section 29.7.5
18h	SCICLEARINTLVL	SCI Clear Interrupt Level Register	Section 29.7.6
1Ch	SCIFLR	SCI Flags Register	Section 29.7.7
20h	SCIINTVECT0	SCI Interrupt Vector Offset 0	Section 29.7.8
24h	SCIINTVECT1	SCI Interrupt Vector Offset 1	Section 29.7.9
28h	SCIFORMAT	SCI Format Control Register	Section 29.7.10
2Ch	BRS	Baud Rate Selection Register	Section 29.7.11
30h	SCIED	Receiver Emulation Data Buffer	Section 29.7.12.1
34h	SCIRD	Receiver Data Buffer	Section 29.7.12.2
38h	SCITD	Transmit Data Buffer	Section 29.7.12.3
3Ch	SCIPIO0	SCI Pin I/O Control Register 0	Section 29.7.13
40h	SCIPIO1	SCI Pin I/O Control Register 1	Section 29.7.14
44h	SCIPIO2	SCI Pin I/O Control Register 2	Section 29.7.15
48h	SCIPIO3	SCI Pin I/O Control Register 3	Section 29.7.16
4Ch	SCIPIO4	SCI Pin I/O Control Register 4	Section 29.7.17
50h	SCIPIO5	SCI Pin I/O Control Register 5	Section 29.7.18
54h	SCIPIO6	SCI Pin I/O Control Register 6	Section 29.7.19
58h	SCIPIO7	SCI Pin I/O Control Register 7	Section 29.7.20
5Ch	SCIPIO8	SCI Pin I/O Control Register 8	Section 29.7.21
90h	IODFTCTRL	Input/Output Error Enable Register	Section 29.7.22

29.7.1 SCI Global Control Register 0 (SCIGCR0)

The SCIGCR0 register defines the module reset. Figure 29-8 and Table 29-4 illustrate this register.

Figure 29-8. SCI Global Control Register 0 (SCIGCR0) [offset = 00]

31	16
Reserved	
R-0	

15	1	0
Reserved		RESET
R-0		R/WP-0

LEGEND: R = Read only; R/WP = Read/Write in privileged mode only; -*n* = value after reset

Table 29-4. SCI Global Control Register 0 (SCIGCR0) Fied Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	RESET		This bit resets the SCI module.
		0	SCI module is in reset.
		1	SCI module is out of reset.
			Note: Read/Write in privileged mode only.

29.7.2 SCI Global Control Register 1 (SCIGCR1)

The SCIGCR1 register defines the frame format, protocol, and communication mode used by the SCI. Figure 29-9 and Table 29-5 illustrate this register.

NOTE: The SCIGCR1 Control Register Bits should not be changed during Frame Transmission or Reception.

Figure 29-9. SCI Global Control Register 1 (SCIGCR1) [offset = 04h]

31	26	25	24
Reserved		TXENA	RXENA
R-0		R/W-0	R/W-0

23	18	17	16
Reserved		CONT	LOOP BACK
R-0		R/W-0	R/W-0

15	10	9	8
Reserved		POWERDOWN	SLEEP
R-0		R/WP-0	R/W-0

7	6	5	4	3	2	1	0
SWnRST	Reserved	CLOCK	STOP	PARITY	PARITY ENA	TIMING MODE	COMM MODE
R/W-0	R-0	R/W-0	R/WC-0	R/WC-0	R/W-0	R/WC-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; WP = Write in privileged mode only; WC = Write in sci-compatible mode only; -*n* = value after reset

Table 29-5. SCI Global Control Register 1 (SCIGCR1) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	Reads return 0. Writes have no effect.
25	TXENA		Transmit enable. Data is transferred from SCITD to the SCITXSHF shift out register only when the TXENA bit is set.
		0	Disable transfers from SCITD to SCITXSHF.
		1	Enable SCI to transfer data from SCITD to SCITXSHF.
			Note: Data written to SCITD or the transmit multi-buffer before TXENA is set is not transmitted. If TXENA is cleared while transmission is ongoing, the data previously written to SCITD is sent.
24	RXENA		Receive enable. RXENA allows or prevents the transfer of data from SCIRXSHF to SCIRD.
		0	The receiver will not transfer data from the shift buffer to the receive buffer.
		1	The receiver will transfer data from the shift buffer to the receive buffer.
			Note: Clearing RXENA stops received characters from being transferred into the receive buffer or multi-buffers, prevents the RX status flags (see Table 29-11) from being updated by receive data, and inhibits both receive and error interrupts. However, the shift register continues to assemble data regardless of the state of RXENA. **Note: If RXENA is cleared before a frame is completely received, the data from the frame is not transferred into the receive buffer.** **Note: If RXENA is set before a frame is completely received, the data from the frame is transferred into the receive buffer. If RXENA is set while SCIRXSHF is in the process of assembling a frame, the status flags are not assured to be accurate for that frame. To ensure that the status flags correctly reflect what was detected on the bus during a particular frame, RXENA should be set before the detection of that frame.**
23-18	Reserved	0	Reads return 0. Writes have no effect.
17	CONT		Continue on suspend. This bit has an effect only when a program is being debugged with an emulator, and it determines how the SCI operates when the program is suspended. The
		0	When debug mode is entered, the SCI state machine is frozen. Transmissions are halted and resume when debug mode is exited.
		1	When debug mode is entered, the SCI continues to operate until the current transmit and receive functions are complete.

Table 29-5. SCI Global Control Register 1 (SCIGCR1) Field Descriptions (continued)

Bit	Field	Value	Description
16	LOOP BACK		Loopback bit. The self-checking option for the SCI can be selected with this bit. If the SCITX and SCIRX pins are configured with SCI functionality, then the SCITX pin is internally connected to the SCIRX pin. Externally, during loop back operation, the SCITX pin outputs a high value and the SCIRX pin is in a high-impedance state. If this bit value is changed while the SCI is transmitting or receiving data, errors may result.
		0	Loop back mode is disabled.
		1	Loop back mode is enabled.
15-10	Reserved	0	Reads return 0. Writes have no effect.
9	POWERDOWN		Power down. When the POWERDOWN bit is set, the SCI attempts to enter local low-power mode. If the POWERDOWN bit is set while the receiver is actively receiving data and the wake-up interrupt is enabled, then the SCI immediately asserts an error interrupt to prevent low-power mode from being entered. Only Privilege mode writes allowed.
		0	Normal operation.
		1	Low-power mode is enabled.
8	SLEEP		SCI sleep. In a multiprocessor configuration, this bit controls the receive sleep function. Clearing this bit brings the SCI out of sleep mode.
		0	Sleep mode is disabled.
		1	Sleep mode is enabled.
			Note: The receiver still operates when the SLEEP bit is set; however, RXRDY is updated and SCIRD is loaded with new data only when an address frame is detected. The remaining receiver status flags (see Table 29-11) are updated and an error interrupt is requested if the corresponding interrupt enable bit is set, regardless of the value of the SLEEP bit. In this way, if an error is detected on the receive data line while the SCI is asleep, software can promptly deal with the error condition.
			Note: The SLEEP bit is not automatically cleared when an address byte is detected.
			See Section 29.6 for more information on using the SLEEP bit for multiprocessor communication.
7	SWnRST		Software reset (active low).
		0	The SCI is in its reset state; no data will be transmitted or received. Writing a 0 to this bit initializes the SCI state machines and operating flags as defined in Table 29-11 and Table 29-12. All affected logic is held in the reset state until a 1 is written to this bit.
		1	The SCI is in its ready state; transmission and reception can be done. After this bit is set to 1, the configuration of the module should not change.
			Note: The SCI should only be configured while SWnRESET = 0.
6	Reserved	0	Reads return 0. Writes have no effect.
5	CLOCK		SCI internal clock enable. The CLOCK bit determines the source of the module clock on the SCICLK pin.
		0	The external SCICLK is the clock source.
		1	The internal SCICLK is the clock source.
			Note: If an external clock is selected, then the internal baud rate generator and baud rate registers are bypassed. The maximum frequency allowed for an externally sourced SCI clock is VCLK/16.
4	STOP		SCI number of stop bits per frame.
		0	One stop bit is used.
		1	Two stop bits are used.
			Note: The receiver checks for only one stop bit. However in idle-line mode, the receiver waits until the end of the second stop bit (if STOP = 1) to begin checking for an idle period.

Table 29-5. SCI Global Control Register 1 (SCIGCR1) Field Descriptions (continued)

Bit	Field	Value	Description
3	PARITY		SCI parity odd/even selection. If the PARITY ENA bit is set, PARITY designates odd or even parity.
		0	Odd parity is used.
		1	Even parity is used.
			The parity bit is calculated based on the data bits in each frame and the address bit (in address-bit mode). The start and stop fields in the frame are not included in the parity calculation.
			For odd parity, the SCI transmits and expects to receive a value in the parity bit that makes odd the total number of bits in the frame with the value of 1.
			For even parity, the SCI transmits and expects to receive a value in the parity bit that makes even the total number of bits in the frame with the value of 1.
2	PARITY ENA		Parity enable. This bit enables or disables the parity function.
		0	Parity is disabled; no parity bit is generated during transmission or is expected during reception.
		1	Parity is enabled. A parity bit is generated during transmission and is expected during reception.
1	TIMING MODE		SCI timing mode bit.
		0	Synchronous timing is used.
		1	Asynchronous timing is used.
0	COMM MODE		SCI communication mode bit.
		0	Idle-line mode is used.
		1	Address-bit mode is used.

29.7.3 SCI Set Interrupt Register (SCISETINT)

Figure 29-10 and Table 29-6 illustrate this register. SCISETINT register is used to enable the required interrupts supported by the module.

Figure 29-10. SCI Set Interrupt Register (SCISETINT) [offset = 0Ch]

31	27	26	25	24
Reserved		SET FE INT	SET OE INT	SET PE INT
R-0		R/W-0	R/W-0	R/W-0

23	19	18	17	16
Reserved		SET RX DMA ALL	SET RX DMA	SET TX DMA
R-0		R/WC-0	R/W-0	R/W-0

15	10	9	8
Reserved		SET RX INT	SET TX INT
R-0		R/W-0	R/W-0

7	2	1	0
Reserved		SET WAKEUP INT	SET BRKDT INT
R-0		R/W-0	R/WC-0

LEGEND: R/W = Read/Write; R = Read only; WC = Write in sci-compatible mode only; -*n* = value after reset

Table 29-6. SCI Set Interrupt Register (SCISETINT) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reads return 0. Writes have no effect.
26	SET FE INT	0	Set framing-error interrupt. Setting this bit enables the SCI module to generate an interrupt when a framing error occurs. *Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.
25	SET OE INT	0	Set overrun-error interrupt. Setting this bit enables the SCI module to generate an interrupt when an overrun error occurs. *Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.
24	SET PE INT	0	Set parity interrupt. Setting this bit enables the SCI module to generate an interrupt when a parity error occurs. *Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.
23-19	Reserved	0	Reads return 0. Writes have no effect.
18	SET RX DMA ALL	0	Set receive DMA all. This bit determines if a separate interrupt is generated for the address frames sent in multiprocessor communications. When this bit is 0, RX interrupt requests are generated for address frames and DMA requests are generated for data frames. When this bit is 1, RX DMA requests are generated for both address and data frames. *Read:* The DMA request is disabled for address frames (the receive interrupt request is enabled for address frames). *Write:* Writing a 0 to this bit has no effect.
		1	*Read and write:* The DMA request is enabled for address and data frames
17	SET RX DMA	0	Set receiver DMA. To enable receiver DMA requests, this bit must be set. If it is cleared, interrupt requests are generated depending on the SET RX INT bit (SCISETINT[9]). *Read:* The DMA request is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read and write:* The DMA request is enabled for address and data frames

Table 29-6. SCI Set Interrupt Register (SCISETINT) Field Descriptions (continued)

Bit	Field	Value	Description
16	SET TX DMA		Set transmit DMA. To enable DMA requests for the transmitter, this bit must be set. If it is cleared, interrupt requests are generated depending on the SET TX INT bit (SCISETINT[8]).
		0	*Read:* Transmit DMA request is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read and write:* Transmit DMA request is enabled.
15-10	Reserved	0	Reads return 0. Writes have no effect.
9	SET RX INT		Receiver interrupt enable. Setting this bit enables the SCI to generate a receive interrupt after a frame has been completely received and the data is being transferred from SCIRXSHF to SCIRD.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.
8	SET TX INT		Set transmitter interrupt. Setting this bit enables the SCI to generate a transmit interrupt as data is being transferred from SCITD to SCITXSHF and the TXRDY bit is being set.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.
7-2	Reserved	0	Reads return 0. Writes have no effect.
1	SET WAKEUP INT		Set wakeup interrupt. Setting this bit enables the SCI to generate a wakeup interrupt and thereby exit low-power mode. If enabled, the wakeup interrupt is asserted when local lowpower mode is requested while the receiver is busy or if a low level is detected on the SCIRX pin during low-power mode.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.
0	SET BRKDT INT		Set break detect interrupt. Setting this bit enables the SCI to generate an error interrupt if a break condition is detected on the SCIRX pin.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt is enabled.

29.7.4 SCI Clear Interrupt Register (SCICLEARINT)

Figure 29-11 and Table 29-7 illustrate this register. SCICLEARINT register is used to clear the selected enabled interrupts with out accessing SCISETINT register.

Figure 29-11. SCI Clear Interrupt Register (SCICLEARINT) [offset = 10h]

31 - 27	26	25	24
Reserved	CLR FE INT	CLR OE INT	CLR PE INT
R-0	R/W-0	R/W-0	R/W-0

23 - 19	18	17	16
Reserved	CLR RX DMA ALL	CLR RX DMA	CLR TX DMA
R-0	R/WC-0	R/W-0	R/W-0

15 - 10	9	8
Reserved	CLR RX INT	CLR TX INT
R-0	R/W-0	R/W-0

7 - 2	1	0
Reserved	CLR WAKEUP INT	CLR BRKDT INT
R-0	R/W-0	R/WC-0

LEGEND: R/W = Read/Write; R = Read only; WC = Write in sci-compatible mode only; -*n* = value after reset

Table 29-7. SCI Clear Interrupt Register (SCICLEARINT) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reads return 0. Writes have no effect.
26	CLR FE INT		Clear framing-error interrupt. This bit disables the framing-error interrupt when set.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
25	CLR CE INT		Clear overrun-error interrupt. This bit disables the SCI overrun error interrupt when set.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
24	CLR PE INT		Clear parity interrupt. This bit disables the parity error interrupt when set.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
23-19	Reserved	0	Reads return 0. Writes have no effect.
18	CLR RX DMA ALL		Clear receive DMA all. This bit clears the receive DMA request for address frames when set. Only receive data frames generate a DMA request.
		0	*Read:* Receive DMA request for address frames is disabled; Instead, RX interrupt requests are enabled for address frames. Receive DMA requests are still enabled for data frames. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The receive DMA request for address and data frames is enabled. *Write:* The receive DMA request for address and data frames is disabled.

Table 29-7. SCI Clear Interrupt Register (SCICLEARINT) Field Descriptions (continued)

Bit	Field	Value	Description
17	CLR RX DMA		Clear receive DMA request. This bit disables the receive DMA request when set.
		0	*Read:* The DMA request is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The receive DMA request is enabled. *Write:* The receive DMA request for is disabled.
16	CLR TX DMA		Clear transmit DMA request. This bit disables the transmit DMA request when set.
		0	*Read:* Transmit DMA request is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The transmit DMA request is enabled. *Write:* The transmit DMA request for is disabled.
15-10	Reserved	0	Reads return 0. Writes have no effect.
9	CLR RX INT		Clear receiver interrupt. This bit disables the receiver interrupt when set.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
8	CLR TX INT		Clear transmitter interrupt. This bit disables the transmitter interrupt when set.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
7-2	Reserved	0	Reads return 0. Writes have no effect.
1	CLR WAKEUP INT		Clear wakeup interrupt. This bit disables the wakeup interrupt when set.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.
0	CLR BRKDT INT		Clear break detect interrupt. This bit disables the break-detect interrupt when set.
		0	*Read:* The interrupt is disabled. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt is enabled. *Write:* The interrupt is disabled.

29.7.5 SCI Set Interrupt Level Register (SCISETINTLVL)

Figure 29-12 and Table 29-8 illustrate this register. This register is used to set the interrupt level for the supported interrupts.

Figure 29-12. SCI Set Interrupt Level Register (SCISETINTLVL) [offset = 14h]

31 27	26	25	24
Reserved	SET FE INT LVL	SET OE INT LVL	SET PE INT LVL
R-0	R/W-0	R/W-0	R/W-0

23 19	18	17 16
Reserved	SET RX DMA ALL INT LVL	Reserved
R-0	R/WC-0	R-0

15 10	9	8
Reserved	SET RX INT LVL	SET TX INT LVL
R-0	R/W-0	R/W-0

7 2	1	0
Reserved	SET WAKEUP INT LVL	SET BRKDT INT LVL
R-0	R/W-0	R/WC-0

LEGEND: R/W = Read/Write; R = Read only; WC = Write in sci-compatible mode only; -*n* = value after reset

Table 29-8. SCI Set Interrupt Level Register (SCISETINTLVL) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reads return 0. Writes have no effect.
26	SET FE INT LVL	0	Set framing-error interrupt level. *Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt level is mapped to the INT1 line.
25	SET CE INT LVL	0	Set overrun-error interrupt level. *Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt level is mapped to the INT1 line.
24	SET PE INT LVL	0	Set parity error interrupt level. *Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt level is mapped to the INT1 line.
23-19	Reserved	0	Reads return 0. Writes have no effect.
18	SET RX DMA ALL LVL	0	Set receive DMA all interrupt levels. *Read:*The receive interrupt request for address frames is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The receive interrupt request for address frames is mapped to the INT1 line.
17-10	Reserved	0	Reads return 0. Writes have no effect.
9	SET RX INT LVL	0	Set receiver interrupt level. *Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt level is mapped to the INT1 line.

Table 29-8. SCI Set Interrupt Level Register (SCISETINTLVL) Field Descriptions (continued)

Bit	Field	Value	Description
8	SET TX INT LVL	0	Set transmitter interrupt level. *Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt level is mapped to the INT1 line.
7-2	Reserved	0	Reads return 0. Writes have no effect.
1	SET WAKEUP INT LVL	0	Set wakeup interrupt level. *Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt level is mapped to the INT1 line.
0	SET BRKDT INT LVL	0	Set break detect interrupt level. *Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read or write:* The interrupt level is mapped to the INT1 line.

29.7.6 SCI Clear Interrupt Level Register (SCICLEARINTLVL)

Figure 29-13 and Table 29-9 illustrate this register.

Figure 29-13. SCI Clear Interrupt Level Register (SCICLEARINTLVL) [offset = 18h]

31 27	26	25	24
Reserved	CLR FE INT LVL	CLR OE INT LVL	CLR PE INT LVL
R-0	R/W-0	R/W-0	R/W-0

23 19	18	17 16
Reserved	CLR RX DMA ALL INT LVL	Reserved
R-0	R/WC-0	R-0

15 10	9	8
Reserved	CLR RX INT LVL	CLR TX INT LVL
R-0	R/W-0	R/W-0

7 2	1	0
Reserved	CLR WAKEUP INT LVL	CLR BRKDT INT LVL
R-0	R/W-0	R/WC-0

LEGEND: R/W = Read/Write; R = Read only; WC = Write in sci-compatible mode only; -*n* = value after reset

Table 29-9. SCI Clear Interrupt Level Register (SCICLEARINTLVL) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reads return 0. Writes have no effect.
26	CLR FE INT LVL	0	Clear framing-error interrupt. *Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.

Table 29-9. SCI Clear Interrupt Level Register (SCICLEARINTLVL) Field Descriptions (continued)

Bit	Field	Value	Description
25	CLR CE INT LVL		Clear overrun-error interrupt.
		0	*Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
24	CLR PE INT LVL		Clear parity interrupt.
		0	*Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
23-19	Reserved	0	Reads return 0. Writes have no effect.
18	CLR RX DMA ALL LVL		Clear receive DMA interrupt level.
		0	*Read:* The receive interrupt request for address frames is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The receive interrupt request for address frames is mapped to the INT1 line. *Write:* The receive interrupt request for address frames is mapped to the INT0 line.
17-10	Reserved	0	Reads return 0. Writes have no effect.
9	CLR RX INT LVL		Clear receiver interrupt.
		0	*Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
8	CLR TX INT LVL		Clear transmitter interrupt.
		0	*Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
7-2	Reserved	0	Reads return 0. Writes have no effect.
1	CLR WAKEUP INT LVL		Clear wakeup interrupt.
		0	*Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.
0	CLR BRKDT INT LVL		Clear break detect interrupt.
		0	*Read:* The interrupt level is mapped to the INT0 line. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The interrupt level is mapped to the INT1 line. *Write:* The interrupt level is mapped to the INT0 line.

29.7.7 SCI Flags Register (SCIFLR)

Figure 29-14 and Table 29-10 illustrate this register.

Figure 29-14. SCI Flags Register (SCIFLR) [offset = 1Ch]

31	27	26	25	24
Reserved		FE	OE	PE
R-0		R/W-0	R/W-0	R/W-0

23	16
Reserved	
R-0	

15	13	12	11	10	9	8
Reserved		RX WAKE	TX EMPTY	TX WAKE	RX RDY	TX RDY
R-0		R/WC-0	R/W-1	R/WC-0	R/W-0	R/W-1

7	4	3	2	1	0
Reserved		BUSY	IDLE	WAKE UP	BRKDT
R-0		R/W-0	R-0	R/WL-0	R/WC-0

LEGEND: R/W = Read/Write; R = Read only; WC = Write in sci-compatible mode only; -*n* = value after reset

Table 29-10. SCI Flags Register (SCIFLR) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved		Reads return 0. Writes have no effect.
26	FE		Framing error flag. This bit is set when an expected stop bit is not found. In SCI compatibility mode, only the first stop bit is checked. The missing stop bit indicates that synchronization with the start bit has been lost and that the character is incorrectly framed. Detection of a framing error causes the SCI to generate an error interrupt, if the SET FE INT bit (SCISETINT[26]) is set. The framing error flag is cleared by the following: • Setting of the SW nRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reading the corresponding interrupt offset in SCIINTVECT0/1 • Reception of a new character In multi-buffer mode, the frame is defined in the SCIFORMAT register.
		0	*Read:* No framing error has been detected since the last clear. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* A framing error has been detected since the last clear. *Write:* The bit is cleared to 0.
25	OE		Overrun error flag. This bit is set when the transfer of data from SCIRXSHF to SCIRD overwrites unread data already in SCIRD. Detection of an overrun error causes the SCI to generate an error interrupt if the SET OE INT bit (SCISETINT[25]) is set. The OE flag is reset by the following: • Setting of the SW nRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reading the corresponding interrupt offset in SCIINTVECT0/1
		0	*Read:* No overrun error has been detected since the last clear. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* An overrun error has been detected since the last clear. *Write:* The bit is cleared to 0.

Table 29-10. SCI Flags Register (SCIFLR) Field Descriptions (continued)

Bit	Field	Value	Description
24	PE		Parity error flag. This bit is set when a parity error is detected in the received data. In SCI address-bit mode, the parity is calculated on the data and address bit fields of the received frame. In idle-line mode, only the data is used to calculate parity. An error is generated when a character is received with a mismatch between the number of 1s and its parity bit. If the parity function is disabled (SCIGCR[2] = 0), the PE flag is disabled and read as 0. Detection of a parity error causes the SCI to generate an error interrupt if the SET PE INT bit (SCISETINT[24]) is set. The PE bit is reset by the following: • Setting of the SW nRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reception of a new character • Reading the corresponding interrupt offset in SCIINTVECT0/1
		0	*Read:* No parity error has been detected since the last clear. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* A parity error has been detected since the last clear. *Write:* The bit is cleared to 0.
23-13	Reserved	0	Reads return 0. Writes have no effect.
12	RXWAKE		Receiver wakeup detect flag. The SCI sets this bit to indicate that the data currently in SCIRD is an address. RXWAKE is cleared by the following: • Setting of the SW nRST bit • Setting of the RESET bit • A system reset • Upon receipt of a data frame.
		0	The data in SCIRD is not an address.
		1	The data in SCIRD is an address.
11	TX EMPTY		Transmitter empty flag. This flag indicates the transmitter's buffer register(s) (SCITD/TDy) and shift register (SCITXSHF) are empty. **Note: The RESET bit, an active SW nRESET (SCIGCR1[7]) or a system reset sets this bit. This bit does not cause an interrupt request.**
		0	Transmitter buffer or shift register (or both) are loaded with data.
		1	Transmitter buffer and shift registers are both empty.
10	TXWAKE		Transmitter wakeup method select. The TXWAKE bit controls whether the data in SCITD should be sent as an address or data frame using multiprocessor communication format. This bit is set to 1 or 0 by software before a byte is written to SCITD and is cleared by the SCI when data is transferred from SCITD to SCITXSHF or by a system reset. **Note: TXWAKE is not cleared by the SW nRESET bit.** *Address-bit mode*
		0	Frame to be transmitted will be data (address bit = 0).
		1	Frame to be transmitted will be an address (address bit = 1).
			Idle-line mode
		0	The frame to be transmitted will be data.
		1	The following frame to be transmitted will be an address (writing a 1 to this bit followed by writing dummy data to the SCITD will result in a idle period of 11 bit periods before the next frame is transmitted).

Table 29-10. SCI Flags Register (SCIFLR) Field Descriptions (continued)

Bit	Field	Value	Description
9	RXRDY		Receiver ready flag. The receiver sets this bit to indicate that the SCIRD contains new data and is ready to be read by the CPU or DMA. The SCI generates a receive interrupt when RXRDY flag bit is set if the SET RX INT bit (SCISETINT[9]) is set; RXRDY is cleared by the following: • Setting of the SW nRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reading the SCIRD register in compatibility mode • Reading the last data byte RDy of the response in SCI mode **Note: The RXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register.**
		0	*Read:* No new data is in SCIRD. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* New data is ready to be read from SCIRD. *Write:* The bit is cleared to 0.
8	TXRDY		Transmitter buffer register ready flag. When set, this bit indicates that the transmit buffer is ready to get another character from a CPU or DMA write. Writing data to SCITD automatically clears this bit. This bit is set after the data of the TX buffer is shifted into the SCITXSHF register. This event can trigger a transmit interrupt after data is copied to the TX shift register SCITXSHF, if the SET TX INT bit (SCISETINT[8]) is set. **Note: 1) TXRDY is also set to 1 by setting of the RESET bit, setting of the RESET bit, or by a system reset.** **2) The TXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register.** **3) The transmit interrupt request can be eliminated until the next series of data written into the transmit buffers SCITD0 and SCITD1, by disabling the corresponding interrupt via the SCICLEARINT register or by disabling the transmitter via the TXENA bit (SCIGCR1[25]).**
		0	SCITD is full.
		1	SCITD is ready to receive the next character.
7-4	Reserved	0	Reads return 0. Writes have no effect.
3	BUSY		Bus busy flag. TThis bit indicates whether the receiver is in the process of receiving a frame. As soon as the receiver detects the beginning of a start bit, the BUSY bit is set to 1. When the reception of a frame is complete, the SCI clears the BUSY bit. If the SET WAKEUP INT bit (SCISETINT[2]) is set and power down is requested while this bit is set, the SCI automatically prevents low-power mode from being entered and generates wakeup interrupt. The BUSY bit is controlled directly by the SCI receiver, but this bit can also be cleared by the following: • Setting the SW nRST bit • Setting of the RESET bit • A system reset occurring
		0	The receiver is not currently receiving a frame.
		1	The receiver is currently receiving a frame.
2	IDLE		SCI receiver in idle state. While this bit is set, the SCI looks for an idle period to resynchronize itself with the bit stream. The receiver does not receive any data while the bit is set. The bus must be idle for 11 bit periods to clear this bit. The SCI enters the idle state if one of the following events occurs: • A system reset • An SCI software reset • A power down • The RX pin is configured as a general I/O pin
		0	The idle period has been detected; the SCI is ready to receive.
		1	The idle period has not been detected; the SCI will not receive any data.

Table 29-10. SCI Flags Register (SCIFLR) Field Descriptions (continued)

Bit	Field	Value	Description
1	WAKEUP		Wakeup flag. This bit is set by the SCI when receiver or transmitter activity has taken the module out of power-down mode. An interrupt is generated if the SET WAKEUP INT bit (SCISETINT[2]) is set. It is cleared by the following: • Setting of the SW nRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reading the corresponding interrupt offset in SCIINTVECT0/1 For compatibility mode, see the SCI document for more information on low-power mode.
		0	*Read:* The module will not wake up from power-down mode. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* Wake up from power-down mode. *Write:* The bit is cleared to 0.
0	BRKDT		SCI break-detect flag. This bit is set when the SCI detects a break condition on the SCIRX pin. A break condition occurs when the SCIRX pin remains continuously low for at least 10 bits after a missing first stop bit, that is, after a framing error. Detection of a break condition causes the SCI to generate an error interrupt if the SET BRKDT INT bit (SCISETINT[0]) is set. The BRKDT bit is reset by the following: • Setting of the SW nRST bit • Setting of the RESET bit • A system reset • Writing a 1 to this bit • Reading the corresponding interrupt offset in SCIINTVECT0/1
		0	*Read:* No break condition has been detected since the last clear. *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* A break condition has been detected. *Write:* The bit is cleared to 0.

Table 29-11. SCI Receiver Status Flags

SCI Flag	Register	Bit	Value After SW nRESET [1]
FE	SCIFLR	26	0
OE	SCIFLR	25	0
PE	SCIFLR	24	0
RXWAKE	SCIFLR	12	0
RXRDY	SCIFLR	9	0
BRKDT	SCIFLR	0	0

[1] The flags are frozen with their reset value while SW nRESET = 0.

Table 29-12. SCI Transmitter Status Flags

SCI Flag	Register	Bit	Value After SW nRESET [1]
TX EMPTY	SCIFLR	11	1
TXRDY	SCIFLR	8	1

[1] The flags are frozen with their reset value while SW nRESET = 0.

29.7.8 SCI Interrupt Vector Offset 0 (SCIINTVECT0)

Figure 29-15 and Table 29-13 illustrate this register.

Figure 29-15. SCI Interrupt Vector Offset 0 (SCIINTVECT0) [offset = 20h]

31–16
Reserved
R-0

15–4	3–0
Reserved	INTVECT0
R-0	R-0

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

Table 29-13. SCI Interrupt Vector Offset 0 (SCIINTVECT0) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	INVECT0	0-Fh	Interrupt vector offset for INT0. This register indicates the offset for interrupt line INT0. A read to this register updates its value to the next highest priority pending interrupt in SCIFLR and clears the flag in SCIFLR corresponding to the offset that was read. See Table 29-1 for a list of the interrupts. **Note: The flags for the receive (SCIFLR[9]) and the transmit (SCIFLR[8]) interrupt cannot be cleared by reading the corresponding offset vector in this register (see detailed description in SCIFLR register).**

29.7.9 SCI Interrupt Vector Offset 1 (SCIINTVECT1)

Figure 29-16 and Table 29-14 illustrate this register.

Figure 29-16. SCI Interrupt Vector Offset 1 (SCIINTVECT1) [offset = 24h]

31–16
Reserved
R-0

15–4	3–0
Reserved	INTVECT1
R-0	R-0

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

Table 29-14. SCI Interrupt Vector Offset 1 (SCIINTVECT1) Field Descriptions

Bit	Field	Value	Description
31-4	Reserved	0	Reads return 0. Writes have no effect.
3-0	INVECT1	0-Fh	Interrupt vector offset for INT1. This register indicates the offset for interrupt line INT1. A read to this register updates its value to the next highest priority pending interrupt in SCIFLR and clears the flag in SCIFLR corresponding to the offset that was read. See Table 29-1 for list of interrupts. **Note: The flags for the receive (SCIFLR[9]) and the transmit (SCIFLR[8]) interrupt cannot be cleared by reading the corresponding offset vector in this register (see detailed description in SCIFLR register).**

29.7.10 SCI Format Control Register (SCIFORMAT)

Figure 29-17 and Table 29-15 illustrate this register.

Figure 29-17. SCI Format Control Register (SCIFORMAT) [offset = 28h]

31 16
Reserved
R-0

15 3	2 0
Reserved	CHAR
R-0	R/WC-0

LEGEND: R/W = Read/Write; R = Read only; WC = Write in SCI-compatible mode only; -*n* = value after reset

Table 29-15. SCI Format Control Register (SCIFORMAT) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2-0	CHAR		Character length control bits. These bits set the SCI character length from 1 to 8 bits. **When data of fewer than eight bits in length is received, it is left-justified in SCIRD and padded with trailing zeros.** **Data read from the SCIRD should be shifted by software to make the received data right-justified.** **Data written to the SCITD should be right-justified but does not need to be padded with leading zeros.**
		0	The character is 1 bit long.
		1h	The character is 2 bits long.
		2h	The character is 3 bits long.
		3h	The character is 4 bits long.
		4h	The character is 5 bits long.
		5h	The character is 6 bits long.
		6h	The character is 7 bits long.
		7h	The character is 8 bits long.

29.7.11 Baud Rate Selection Register (BRS)

This section describes the baud rate selection register. Figure 29-18 and Table 29-16 illustrate this register.

Figure 29-18. Baud Rate Selection Register (BRS) [offset = 2Ch]

31 24	23 16
Reserved	BAUD
R-0	R/W-0

15 0
BAUD
R/W-0

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

Table 29-16. Baud Rate Selection Register (BRS) Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	0	Reads return 0. Writes have no effect.
23-0	BAUD	0-FF FFFFh	SCI 24-bit baud selection. The SCI has an internally-generated serial clock determined by the VCLK and the prescalers BAUD in this register. The SCI uses the 24-bit integer prescaler BAUD value of this register to select one of over 16,700,000. The baud rate can be calculated using the following formulas: $$\textit{Asynchronous baud value} = \left(\frac{\text{VCLK Frequency}}{16(\text{Baud} + 1)}\right) \quad (59)$$ $$\textit{Isosynchronous baud value} = \left(\frac{\text{VCLK Frequency}}{\text{Baud} + 1}\right) \quad (60)$$ For BAUD = 0, $$\textit{Asynchronous baud value} = \left(\frac{\text{VCLK Frequency}}{32}\right) \quad (61)$$ $$\textit{Isosynchronous baud value} = \left(\frac{\text{VCLK Frequency}}{2}\right) \quad (62)$$ Table 29-17 contains comparative baud values for different P values, with VCLK = 50 MHz, for asynchronous mode.

Table 29-17. Comparative Baud Values (Asynchronous Mode) [(1)(2)]

24-Bit Register Value		Baud Selected		Percent Error
Decimal	Hex	Ideal	Actual	
26	00001A	115200	115740	0.47
53	000035	57600	57870	0.47
80	000050	38400	38580	0.47
162	0000A2	19200	19172	-0.15
299	00012B	10400	10417	0.16
325	000145	9600	9586	-0.15
399	00018F	7812.5	7812.5	0.00
650	00028A	4800	4800	0.00
15624	003BA0	200	200	0.00
624999	098967	5	5	0.00

[(1)] VCLK = 50 MHz

[(2)] Values are in decimal except for column 2.

29.7.12 SCI Data Buffers (SCIED, SCIRD, SCITD)

The SCI has three addressable registers in which transmit and receive data is stored.

29.7.12.1 Receiver Emulation Data Buffer (SCIED)

The SCIED register is addressed at a location different from SCIRD, but is physically the same register. Figure 29-19 and Table 29-18 illustrate this register.

Figure 29-19. Receiver Emulation Data Buffer (SCIED) [offset = 30h]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		ED	
R-0		R-0	

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

Table 29-18. Receiver Emulation Data Buffer (SCIED) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	ED	0-FFh	Emulator data. Reading SCIED[7:0] does not clear the RXRDY flag, unlike reading SCIRD. This register should be used only by an emulator that must continually read the data buffer without affecting the RXRDY flag.

29.7.12.2 Receiver Data Buffer (SCIRD)

This register provides a location for the receiver data. Figure 29-20 and Table 29-19 illustrate this register.

NOTE: When the SCI receives data that is fewer than eight bits in length, it loads the data into this register in a left-justified format padded with trailing zeros. Therefore, the user software should perform a logical shift on the data by the correct number of positions to make it right justified.

Figure 29-20. Receiver Data Buffer (SCIRD) [offset = 34h]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		RD	
R-0		R-0	

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

Table 29-19. Receiver Data Buffer (SCIRD) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	RD	0-FFh	Receiver data. When a frame has been completely received, the data in the frame is transferred from the receiver shift register SCIRXSHF to this register. As this transfer occurs, the RXRDY flag is set and a receive interrupt is generated if SET RX INT is set. **Note: When the data is read from SCIRD, the RXRDY flag is automatically cleared.**

29.7.12.3 Transmit Data Buffer Register (SCITD)

Data to be transmitted is written to the SCITD register. Figure 29-21 and Table 29-20 illustrate this register.

NOTE: Data written to the SCITD register that is fewer than eight bits long must be right justified, but it does not need to be padded with leading zeros.

Figure 29-21. Transmit Data Buffer Register (SCITD) [offset = 38h]

31 — 16
Reserved
R-0

15 — 8	7 — 0
Reserved	TD
R-0	R/W-0

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

Table 29-20. Transmit Data Buffer Register (SCITD) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	TD	0-FFh	Transmit data. Data to be transmitted is written to the SCITD register. The transfer of data from this register to the transmit shift register SCITXSHF sets the TXRDY flag, which indicates that SCITD is ready to be loaded with another byte of data. **Note: If TX INT ENA is set, this data transfer also causes an interrupt.**

29.7.13 SCI Pin I/O Control Register 0 (SCIPIO0)

Figure 29-22 and Table 29-21 illustrate this register.

Figure 29-22. SCI Pin I/O Control Register 0 (SCIPIO0) [offset = 3Ch]

31 — 8
Reserved
R-0

7 — 3	2	1	0
Reserved	TX FUNC	RX FUNC	Reserved
R-0	R/W-0	R/W-0	R/W-0

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

Table 29-21. SCI Pin I/O Control Register 0 (SCIPIO0) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TX FUNC		Transfer function. This bit defines the function of pin SCITX.
		0	SCITX is a general-purpose digital I/O pin.
		1	SCITX is the SCI transmit pin.
1	RX FUNC		Receive function. This bit defines the function of pin SCIRX.
		0	SCIRX is a general-purpose digital I/O pin.
		1	SCIRX is the SCI receive pin.
0	Reserved	0	Reads return 0. Writes have no effect.

29.7.14 SCI Pin I/O Control Register 1 (SCIPIO1)

Figure 29-23 and Table 29-22 illustrate this register.

Figure 29-23. SCI Pin I/O Control Register 1 (SCIPIO1) [offset = 40h]

31–8
Reserved
R-0

7–3	2	1	0
Reserved	TX DIR	RX DIR	Reserved
R-0	R/W-0	R/W-0	R/W-0

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

Table 29-22. SCI Pin I/O Control Register 1 (SCIPIO1) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TX DIR		Transmit pin direction. This bit determines the data direction on the SCITX pin if it is configured with general-purpose I/O functionality (TX FUNC = 0). See Table 29-23 for the SCITX pin control with this bit and others.
		0	SCITX is a general-purpose input pin.
		1	SCITX is a general-purpose output pin.
1	RX DIR		Receive pin direction. This bit determines the data direction on the SCIRX pin if it is configured with general-purpose I/O functionality (RX FUNC = 0). See Table 29-24 for the SCIRX pin control with this bit and others.
		0	SCIRX is a general-purpose input pin.
		1	SCIRX is a general-purpose output pin.
0	Reserved	0	Reads return 0. Writes have no effect.

Table 29-23. SCITX Pin Control

Function	TX IN [1]	TX OUT	TX FUNC	TX DIR
SCITX	X	X	1	X
General-purpose input	X	X	0	0
General-purpose output, high	X	1	0	1
General-purpose output, low	X	0	0	1

[1] TX IN is a read-only bit. Its value always reflects the level of the SCITX pin.

Table 29-24. SCIRX Pin Control

Function	RX IN [1]	RX OUT	RX FUNC	RX DIR
SCIRX	X	X	1	X
General-purpose input	X	X	0	0
General-purpose output, high	X	1	0	1
General-purpose output, low	X	0	0	1

[1] RX IN is a read-only bit. Its value always reflects the level of the SCIRX pin.

29.7.15 SCI Pin I/O Control Register 2 (SCIPIO2)

Figure 29-24 and Table 29-25 illustrate this register.

Figure 29-24. SCI Pin I/O Control Register 2 (SCIPIO2) [offset = 44h]

31–8
Reserved
R-0

7–3	2	1	0
Reserved	TX IN	RX IN	Reserved
R-0	R-x	R-x	R-x

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

Table 29-25. SCI Pin I/O Control Register 2 (SCIPIO2) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TX IN		Transmit pin in. This bit contains the current value on the SCITX pin.
		0	The SCITX pin is at logic low (0).
		1	The SCITX pin is at logic high (1).
1	RX IN		Receive pin in. This bit contains the current value on the SCIRX pin.
		0	The SCIRX pin is at logic low (0).
		1	The SCIRX pin is at logic high (1).
0	Reserved	0	Writes have no effect.

29.7.16 SCI Pin I/O Control Register 3 (SCIPIO3)

Figure 29-25 and Table 29-26 illustrate this register.

Figure 29-25. SCI Pin I/O Control Register 3 (SCIPIO3) [offset = 48h]

31 … 8
Reserved
R-0

7 … 3	2	1	0
Reserved	TX OUT	RX OUT	Reserved
R-0	R/W-0	R/W-0	R/W-0

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

Table 29-26. SCI Pin I/O Control Register 3 (SCIPIO3) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TX OUT		Transmit pin out. This pin specifies the logic to be output on pin SCITX if the following conditions are met: • TX FUNC = 0 (SCITX pin is a general-purpose I/O.) • TX DIR = 1 (SCITX pin is a general-purpose output.) See Table 29-23 for an explanation of this bit's effect in combination with other bits.
		0	The output on the SCITX is at logic low (0).
		1	The output on the SCITX pin is at logic high (1). (Output voltage is V_{OH} or higher if TXPDR = 0 and output is in high impedance state if TXPDR = 1)
1	RX OUT		Receive pin out. This bit specifies the logic to be output on pin SCIRX if the following conditions are met: • RX FUNC = 0 (SCIRX pin is a general-purpose I/O.) • RX DIR = 1 (SCIRX pin is a general-purpose output.) See Table 29-24 for an explanation of this bit's effect in combination with the other bits.
		0	The output on the SCIRX pin is at logic low (0).
		1	The output on the SCIRX pin is at logic high (1). (Output voltage is V_{OH} or higher if RXPDR = 0, and output is in high impedance state if RXPDR = 1)
0	Reserved	0	Reads return 0. Writes have no effect.

29.7.17 SCI Pin I/O Control Register 4 (SCIPIO4)

Figure 29-26 and Table 29-27 illustrate this register.

Figure 29-26. SCI Pin I/O Control Register 4 (SCIPIO4) [offset = 4Ch]

31 … 8
Reserved
R-0

7 … 3	2	1	0
Reserved	TX SET	RX SET	Reserved
R-0	R/W-0	R/W-0	R/W-0

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

Table 29-27. SCI Pin I/O Control Register 4 (SCIPIO4) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TX SET		Transmit pin set. This bit sets the logic to be output on pin SCITX if the following conditions are met: • TX FUNC = 0 (SCITX pin is a general-purpose I/O.) • TX DIR = 1 (SCITX pin is a general-purpose output.) See Table 29-23 for an explanation of this bit's effect in combination with other bits.
		0	*Read:* The output on SCITX is at logic low (0). *Write:* Writing a 0 to this bit has no effect.
		1	*Read and write:* The output on SCITX is at logic high (1).
1	RX SET		Receive pin set. This bit sets the data to be output on pin SCIRX if the following conditions are met: • RX FUNC = 0 (SCIRX pin is a general-purpose I/O.) • RX DIR = 1 (SCIRX pin is a general-purpose output.) See Table 29-24 for an explanation of this bit's effect in combination with the other bits.
		0	*Read:* The output on SCIRX is at logic low (0). *Write:* Writing a 0 to this bit has no effect.
		1	*Read and write:* The output on SCIRX is at logic high (1).
0	Reserved	0	Reads return 0. Writes have no effect.

29.7.18 SCI Pin I/O Control Register 5 (SCIPIO5)

Figure 29-27 and Table 29-28 illustrate this register.

Figure 29-27. SCI Pin I/O Control Register 5 (SCIPIO5) [offset = 50h]

31–8
Reserved
R-0

7–3	2	1	0
Reserved	TX CLR	RX CLR	Reserved
R-0	R/W-0	R/W-0	R/W-0

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

Table 29-28. SCI Pin I/O Control Register 5 (SCIPIO5) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TX CLR		Transmit pin clear. This bit clears the logic to be output on pin SCITX if the following conditions are met: • TX FUNC = 0 (SCITX pin is a general-purpose I/O.) • TX DIR = 1 (SCITX pin is a general-purpose output.)
		0	*Read:* The output on SCITX is at logic low (0). *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The output on SCITX is at logic high (1). *Write:* The output on SCITX is at logic low (0).
1	RX CLR		Receive pin clear. This bit clears the logic to be output on pin SCIRX if the following conditions are met: • RX FUNC = 0 (SCIRX pin is a general-purpose I/O.) • RX DIR = 1 (SCIRX pin is a general-purpose output.)
		0	*Read:* The output on SCIRX is at logic low (0). *Write:* Writing a 0 to this bit has no effect.
		1	*Read:* The output on SCIRX is at logic high (1). *Write:* The output on SCIRX is at logic low (0).
0	Reserved	0	Reads return 0. Writes have no effect.

29.7.19 SCI Pin I/O Control Register 6 (SCIPIO6)

Figure 29-28 and Table 29-29 illustrate this register.

Figure 29-28. SCI Pin I/O Control Register 6 (SCIPIO6) [offset = 54h]

31	8
Reserved	
R-0	

7	3	2	1	0
Reserved		TX PDR	RX PDR	Reserved
R-0		R/W-0	R/W-0	R/W-0

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

Table 29-29. SCI Pin I/O Control Register 6 (SCIPIO6) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TX PDR		Transmit pin open drain enable. This bit enables open-drain capability in the output pin SCITX if the following conditions are met: • TX FUNC = 0 (SCITX pin is a general-purpose I/O.) • TX DIR = 1 (SCITX pin is a general-purpose output.)
		0	Open drain functionality is disabled; the output voltage is V_{OL} or lower if TXOUT = 0 and V_{OH} or higher if TXOUT = 1.
		1	Open drain functionality is enabled; the output voltage is V_{OL} or lower if TXOUT = 0 and high impedance if TXOUT = 1.
1	RX PDR		Receive pin open drain enable. This bit enables open-drain capability in the output pin SCIRX if the following conditions are met: • RX FUNC = 0 (SCIRX pin is a general-purpose I/O.) • RX DIR = 1 (SCIRX pin is a general-purpose output.)
		0	Open drain functionality is disabled; the output voltage is V_{OL} or lower if RXOUT = 0 and V_{OH} or higher if RXOUT = 1.
		1	Open drain functionality is enabled; the output voltage is V_{OL} or lower if RXOUT = 0 and high impedance if RXOUT = 1.
0	Reserved	0	Reads return 0. Writes have no effect.

29.7.20 SCI Pin I/O Control Register 7 (SCIPIO7)

Figure 29-29 and Table 29-30 illustrate this register.

Figure 29-29. SCI Pin I/O Control Register 7 (SCIPIO7) [offset = 58h]

31	8
Reserved	
R-0	

7	3	2	1	0
Reserved		TX PD	RX PD	Reserved
R-0		R/W-n	R/W-n	R/W-n

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset, Refer to the Terminal Functions in the device datasheet for default pin settings.

Table 29-30. SCI Pin I/O Control Register 7 (SCIPIO7) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TX PD		Transmit pin pull control disable. This bit disables pull control capability on the input pin SCITX.
		0	The pull control on the SCITX pin is enabled.
		1	The pull control on the SCITX pin is disabled.
1	RX PD		Receive pin pull control disable. This bit disables pull control capability on the input pin SCIRX.
		0	Pull control on the SCIRX pin is enabled.
		1	Pull control on the SCIRX pin is disabled.
0	Reserved	0	Writes have no effect.

29.7.21 SCI Pin I/O Control Register 8 (SCIPIO8)

Figure 29-30 and Table 29-31 illustrate this register.

Figure 29-30. SCI Pin I/O Control Register 8 (SCIPIO8) [offset = 5Ch]

31	8
Reserved	
R-0	

7	3	2	1	0
Reserved		TX PSL	RX PSL	Reserved
R-0		R/W-n	R/W-n	R/W-n

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset, Refer to the Terminal Functions in the device datasheet for default pin settings.

Table 29-31. SCI Pin I/O Control Register 8 (SCIPIO8) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reads return 0. Writes have no effect.
2	TX PSL		TX pin pull select. This bit selects pull type in the input pin SCITX.
		0	The SCITX pin is a pull down.
		1	The SCITX pin is a pull up.
1	RX PSL		RX pin pull select. This bit selects pull type in the input pin SCIRX.
		0	The SCIRX pin is a pull down.
		1	The SCIRX pin is a pull up.
0	Reserved	0	Writes have no effect.

29.7.22 Input/Output Error Enable (IODFTCTRL) Register

Figure 29-31 and Table 29-32 illustrate this register. After the basic SCI module configuration, enable the required Error mode to be created followed by IODFT Key enable.

NOTE:

1. All the bits are used in IODFT mode only.
2. Each IODFT are expected to be checked individually.

Figure 29-31. Input/Output Error Enable Register (IODFTCTRL) [offset = 90h]

31	27	26	25	24
Reserved		FEN	PEN	BRKDTENA
R-0		R/W-0	R/WC-0	R/WC-0

23	21	20	19	18	16
Reserved		PIN SAMPLE MASK		TX SHIFT	
R-0		R/W-0		R/W-0	

15	12	11			8
Reserved		IODFTENA			
R-0		R/WP-0	R/WP-1	R/WP-0	R/WP-1

7	2	1	0
Reserved		LPB ENA	RXPENA
R-0		R/WP-0	R/WP-0

LEGEND: R/W = Read/Write; R = Read only; WC = Write in sci-compatible mode only; WP = Write in privilege mode only; -*n* = value after reset

Table 29-32. Input/Output Error Enable Register (IODFTCTRL) Field Descriptions

Bit	Field	Value	Description
31-27	Reserved	0	Reads return 0. Writes have no effect.
26	FEN		Frame error enable. This bit is used to create a frame error.
		0	No error is created.
		1	The stop bit received is ANDed with 0 and passed to the stop bit check circuitry.
25	PEN		Parity error enable. This bit is used to create a parity error.
		0	No parity error occurs.
		1	The parity bit received is toggled so that a parity error occurs.
24	BRKD TENA		Break detect error enable. This bit is used to create a BRKDT error.
		0	No error is created.
		1	The stop bit of the frame is ANDed with 0 and passed to the RSM so that a frame error occurs. Then the RX pin is forced to continuous low for 10 T_{BITS} so that a BRKDT error occurs.
32-21	Reserved	0	Reads return 0. Writes have no effect.
20-19	PIN SAMPLE MASK		Pin sample mask. These bits define the sample number at which the TX pin value that is being transmitted will be inverted to verify the receive pin samples majority detection circuitry.
		0	No mask is used.
		1h	Invert the TX Pin value at 7th.SCLK.
		2h	Invert the TX Pin value at 8th.SCLK.
		3h	Invert the TX Pin value at 9th.SCLK.

Table 29-32. Input/Output Error Enable Register (IODFTCTRL) Field Descriptions (continued)

Bit	Field	Value	Description
18-16	TX SHIFT		Transmit shift. These bits define the amount by which the value on TX pin is delayed so that the value on the RX pin is asynchronous. This feature is not applicable to the start bit.
		0	No delay occurs.
		1h	The value is delayed by 1 SCLK.
		2h	The value is delayed by 2 SCLK.
		3h	The value is delayed by 3 SCLK.
		4h	The value is delayed by 4 SCLK.
		5h	The value is delayed by 5 SCLK.
		6h	The value is delayed by 6 SCLK.
		7h	No delay occurs.
15-12	Reserved	0	Reads return 0. Writes have no effect.
11-8	IODFTENA		IODFT enable key. Write access permitted in Privilege mode only.
		Ah	IODFT is enabled.
		All Others	IODFT is disabled.
7-2	Reserved	0	Reads return 0. Writes have no effect.
1	LPBENA		Module loopback enable. Write access permitted in Privilege mode only. **Note: In analog loopback mode the complete communication path through the I/Os can be tested, whereas in digital loopback mode the I/O buffers are excluded from this path.**
		0	Digital loopback is enabled.
		1	Analog loopback is enabled in module I/O DFT mode when IODFTENA = 1010.
0	RXPENA		Module analog loopback through receive pin enable. Write access permitted in Privilege mode only. This bit defines whether the I/O buffers for the transmit or the receive pin are included in the communication path (in analog loopback mode).
		0	Analog loopback through the transmit pin is enabled.
		1	Analog loopback through the receive pin is enabled.

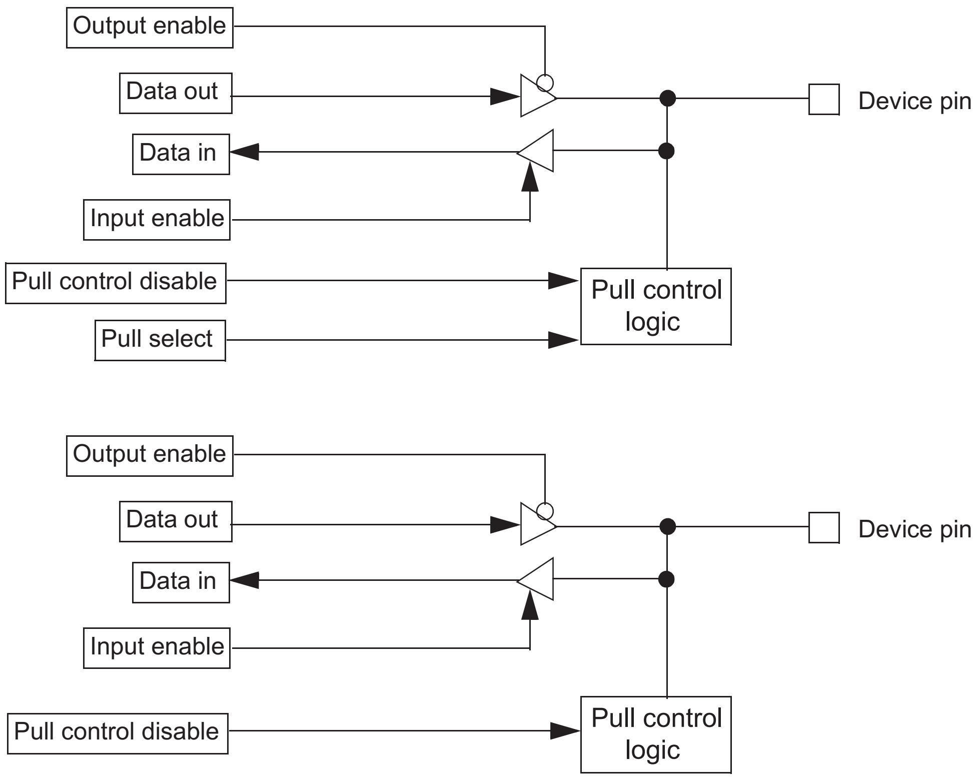
29.8 GPIO Functionality

The following sections apply to all device pins that can be configured as functional or general-purpose I/O pins.

29.8.1 GPIO Functionality

Figure 29-32 illustrates the GPIO functionality.

Figure 29-32. GPIO Functionality



29.8.2 Under Reset

The following apply if a device is under reset:

- Pull control. The reset pull control on the pins is enabled.
- Input buffer. The input buffer is enabled.
- Output buffer. The output buffer is disabled.

29.8.3 Out of Reset

The following apply if the device is out of reset:

- Pull control. The pull control is enabled by clearing the PD (pull control disable) bit in the SCIPIO7 register (Section 29.7.20). In this case, if the PSL (pull select) bit in the SCIPIO8 register (Section 29.7.21) is set, the pin will have a pull-up. If the PSL bit is cleared, the pin will have a pull-down. If the PD bit is set in the control register, there is no pull-up or pull-down on the pin.
- Input buffer. The input buffer is always enabled in functional mode.

NOTE: The pull-disable logic depends on the pin direction. It is independent of whether the device is in I/O or functional mode. If the pin is configured as output or transmit, then the pulls are disabled automatically. If the pin is configured as input or receive, the pulls are enabled or disabled depending on bit PD in the pull disable register SCIPIO7 (Section 29.7.20).

- Output buffer. A pin can be driven as an output pin if the TX DIR bit is set in the pin direction control register (SCIPIO1; Section 29.7.14) AND the open-drain feature is not enabled in the SCIPIO6 register (Section 29.7.19).

29.8.4 Open-Drain Feature Enabled on a Pin

The following apply if the open-drain feature is enabled on a pin:

- The output buffer is enabled, if a low signal is being driven on to the pin.
- The output buffer is disabled (the direction control signal DIR is internally forced low), if a high signal is being driven on to the pin.

NOTE: The open-drain feature is available only in I/O mode (SCIPIO0; Section 29.7.13).

29.8.5 Summary

The behavior of the input buffer, output buffer, and the pull control is summarized in Table 29-33.

Table 29-33. Input Buffer, Output Buffer, and Pull Control Behavior as GPIO Pins

Device under Reset?	Pin Direction (DIR) [(1)(2)]	Pull Disable (PULDIS) [(1)(3)]	Pull Select (PULSEL) [(1)(4)]	Pull Control	Output Buffer	Input Buffer
Yes	X	X	X	Enabled	Disabled	Enabled
No	0	0	0	Pull down	Disabled	Enabled
No	0	0	1	Pull up	Disabled	Enabled
No	0	1	0	Disabled	Disabled	Enabled
No	0	1	1	Disabled	Disabled	Enabled
No	1	X	X	Disabled	Enabled	Enabled

[(1)] X = Don’t care

[(2)] DIR = 0 for input, = 1 for output

[(3)] PULDIS = 0 for enabling pull control
= 1 for disabling pull control

[(4)] PULSEL= 0 for pull-down functionality
= 1 for pull-up functionality

Chapter 30

SPNU514C – March 2018

Inter-Integrated Circuit (I2C) Module

This chapter describes the inter-integrated circuit (I2C or I^{2}C) module. The I2C is a multi-master communication module providing an interface between the Texas Instruments (TI) microcontroller and devices compliant with Philips Semiconductor I^{2}C-bus specification version 2.1 and connected by an I2C-bus. This module will support any slave or master I2C compatible device.

Topic **Page**

30.1 Overview **1656**

30.2 I2C Module Operation **1660**

30.3 I2C Operation Modes **1664**

30.4 I2C Module Integrity **1666**

30.5 Operational Information **1668**

30.6 I2C Control Registers **1671**

30.7 Sample Waveforms **1692**

30.1 Overview

The I2C has the following features:

- Compliance to the Philips I^{2}C bus specification, v2.1 (*The I2C Specification,* Philips document number 9398 393 40011)
  - Bit/Byte format transfer
  - 7-bit and 10-bit device addressing modes
  - General call
  - START byte
  - Multi-master transmitter/ slave receiver mode
  - Multi-master receiver/ slave transmitter mode
  - Combined master transmit/receive and receive/transmit mode
  - Transfer rates of 10 kbps up to 400 kbps (Phillips fast-mode rate)
- Free data format
- Two DMA events (transmit and receive)
- DMA event enable/disable capability
- Seven interrupts that can be used by the CPU
- Operates with VBUS frequency from 6.7 MHz up
- Operates with module frequency between 6.7 MHz to 13.3 MHz
- Module enable/disable capability
- The SDA and SCL are optionally configurable as general purpose I/O
- Slew rate control of the outputs
- Open drain control of the outputs
- Programmable pullup/pulldown capability on the inputs
- Supports Ignore NACK mode

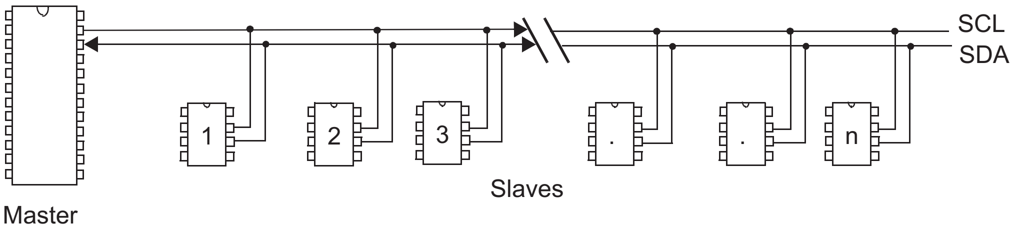
NOTE: This I2C module does **not** support:

- High-speed (HS) mode
- C-bus compatibility mode
- The combined format in 10-bit address mode (the I2C sends the slave address second byte every time it sends the slave address first byte)

30.1.1 Introduction to the I2C Module

The I2C module supports any slave or master I2C-compatible device. Figure 30-1 shows an example of multiple I2C serial ports connected for a two-way transfer from one device to another device.

Figure 30-1. Multiple I2C Modules Connection Diagram



30.1.2 Functional Overview

The I2C module is a serial bus that supports multiple master devices. In multimaster mode, one or more devices can be connected to the same bus and are capable of controlling the bus. Each I2C device on the bus is recognized by a unique address and can operate as either a transmitter or a receiver, depending on the function of the device. In addition to being a transmitter or receiver, a device connected to the I2C bus can also be considered a master or a slave when performing data transfers.

NOTE: A master device is the device that initiates the data transfer on a bus and generates the clock signal that permits the transfer. During the transmission, any device addressed by the master is considered the slave.

Data is communicated to devices interfacing to the I2C module using the serial data pin (SDA) and the serial clock pin (SCL) as shown in Figure 30-2. These two wires carry information between the device and the other devices connected to the I2C bus. Both SDA and SCL pins on the device are bidirectional. They must be connected to a positive supply voltage through 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 wired-AND function.

The device has a special mode that can be entered to ignore a NACK generated from non-compliant I2C devices that are incapable of generating an ACK.

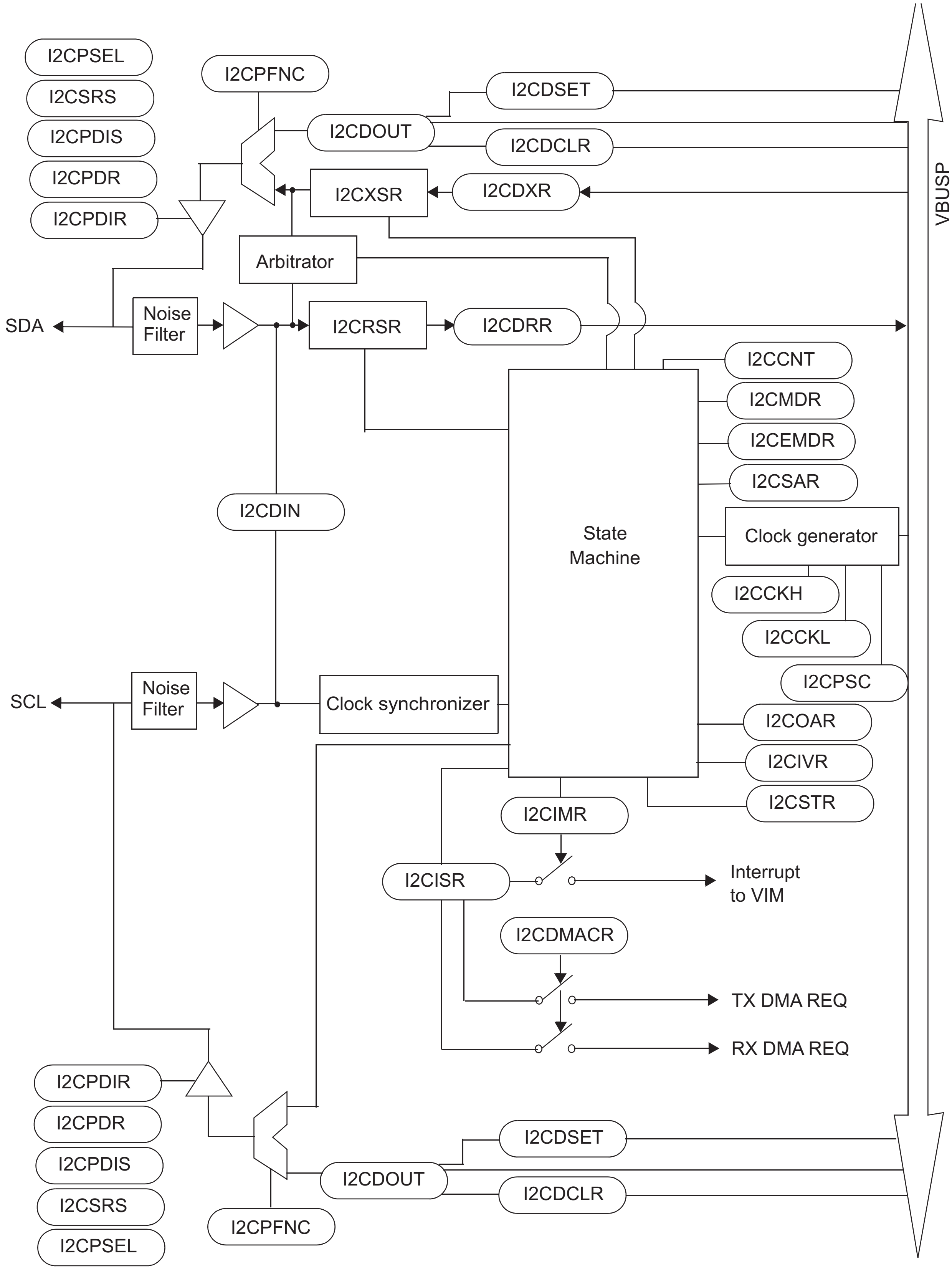
The I2C module consists of the following primary blocks:

- A serial Interface: one data pin (SDA) and one clock pin (SCL)
- The device register interface
  - Data registers to temporarily hold received data and transmitted data traveling between the SDA pin and the CPU or the DMA
  - Control and status registers
- A prescaler to divide down the input clock that is driven to the I2C module
- A peripheral bus interface to enable the CPU and DMA to access the I2C module registers
- An arbitrator to handle arbitration between the I2C module (when configured as a master) and another master
- Interrupt generation logic (interrupts can be sent to the CPU)
- A clock synchronizer that synchronizes the I2C input clock (from the system module) and the clock on the SCL pin, and synchronizes data transfers with masters of different clock speeds
- A noise filter on each of the two serial pins
- DMA event generation logic that synchronizes data reception and data transmission in the I2C module for DMA transmission

  In Figure 30-2, the CPU or the DMA writes data for transmission to I2CDXR and reads received data from I2CDRR. When the I2C module is configured as a transmitter, data written to I2CDXR is copied to I2CXSR and shifted out one bit at a time. When the I2C module is configured as a receiver, received data is shifted into I2CRSR and then copied to I2CDRR.

  When the I2C function is not needed, the pins may be controlled as general-purpose input/output (GPIO) pins. The I/O structure of each pin includes:
- programmable slew rate control of the outputs
- open drain mode
- programmable pull enable/disable on the input
- programmable pull up/pull down function on the input

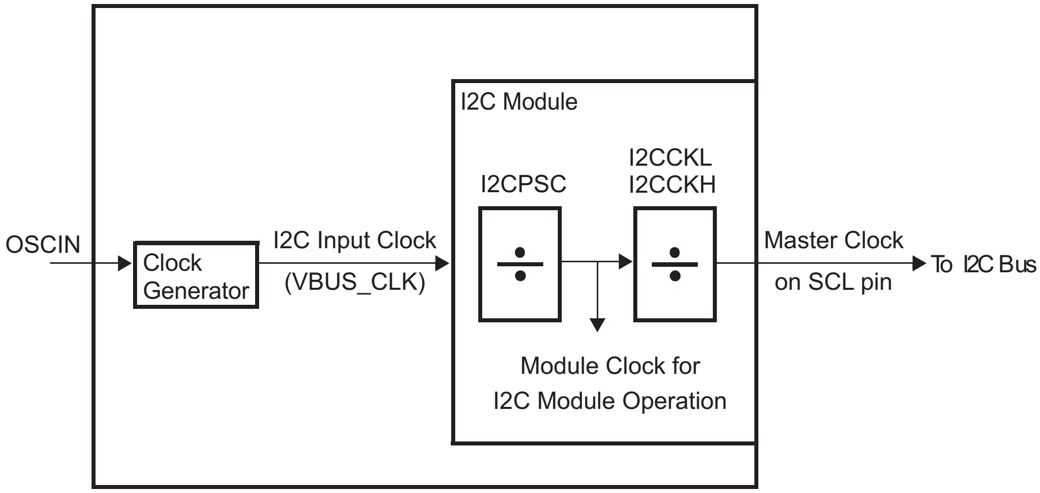
Figure 30-2. Simple I2C Block Diagram



30.1.3 Clock Generation

As shown in Figure 30-3, the I2C module uses the input clock generated from the device clock generator to generate the module clock and master clock. The I2C input clock is the device peripheral clock (VBUS_CLK). The clock is then divided twice more inside the I2C module to produce the module clock and the master clock.

Figure 30-3. Clocking Diagram for the I2C Module



The module clock determines the frequency at which the I2C module operates. A programmable prescaler in the I2C module divides down the input clock to produce the module clock. To specify the divide-down value, initialize the I2CPSC field of the prescaler register, I2CPSC. The resulting frequency is:

$$ModuleClockFrequency = \frac{I2CInputClockFrequency}{(I2CPSC + 1)} \tag{63}$$

The module clock frequency must be between 6.7MHz and 13.3MHz. The prescaler can only be initialized while the I2C module is in the reset state (IRS = 0 in I2CMDR). The prescaled frequency takes effect only when IRS is changed to 1. Changing the I2CPSC value while IRS = 1 has no effect.

The master clock appears on the SCL pin when the I2C module is configured to be a master on the I2C bus. This clock controls the timing of the communication between the I2C module and a slave. As shown in Figure 30-3, a second clock divider in the I2C module divides down the module clock to produce the master clock. The clock divider uses the I2CCKL to divide down the low portion of the module clock signal and uses the I2CCKH to divide down the high portion of the module clock signal.

The resulting frequency is:

$$MasterClockFrequency = \frac{ModuleClockFrequency}{(I2CCKL + d) + (I2CCKH + d)} \tag{64}$$

$$MasterClockFrequency = \frac{I2CInputClockFrequency}{(I2CPSC + 1)((I2CCKL + d) + (I2CCKH + d))} \tag{65}$$

where *d* depends on the value of I2CPSC:

I2CPSC	d
0	7
1	6
Greater than 1	5

NOTE: The master clock frequency defined above does not include rise/fall time and latency of the synchronizer inside the module. The actual transfer rate will be slower than the value calculated from the formula above. Also, due to the nature of SCL synchronization, the SCL clock period could change if SCL synchronization is taking place.

30.2 I2C Module Operation

The following section discusses how the I2C module operates.

30.2.1 Input and Output Voltage Levels

One clock pulse is generated by the master device for each data bit transferred. Because of a variety of different technology devices that can be connected to the I2C-bus, the levels of logic 0 (low) and logic 1 (high) are not fixed and depend on the associated level of V_{CCIO}. For details, see the device specific data sheet.

30.2.2 I2C Module Reset Conditions

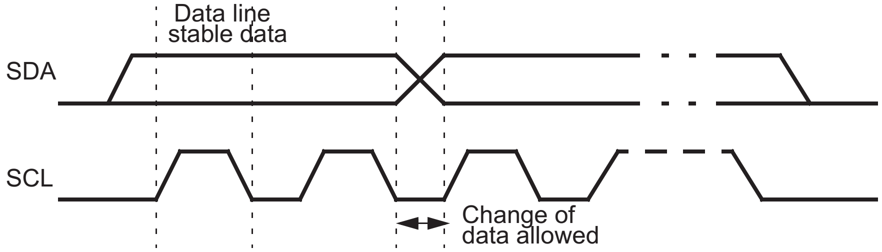
The I2C module can be reset in the following two ways:

- Through the global peripheral reset. A device reset causes a global peripheral reset.
- By clearing the $\overline{\text{IRS}}$ bit in the I2C mode register (I2CMDR). When the global peripheral reset is removed, the $\overline{\text{IRS}}$ bit is cleared to 0, keeping the I2C module in the reset state.

30.2.3 I2C Module Data Validity

The data on the SDA must be stable during the high period of the clock. See Figure 30-4. The high and low state of the data line, the SDA, can only change when the clock signal is low.

Figure 30-4. Bit Transfer on the I2C Bus

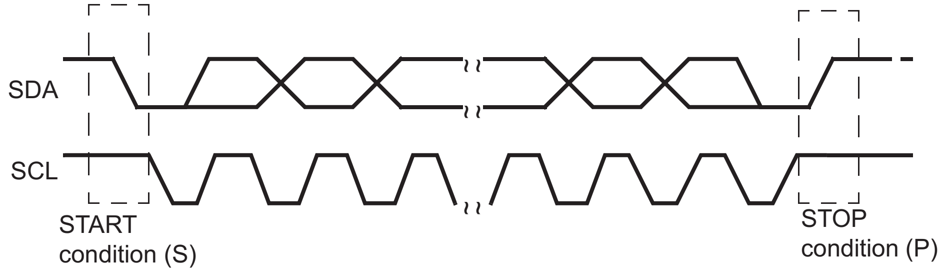


30.2.4 I2C Module Start and Stop Conditions

START and STOP conditions are generated by a master I2C module.

- The START condition is defined as a high-to-low transition on the SDA line while SCL is high. A master drives this condition to indicate the start of data transfer. The bus is considered to be busy after the START condition, and the bus busy bit (BB) in I2CSR is set to 1.
- The STOP condition is defined as a low-to-high transition on the SDA line while SCL is high. A master drives this condition to indicate the end of data transfer. The bus is considered to be free after the STOP condition, therefore the BB bit in I2CSR is cleared to 0.

Figure 30-5. I2C Module START and STOP Conditions

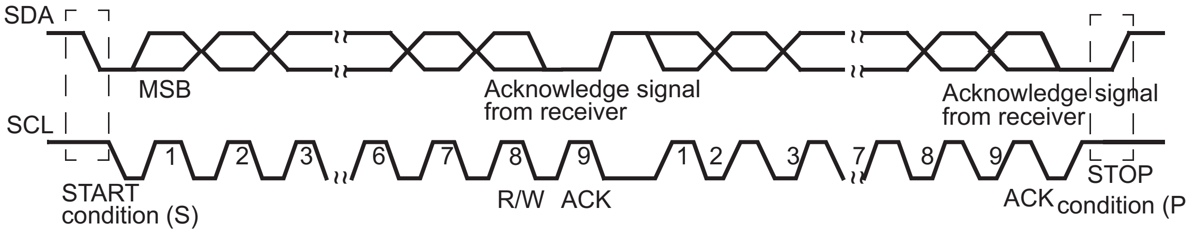


For the I2C module to start a data transfer with a START condition, the master mode bit (MST) and the START condition bit (STT) in the I2CMDR must both be set to 1. For the I2C module to end a data transfer with a STOP condition, the STOP condition bit (STP) must be set to 1. When the BB bit is set to 1 and the STT bit is set to 1, a repeated START condition is generated.

30.2.5 Serial Data Formats

The I2C module operates in byte data format. Each message put on the SDA line is 2 to 8-bits long. The number of messages that can be transmitted or received is unrestricted. The data is transferred with the most significant bit (MSB) first (Figure 30-6). Each message is followed by an acknowledge bit from the I2C if it is in receiver mode. The I2C module does not support little endian systems.

Figure 30-6. I2C Module Data Transfer



The first byte after a START condition (S) always consists of 8 bits that comprise either a 7-bit address plus the R/$\overline{\text{W}}$ bit, or 8 data bits. The eighth bit, R/W, in the first byte determines the direction of the data. When the R/$\overline{\text{W}}$ bit is 0, the master writes (transmits) data to a selected slave device; when the R/$\overline{\text{W}}$ bit is 1, the master reads (receives) data from the slave device. In acknowledge mode, an extra bit dedicated for the acknowledgement (ACK) bit is inserted after each message.

The I2C module supports the following formats:

- 7-bit addressing format (Figure 30-7)
- 10-bit addressing format (Figure 30-8)
- 7-bit/10-bit addressing format with repeated START condition (Figure 30-9)
- Free-data format (Figure 30-10)

30.2.5.1 7-Bit Addressing Format

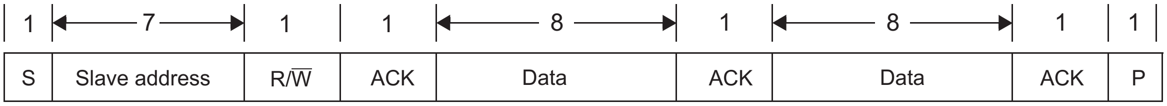
In the 7-bit addressing format (Figure 30-7), the first byte after the START condition consists of a 7-bit slave address followed by the R/$\overline{W}$ bit (in the LSB). The R/$\overline{W}$ bit determines the direction of the data transfer:

- 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 acknowledgement (ACK) is inserted after each byte. If the ACK is inserted by the slave after the first byte from the master, it is followed by n bits of data from the transmitter (master or slave, depending on the R/$\overline{W}$ bit). The device I2C allows n to be a number between 2 to 8, programmable by the bit count (BC) field of I2CMDR. After the data bits have been transferred, the receiver inserts an ACK bit.

To select the 7-bit addressing format, write 0 to the expanded address enable (XA) bit of I2CMDR and make sure the free data format mode is off (FDF = 0 in I2CMDR).

Figure 30-7. I2C Module 7-Bit Addressing Format



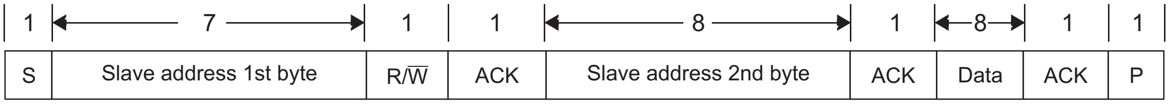
1	7	1	1	8	1	8	1	1
S	Slave address	R/$\overline{W}$	ACK	Data	ACK	Data	ACK	P

30.2.5.2 10-Bit Addressing Format

The 10-bit addressing format is similar to the 7-bit addressing format, but the master sends the slave address in two separate byte transfers. In the 10-bit addressing format (Figure 30-8), the first byte is 11110b, the two MSBs of the 10-bit slave address, and the R/$\overline{W}$ bit. The ACK bit is inserted after each byte. The second byte is the remaining 8 bits of the 10-bit slave address. The slave must send an acknowledgement 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 repeated a START condition to change the data direction.

To select the 10-bit addressing format, write 1 to the expanded address enable (XA) bit of I2CMDR and make sure the free data format mode is off (FDF = 0 in I2CMDR).

Figure 30-8. I2C Module 10-bit Addressing Format

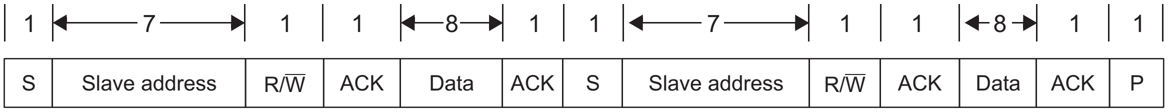


1	7	1	1	8	1	8	1	1
S	Slave address 1st byte	R/$\overline{W}$	ACK	Slave address 2nd byte	ACK	Data	ACK	P

30.2.5.3 Using the Repeated START Condition

At the end of each byte, the master can drive another START condition (Figure 30-9). Using this capability, a master can transmit/receive any number of data bytes before generating a STOP condition. The length of a data byte can be from 2 to 8 bits. The repeated START condition can be used with the 7-bit addressing, 10-bit addressing, or the free data formats.

Figure 30-9. I2C Module 7-Bit Addressing Format with Repeated START



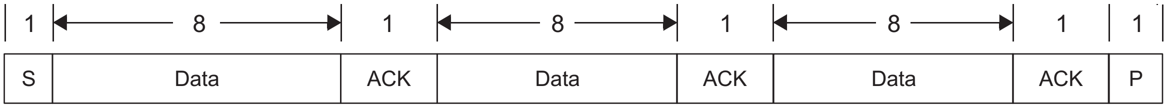
1	7	1	1	8	1	1	7	1	1	8	1	1
S	Slave address	R/$\overline{W}$	ACK	Data	ACK	S	Slave address	R/$\overline{W}$	ACK	Data	ACK	P

30.2.5.4 Free Data Format

In this format (Figure 30-10), the first byte after a START condition is a data byte. The ACK bit is inserted after each byte, followed by another 8 bits of data. No address or data direction bit is sent. Therefore, the transmitter and receiver must both support the free data format. The direction of data transmission (transmit or receive) remains constant throughout the transfer.

To select the free data format, write a 1 to the free data format (FDF) bit of the I2CMDR. The free data format is not supported in the digital loop back mode.

Figure 30-10. I2C Module in Free Data Format



30.2.6 NACK Bit Generation

When the I2C module is a receiver (master or slave), it can acknowledge or ignore bits sent by the transmitter. To ignore any new bits, the I2C module must send a no-acknowledge (NACK) bit during the acknowledge cycle on the bus. Table 30-1 summarizes the various ways a NACK can be generated.

Table 30-1. Ways to Generate a NACK Bit

I2C Module Condition	Basic NACK Bit Generation Options	Additional Option
Slave receiver mode	Disable data transfers (STT = 0) Allow an overrun condition (RSFULL = 1) Reset the module (IRS = 0)	Set the NACKMOD bit before the rising edge of the last data bit you intend to receive.
Master receiver mode and repeat mode (RM = 1)	Generate a STOP condition (STP = 1) Reset the module (IRS = 0)	Set the NACKMOD bit before the rising edge of the last data bit you intend to receive.
Master receiver mode with non-repeat mode (RM = 0)	If STP = 1, allow the internal data counter to count down to 0 and thus force a STOP condition. If STP = 0, make STP = 1 to generate a STOP condition. Reset the module (IRS = 0)	Set the NACKMOD bit before the rising edge of the last data bit you intend to receive.

In some applications, the slave cannot generate the ACK signal. If the IGNACK bit is set in the I2CEMDR register, the resulting NACK will be ignored and the I2C block will continue the data transfer.

30.3 I2C Operation Modes

30.3.1 Master Transmitter Mode

All masters begin in this mode. The I2C module is a master and transmits control information and data to a slave. In this mode, data assembled in any of the addressing formats shown in Figure 30-7, Figure 30-8, or Figure 30-9 is shifted out onto the SDA pin and synchronized with the self-generated clock pulses on the SCL pin. The clock pulses are inhibited and the SCL pin is held low when the intervention of the device is required ($\overline{\text{XSMT}}$ = 0) after a byte has been transmitted.

NOTE: If the I2C is configured for two simultaneous master transmissions, wait until the MST and BB have been reset before performing the second master transmission.

Failure to wait for the MST and BB to reset will prevent the start condition on the second transfer from being issued and the bus BB will not be set. Typically the end of the first transfer is handled by polling BB. However, the MST bit is not reset at the same instant as the BB bit. As a result, when the second master transmission is initiated before the resetting of the MST, the MST bit for the second transfer is reset. This prevents the I2C from recognizing itself as the master, thus failing to occupy the bus.

30.3.2 Master Receiver Mode

In this mode, the I2C module is a master and receives data from a slave. This mode can only be entered from the master transmitter mode (the I2C module must first transmit a command to the slave). In any of the addressing formats shown in Figure 30-7, Figure 30-8, or Figure 30-9, the master receiver mode is entered after the slave address byte and the R/$\overline{\text{W}}$ bit have been transmitted (if the R/$\overline{\text{W}}$ bit is 1). Serial data bits received on the SDA pin are shifted in with the self-generated clock pulses on the SCL pin. The clock pulses are inhibited and the SCL is held low when the intervention of the device is required (RSFULL = 1) after a byte has been received. At the end of the transfer, the master-receiver signals the end of data to the slave-transmitter by not generating an acknowledge on the last byte that was clocked out of the slave. The slave-transmitter then releases the data line allowing the master-receiver to generate a STOP condition or a repeated START condition.

In many applications, the size of the message is in the initial bytes of the message itself. Since the size of the message is not known to the master before the transmission/reception starts, the master must use the repeat mode in order to force the stop condition when the reception is completed. The repeat mode is enabled by setting the RM bit to 1. Due to the double buffer implementation on the receive side, the master must generate the stop condition (STP =1) after reading the (message size - 1)$^{\text{th}}$ data.

30.3.3 Slave Transmitter Mode

In this mode, the I2C module is a slave and transmits data to a master. This mode can only be entered from the slave receiver mode (The I2C module must first receive a command from the master). In any of the addressing formats shown in Figure 30-7, Figure 30-8, or Figure 30-9, the slave transmitter mode is entered if the slave address byte is the same as its own address and the R/$\overline{\text{W}}$ bit has been transmitted (if the R/$\overline{\text{W}}$ bit is set to 1). The slave transmitter shifts the serial data out on the SDA pin with the clock pulses that are generated by the master device. The slave device does not generate the clock, but it can hold the SCL pin low when intervention of the device is required ($\overline{\text{XSMT}}$ = 0) after a byte has been transmitted.

30.3.4 Slave Receiver Mode

In this mode, the I2C module is a slave and receives data from a master. All slaves begin in this mode. Serial data bits received on the SDA pin are shifted in with the clock pulses that are generated by the master device. The slave device does not generate the clock, but it can hold the SCL pin low while intervention of the device is required (RSFULL = 1) after a byte has been received.

30.3.5 Low Power Mode

The I2C module can be placed in low-power mode by a global low-power mode initiated by the system (by writing to the Peripheral Power-Down Set Register in the Peripheral Central Resource (PCR) module.

In effect, low-power mode shuts down all the clocks to the module. In global low-power mode, no registers are visible to the software; nothing can be written to or read from any register.

30.3.6 Free Run Mode

The I2C module can be placed in free run mode when the FREE bit (I2CMDR.14) is set to 1. This bit is primarily used on an emulator when encountering a breakpoint while debugging software. When the FREE bit is set to 0, the I2C responds differently depending on whether the SCL is high or low. If the SCL is low, the I2C stops immediately and keeps driving the SCL low whether the I2C is the master transmitter or receiver. If the SCL is high, the I2C waits until the SCL becomes a low and then stops. If the I2C is a slave, it stops when the transmission/reception completes.

30.3.7 Ignore NACK Mode

The I2C module can be placed in the ignore NACK mode by setting the IGNACK bit in the I2CEMDR register. This mode allows an I2C module that is configured as a master transmitter to ignore a NACK from a slave device that is not capable of generating a proper ACK signal.

30.4 I2C Module Integrity

The following section discusses how the I2C module maintains priorities and order among signals and commands.

30.4.1 Arbitration

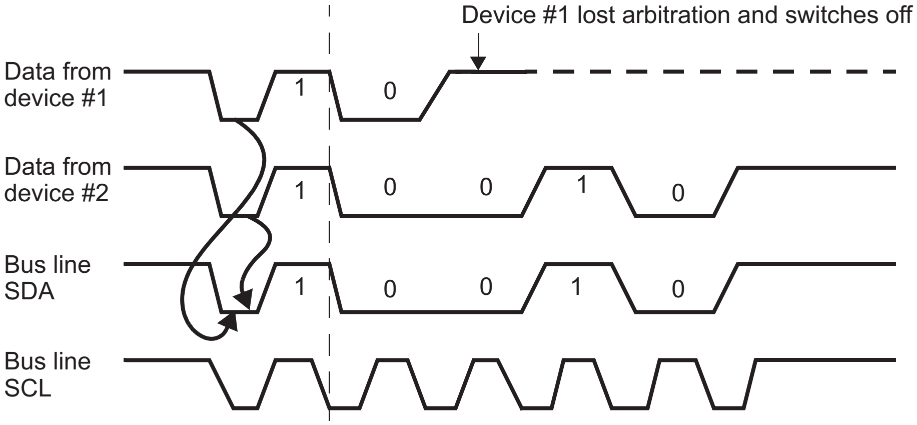
If two or more master transmitters simultaneously start a transmission on the same bus, an arbitration procedure is invoked. Figure 30-11 illustrates the arbitration procedure between two devices. The arbitration procedure uses the data presented on the SDA bus by the competing transmitters. The first master transmitter that generates a high is overruled by the other master that generates a low. The arbitration procedure gives priority to the device that transmits the serial data stream with the lowest binary value. The master transmitter that loses the arbitration switches to the slave receiver mode, sets the arbitration lost (AL) flag, and generates the arbitration-lost interrupt. The data transmitted by the other master module is salvaged, and the I2C continues to receive data from the master module. Should two or more devices send identical first bytes, arbitration continues on the subsequent bytes.

If, during a serial transfer, the arbitration procedure is still in progress when a repeated START condition or STOP condition is transmitted to I2C bus, the master transmitters involved must send the repeated START condition or STOP condition at the same position in the format frame. In other words, 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

  Slaves are not involved in the arbitration procedure.

Figure 30-11. Arbitration Procedure Between Two Master Transmitters



30.4.2 I2C Clock Generation and Synchronization

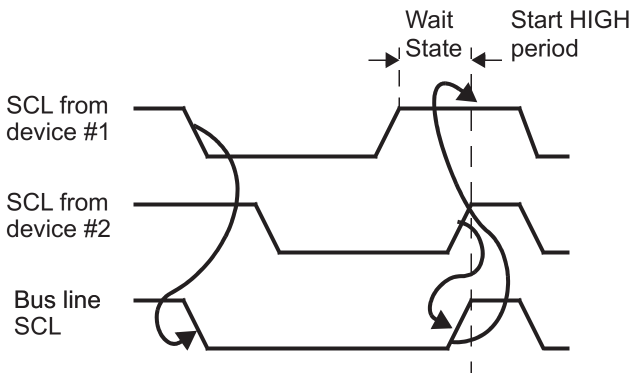
Under normal conditions only one master device generates the clock signal; the SCL. During the arbitration procedure, however, there are two or more master devices and the clock must be synchronized so that the data output can be compared. Figure 30-12 illustrates clock synchronization. The wired-AND property of the SCL line means that a device that first generates a low period on the SCL overrules the other devices. At this high-to-low transition, the clock generators of the other devices are forced to start their own low period. The SCL line is held low by the device with the longest low period. The other devices that finish their low periods must wait for the SCL line to be released before starting their high periods. A synchronized signal on the SCL is obtained where the slowest device determines the length of the low period and the fastest device determines the length of the high period.

If a device pulls down the clock line for a longer time, the result is that all clock generators must enter the wait state. In this way, a slave slows down a fast master and the slow device creates enough time to store a received byte or to prepare a byte to be transmitted.

NOTE: I2C Protocol Fault

The following conditions violate the clock spec as defined in the Philips I^2C bus specification, v2.1 (*The I2C Specification,* Philips document number 9398 393 40011), and will result in an I2C protocol fault: I2CCLKH = 2 I2CCLKL = 2I2CPSC = 2. This will cause the SDA data transition to occur while the SCL is high.

Figure 30-12. Synchronization of Two I2C Clock Generators During Arbitration



30.4.3 Prescaler

The I2C module is operated by the module clock. This clock is generated by way of the I2C prescaler block. The prescaler block consists of a 8-bit register, I2CPSC, used for dividing down the device peripheral clock (VBUS_CLK) to obtain a module clock between 6.7 MHz and 13.3 MHz.

30.4.4 Noise Filter

The noise filter is used to suppress any noises that are 50ns or less. It is designed to suppress noise with one module clock, assuming the lower and upper limits of the module clock are 6.7MHz and 13.3MHz, respectively.

30.5 Operational Information

The following section provides specific information about how the I2C module operates.

30.5.1 I2C Module Interrupts

The I2C module generates seven types of interrupts. These seven interrupts are accompanied with seven interrupt mask bits in the interrupt mask register (I2CIMR) and with seven interrupt flag bits in the status register (I2CSR).

30.5.1.1 I2C Interrupt Requests

The I2C module generates the interrupt requests described below. All requests are multiplexed through an arbiter into a single I2C interrupt request to the CPU. Each interrupt request has a flag bit and an enable bit. Interrupts must be enabled prior to the occurrence of the expected interrupt condition. When one of the specified events occurs, the flag bit is set. If the corresponding enable bit is 0, the interrupt request is blocked. If the enable bit is 1, the interrupt request is forwarded to the CPU as an I2C interrupt request. As an alternative, the CPU can poll all of the bits shown in Table 30-2.

Table 30-2. Interrupt Requests Generated by I2C Module

Flag	Name	Generated
AL	Arbitration-lost interrupt	Generated when the I2C module has lost an arbitration contest with another master-transmitter
NACK	No-acknowledge interrupt	Generated when the master I2C does not receive an acknowledge from the receiver
ARDY	Register-access-ready interrupt	Generated when the previously programmed address, data and command have been performed and the status bits have been updated. The interrupt is used to notify the device that the I2C registers are ready to be accessed.
RXRDY	Receive-data-ready interrupt	Generated when the received data in the receive-shift register (I2CSR) has been copied into the data receive register (I2CDRR). The RXRDY bit can also be polled by the device to determine when to read the received data in the I2CDRR.
TXRDY	Transmit-data-ready interrupt	Generated when the transmitted data has been copied from the data transmit register (I2CDXR) into the transmit-shift register (I2CXSR). The TXRDY bit can also be polled by the device to determine when to write the next data into I2CDXR.
SCD	Stop-condition-detect interrupt	Generated when a STOP condition has been detected.
AAS	Address-as-slave interrupt	Generated when the I2C has recognized its own slave address or an address of all zeroes.

The interrupt vector register (I2CIVR) contains the binary-coded-interrupt vector that indicates the highest priority interrupt that is pending and enabled. When I2CIVR is read, the corresponding interrupt flags for AL, NACK and SCD are automatically cleared, if their interrupts are enabled. Reading the I2CIVR will not clear the AAS, ARDY, RXRDY, or TXRDY interrupt pending flags. Please see Section 30.6.3 for the method to clear these four flags.

If more than one interrupt is pending, a new interrupt will be generated for the next highest priority pending interrupt when you re-enable the I2C interrupt.

A transmit interrupt is generated just after the START condition in master transmitter mode. This ensures that the CPU will get an interrupt even if no slave returns an ACK to the slave address following the START condition.

It is important to note that when the I2C is configured to generate interrupts as a slave transmitter and the backward compatibility mode (BCM) bit is set to 1, an extra transmit interrupt occurs. The application should monitor the ACK from the master to determine whether to load another byte into I2CDXR.

30.5.2 DMA Controller Events

The I2C module has two events that use the DMA controller to synchronously read received data (I2CREVNT) from I2CDRR, and synchronously write data (I2CWEVNT) to the transmit buffer, I2CDXR. The read and write events have the same timing as I2CRRDY (I2CRINT) and I2CXRDY (I2CXINT), respectively.

The CPU or the DMA controller reads the received data from I2CDRR and writes the data to be transmitted to I2CDXR. The RXRDY bit is automatically cleared when the DMA controller reads the I2CDRR register, and the TXRDY bit is automatically cleared when the DMA controller writes to the I2CDXR register.

Data written to I2CDXR is copied to I2CXSR and shifted out from the SDA pin when the I2C module is configured as a transmitter. When the I2C module is configured as a receiver, received data is shifted into ICRSR and copied to I2CDRR, which can be read by the CPU or the DMA controller.

A transmit event (I2CWEVNT) is generated after a START condition in master transmitter mode. This ensures that the DMA gets an event even if no slave returns an ACK to the slave address following the START condition.

NOTE: Unexpected DMA transmit and receive event

An unexpected DMA transmit event (ICXEVT) and a DMA receive event (ICXRDY) are generated in 10-bit, master transmit, repeat mode. This event occurs soon after the start condition but before the first bit of the address is transmitted. In this event, no DMA activity should be initiated without the slave ACK being received.

30.5.3 I2C Enable/Disable

The I2C module can be enabled or disabled with the I2C reset enable bit (IRS) in the I2C module register (I2CMDR). This occurs in one of two ways:

- Write 0 to the I2C reset bit (IRS) in I2CMDR. All status bits are forced to the default values and the I2C mode remains disabled until IRS is changed to 1. The SDA and SCL pins are in the high impedance state.
- Initiate a device reset by driving the $\overline{\text{PORRST}}$ pin low. The entire device is reset and is held in the reset state until the pin is released and is driven high. When $\overline{\text{PORRST}}$ is released, all I2C module registers are reset to their default values. The IRS bit is forced to 0, which resets the I2C module. The I2C module stays in the reset state until a 1 is written to the IRS bit.

IRS must be 0 while the I2C module is being configured. Forcing IRS to 0 can be used to save power and also clear error conditions.

30.5.4 General Purpose I/O

Both of the I2C pins can be programmed to be general-purpose I/O pins via the I2C pin control registers (I2CPFNC, I2CDIR, I2CDOUT, and I2CDIN).

When the I2C module is not used, the I2C pins may be programmed to be either general purpose input or general-purpose output pins. This function is controlled in the I2CDIR and I2CPFNC registers. Note that each pin can be programmed to be either an I2C pin or a GIO pin.

If the I2C function is to be used, the application software must ensure that each pin is configured as an I2C pin and not a GIO pin, or else unexpected behavior may result.

30.5.5 Pull Up/Pull Down Function

I2C module pins can have either an active pull up or active pull down that makes it possible to leave the pins unconnected externally. The pins can be programmed to have the active pull function enabled or disabled by writing to the corresponding bit in the I2CPDIS register. Please see the device-specific data sheet for the default internal pull (pull-up, pull-down or no pull) on the pins.

The pull on the pins is programmable to a setting other than the default internal pull as specified in the data sheet. The pins can be programmed to have either an active pull up or an active pull down function by writing to the corresponding bit in I2CPSEL register. The pull up/pull down function is active on the pin only when the pull enabled is programmed in the I2CPDIS register.

The pull up/pull down functions are deactivated when a bidirectional pin is configured as an output. At system reset, the pull up function of all the pins is enabled. Please see the device-specific data sheet for the current supplied by the pull up/pull down.

30.5.6 Open Drain Function

The I2C pins can be programmed to include an open drain function when they are configured as output pins. This is done by writing to the corresponding bit of the I2CPDR register. When the open drain function is enabled, a low value (0) written to the data output register forces the pin to a low output voltage (V_{OL} or lower), whereas a high value (1) written to the data output register forces the pin to a high-impedance state. The open drain function is disabled when the pin is configured as an input pin.

30.6 I2C Control Registers

Table 30-3 provides a summary of the control registers. The upper word (upper 16 bits) of the registers all read as 0s. Writes have no effect on these bits. The base address for the control registers is FFF7 D400h.

Table 30-3. I2C Control Registers

Offset	Acronym	Register Description	Section
00h	I2COAR	I2C Own Address Manager	Section 30.6.1
04h	I2CIMR	I2C Interrupt Mask Register	Section 30.6.2
08h	I2CSTR	I2C Status Register	Section 30.6.3
0Ch	I2CCKL	I2C Clock Divider Low Register	Section 30.6.4
10h	I2CCKH	I2C Clock Control High Register	Section 30.6.5
14h	I2CCNT	I2C Data Count Register	Section 30.6.6
18h	I2CDRR	I2C Data Receive Register	Section 30.6.7
1Ch	I2CSAR	I2C Slave Address Register	Section 30.6.8
20h	I2CDXR	I2C Data Transmit Register	Section 30.6.9
24h	I2CMDR	I2C Mode Register	Section 30.6.10
28h	I2CIVR	I2C Interrupt Vector Register	Section 30.6.11
2Ch	I2CEMDR	I2C Extended Mode Register	Section 30.6.12
30h	I2CPSC	I2C Prescale Register	Section 30.6.13
34h	I2CPID1	I2C Peripheral ID Register 1	Section 30.6.14
38h	I2CPID2	I2C Peripheral ID Register 2	Section 30.6.15
3Ch	I2CDMACR	I2C DMA Control Register	Section 30.6.16
40h-44h	Reserved	Reserved	
48h	I2CPFNC	I2C Pin Function Register	Section 30.6.17
4Ch	I2CPDIR	I2C Pin Direction Register	Section 30.6.18
50h	I2CDIN	I2C Data Input Register	Section 30.6.19
54h	I2CDOUT	I2C Data Output Register	Section 30.6.20
58h	I2CDSET	I2C Data Set Register	Section 30.6.21
5Ch	I2CDCLR	I2C Data Clear Register	Section 30.6.22
60h	I2CPDR	I2C Pin Open Drain Register	Section 30.6.23
64h	I2CPDIS	I2C Pull Disable Register	Section 30.6.24
68h	I2CPSEL	I2C Pull Select Register	Section 30.6.25
6Ch	I2CSRS	I2C Pins Slew Rate Select Register	Section 30.6.26

30.6.1 I2C Own Address Manager (I2COAR)

The 16-bit memory-mapped I2C own address register is used to specify its own address. Figure 30-13 and Table 30-4 describe this register.

Figure 30-13. I2C Own Address Manager Register (I2COAR) [offset = 00h]

15	10	9	0
Reserved		OA	
R-0		R/W-0	

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

Table 30-4. I2C Own Address Manager Register (I2COAR) Field Descriptions

Bit	Field	Value	Description
15-10	Reserved	0	Reads return 0. Writes have no effect.
9-0	OA	0-3FFh	Own address These bits reflect the bus address of the I2C module. When the expand address (XA) bit I2CMDR.8 is set to 1, the I2C is in expand address mode (10-bit addressing mode). In either 7 or 10-bit address mode, all 10-bits are both readable and writable. Bits 7, 8, and 9 should only be used in 10-bit address mode. Table 30-5 provides the correct modes for these bits. Note that the user can program the I2C own address to any value as long as it does not conflict with other components in the system.

Table 30-5. Correct Mode for OA Bits

Bits Used	Mode	Value of XA
OA.6:0	7 Bit Addressing	0
OA.9:0	10 Bit Addressing	1

30.6.2 I2C Interrupt Mask Register (I2CIMR)

The 7-bit memory-mapped I2C interrupt mask register is used by the device to enable/disable the interrupts. Figure 30-14 and Table 30-6 describe this register.

Figure 30-14. I2C Interrupt Mask Register (I2CIMR) [offset = 04h]

15	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
Reserved	AASEN	SCDEN	TXRDYEN	RXRDYRN	ARDYEN	NACKEN	ALEN
R-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

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

Table 30-6. I2C Interrupt Mask Register (I2CIMR) Field Descriptions

Bit	Field	Value	Description
15-7	Reserved	0	Reads return 0. Writes have no effect.
6	AASEN		Address As Slave Interrupt Enable
		0	The AASEN interrupt is disabled.
		1	The AASEN interrupt is enabled.
5	SCDEN		Stop Condition Interrupt Enable
		0	The SCDEN interrupt is disabled.
		1	The SCDEN interrupt is enabled.
4	TXRDYEN		Transmit Data Ready Interrupt Enable
		0	The TXRDYEN interrupt is disabled.
		1	The TXRDYEN interrupt is enabled.
3	RXRDYEN		Receive Data Ready Interrupt Enable
		0	The RXRDYEN interrupt is disabled.
		1	The RXRDYEN interrupt is enabled.
2	ARDYEN		Register Access Ready Interrupt Enable
		0	The ARDYEN interrupt is disabled.
		1	The ARDYEN interrupt is enabled.
1	NACKEN		No Acknowledgement Interrupt Enable
		0	The NACKEN interrupt is disabled.
		1	The NACKEN interrupt is enabled.
0	ALEN		Arbitration Lost Interrupt Enable
		0	The ALEN interrupt is disabled.
		1	The ALEN interrupt is enabled.

30.6.3 I2C Status Register (I2CSTR)

Figure 30-15 and Table 30-7 describe this register.

Figure 30-15. I2C Status Register (I2CSR) [offset = 08h]

15	14	13	12	11	10	9	8
Reserved	SDIR	NACKSNT	BB	RSFULL	XSMT	AAS	AD0
R-0	R/W1C-0	R/W1C-0	R-0	R-0	R/W-1	R-0	R-0

7	6	5	4	3	2	1	0
Reserved		SCD	TXRDY	RXRDY	ARDY	NACK	AL
R-0		R/W1C-0	R/W-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; -*n* = value after reset

Table 30-7. I2C Status Register (I2CSTR) Field Descriptions

Bit	Field	Value	Description
15	Reserved	0	Reads return 0. Writes have no effect.
14	SDIR		Slave direction Setting this bit to 1 indicates that the I2C slave is a transmitter. Clearing this bit to 0 indicates that the I2C is a master transmitter/receiver or a slave receiver. This bit is also cleared by the STOP or START conditions. In DLB mode (in which the configuration should be master-transmitter slave-receiver), this bit is cleared to 0. **Writing a 1 to this bit will clear it.**
		0	The I2C is a master transmitter/receiver or a slave receiver.
		1	The I2C is a slave transmitter.
13	NACKSNT		No acknowledge sent This bit is set to 1 to indicate that a no acknowledgement (NACK) has been sent because the NACKMOD bit was set to 1. **Writing a 1 to this bit will clear it.**
		0	A NACK has not been sent.
		1	A NACK was sent because the NACKMOD was set to 1.
12	BB		Bus busy This bit indicates the state of the serial bus. On reception of a START condition or if the I2C detects a low state on I2CSCL, the device sets BB = 1. If the nIRS is set to 1 during transaction between other I2C devices, the BB bit is set at the first falling edge of SCL or START condition. BB is cleared to 0 after the reception of a STOP condition. BB is kept to 0 regardless of the SCL state when the I2C is in reset (nIRS = 0).
		0	The bus is free.
		1	The bus is busy.
11	RSFULL		Receiver shift full This bit is set to 1 to indicate that the receiver has experienced overrun. Overrun occurs when the receive shift register is full and I2CDRR has not been read since the receive shift register to I2CDRR transfer. The contents of I2CDRR are not lost. The I2C core logic is holding for I2CDRR read access. This bit is also set when, in master-repeat-mode, the I2C receives a byte of data. There is no difference between RXRDY and RSFULL in this case. The I2C master will not continue the transfer as long as the received data is in the I2CDRR or receive shift register. **RSFULL is cleared when reading the I2CDRR, resetting the I2C (nIRS = 0), or resetting the device.**
		0	No overrun has occurred.
		1	An overrun has occurred.

Table 30-7. I2C Status Register (I2CSTR) Field Descriptions (continued)

Bit	Field	Value	Description
10	XSMT		Transmit shift empty
			This bit is cleared to 0 to indicate that the transmitter has experienced underflow. Underflow occurs when the transmit shift register is empty and I2CDXR has not been loaded since the last I2CDXR to transmit shift register transfer. The I2C core logic is waiting for I2CDXR write access.
			XSMT is set to 1 as a result of writing to I2CDXR, by resetting the I2C block (nIRS = 0), or by resetting the device.
			In repeat mode, if the I2C in master transmitter mode is holding transfer with $\overline{\text{XSMT}}$ = 0 (that is, waiting for further action) and the STT or STP bit is set, XSMT is set to 1 by hardware.
		0	An underflow has occurred.
		1	No underflow has occurred.
9	AAS		Address as slave
			This bit cannot be cleared by writing a 1 to the bit or by reading the I2CIVR register.
		0	This bit is cleared by a STOP condition or detection of any address byte that does not match I2COAR.
		1	This bit is set to 1 by the device when it has recognized its own slave address or an address of all zeros (general call).
8	AD0		Address zero status
		0	A START or STOP condition was detected. No general call was detected.
		1	An address of all zeros (general call) was detected.
7-6	Reserved	0	Reads return 0. Writes have no effect.
5	SCD		Stop condition detect interrupt flag
			This bit is set to 1 when the I2C receives or sends a STOP condition.
			This bit is cleared to 0 by writing a 1 to this bit or reading the value 0x0006 from I2CIVR. Writing a 1 to this bit will clear the value 0x0006 from I2CIVR.
		0	No STOP condition has been sent or received.
		1	A STOP condition has been sent or received.
4	TXRDY		Transmit data ready interrupt flag
			This bit is set to 1 to indicate when data in the transmit data register, I2CDXR, has been copied into the transmit shift register. This bit can also be polled by the device to indicate when to write the next transmitted data into the I2CDXR. **Writing a 1 to this bit will set it.**
			This bit is cleared to 0 and code 0x0005 in I2CIVR is cleared when the I2CDXR is written. This bit cannot be cleared by reading the I2CIVR register.
		0	I2CDXR contains data to transmit.
		1	I2CDXR is empty.
3	RXRDY		Receive data ready interrupt flag
			This bit is set to 1 to indicate when the data in the receive shift register has been copied into the data receive register (I2CDRR). This bit can also be polled by the device to indicate when to read the received data in the I2CDRR.
			Writing a 1 to this bit or reading from I2CDRR will clear this bit, and will also clear code 0x0004 from I2CIVR. This bit cannot be cleared by reading the I2CIVR register.
		0	The I2CDRR has been read.
		1	The received data has been written into the I2CDRR.

Table 30-7. I2C Status Register (I2CSTR) Field Descriptions (continued)

Bit	Field	Value	Description
2	ARDY		Register access ready interrupt flag This bit is set to 1 when the previously programmed address, data and command has been performed and the status bit has been updated. The flag is used by the device to indicate that the I2C registers are ready to be accessed again. **This bit is automatically cleared by hardware when writing data to I2CDXR in transmit mode, reading data from I2CDRR in receive mode, or setting the STT or STP bit. Writing a 1 to this bit will clear this bit. This bit cannot be cleared by reading the I2CIVR register.** When RM = 0, ARDY is set when I2CCNT is passed 0 if STP register bit has not been set. When RM = 1, ARDY is set at each byte end. When FDF = 0, ARDY is asserted after the ACK for the slave address. When FDF = 1, there is no slave address. Therefore, ARDY is asserted after sending the start condition.
		0	*Nonrepeat mode, (RM = 0):* I2C registers are not ready to be accessed. *Repeat mode (RM = 1):* I2C registers are not ready to be accessed.
		1	*Nonrepeat mode, (RM = 0):* ICCNT passes 0 (if STP bit has not been set). *Repeat mode (RM = 1)*: The end of each byte was transmitted from I2CDXR.
1	NACK		No acknowledgement interrupt This bit is set to 1 when the master I2C does not receive an acknowledgement from the receiver. This bit is set only when the I2C has received a no-acknowledge in master mode. This bit is not set by no-acknowledgement after Start byte. In master start byte mode, the first byte (address of all zeroes) receives a NACK but does not clear the stop bit. **Writing a 1 to this bit or reading the value 0x0002 from I2CIVR will clear this bit.**
		0	An acknowledge was detected.
		1	No acknowledge was detected or the I2C is operating in the general call, even though an acknowledgement was received. This value clears the STP bit.
0	AL		Arbitration lost interrupt flag This bit is set to 1 when arbitration has been lost. **Writing a 1 to this bit or reading the value 0x0001 from I2CIVR will clear this bit.**
		0	No loss of arbitration has been detected.
		1	The device in the master transmitter mode senses it has lost an arbitration. This occurs when two or more transmitters start a transmission almost simultaneously or when the I2C attempts to start a transfer while BB=1. When this is set to 1 due to arbitration lost, the device becomes a slave receiver and the MST, STT, and STP bits in I2CMDR are cleared to 0.

30.6.4 I2C Clock Divider Low Register (I2CCKL)

The I2C clock divider low register is a 16-bit memory-mapped register used to divide the master clock down to obtain the I2C serial clock low time. Figure 30-16 and Table 30-8 describe this register.

Figure 30-16. I2C Clock Divider Low Register (I2CCKL) [offset = 0Ch]

15	0
CLKL	
R/W-0	

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

Table 30-8. I2C Clock Divider Low Register (I2CCKL) Field Descriptions

Bit	Field	Value	Description
15-0	CLKL	0-FFFFh	Low time clock division factor Used to divide down the module clock to create the low time portion of the master clock signal that will appear on the SCL pin.: $$LowTime = \left(\frac{I2CCLKL + d}{ModuleClockFrequency} \right) \qquad (66)$$ where *d* is the value that depends on the I2CPSC (see Section 30.1.3). This register must be configured while the I2C is still in reset (nIRS = 0).

30.6.5 I2C Clock Control High Register (I2CCKH)

The I2C clock divider high register is a 16-bit memory-mapped register used to divide the master clock down to obtain the I2C serial clock high time. Figure 30-17 and Table 30-9 describe this register.

Figure 30-17. I2C Clock Control High Register (I2CCKH) [offset = 10h]

15	0
CLKH	
R/W-0	

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

Table 30-9. I2C Clock Control High Register (I2CCKH) Field Descriptions

Bit	Field	Value	Description
15-0	CLKH	0-FFFFh	High time clock division factor Used to divide down the module clock to create the high time portion of the master clock signal that will appear on the SCL pin: $$HighTime = \left(\frac{I2CCLKH + d}{ModuleClockFrequency} \right) \qquad (67)$$ where *d* is the value that depends on the I2CPSC (see Section 30.1.3). This register must be configured while the I2C is still in reset (nIRS = 0).

30.6.6 I2C Data Count Register (I2CCNT)

The I2C data count register is a 16-bit memory-mapped register used to count received or transmitted data bytes. This register is also used to generate the STOP condition that terminates the transfer after the counter reaches zero. Figure 30-18 and Table 30-10 describe this register.

Figure 30-18. I2C Data Count Register (I2CCNT) [offset = 14h]

15 — 0
CNT
R/W-0

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

Table 30-10. I2C Data Count Register (I2CCNT) Field Descriptions

Bit	Field	Value	Description
15-0	CNT	0-FFFFh	Data counter This down counter is used to generate a stop condition, if a stop condition is specified (STP = 1). **Note: ICCNT is a don’t care when RM is set to 1.**
		0	The data counter is 65536.
		1	The data counter is 1.

30.6.7 I2C Data Receive Register (I2CDRR)

The I2C data receive register is a 16-bit memory-mapped register used by the device to read the received data. Figure 30-19 and Table 30-11 describe this register.

Figure 30-19. I2C Data Receive Register (I2CDRR) [offset = 18h]

15 — 8	7 — 0
Reserved	DATARX
R-0	R-0

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

Table 30-11. I2C Data Receive Register (I2CDRR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	DATARX	0-FFh	Receive data A read from this register clears the RXRDY bit and clears code 4h from the I2CIVR register.

30.6.8 I2C Slave Address Register (I2CSAR)

The I2C slave address register is a 16-bit memory-mapped register used to specify the address of the slave device to communicate to on the I2C bus. Figure 30-20 and Table 30-12 describe this register.

Figure 30-20. I2C Slave Address Register (I2CSAR) [offset = 1Ch]

15 10	9 0
Reserved	SA
R-0	R/W-3FFh

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

Table 30-12. I2C Slave Address Register (I2CSAR) Field Descriptions

Bit	Field	Value	Description
15-10	Reserved	0	Reads return 0. Writes have no effect.
9-0	SA		7- or 10-bit programmable slave address In either mode, all 10-bits are readable and writable. Bits 7, 8, and 9 should only be used in 10-bit address mode. Table 30-13 lists the correct mode for each bit.

Table 30-13. Correct Mode for SA Bits

Bits Used	Mode	Value of XA
SA(6–0)	7-bit addressing	0
SA(9–0)	10-bit addressing	1

30.6.9 I2C Data Transmit Register (I2CDXR)

Figure 30-21 and Table 30-14 describe this register.

Figure 30-21. I2C Data Transmit Register (I2CDXR) [offset = 20h]

15 8	7 0
Reserved	DATATX
R-0	R/W-0

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

Table 30-14. I2C Data Transmit Register (I2CDXR) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	DATATX	0-FFh	Transmit data Data written to this register will be transmitted on the I2C bus. A write to this register clears the TXRDY bit and clears code 0x05 from the I2CIVR register.

30.6.10 I2C Mode Register (I2CMDR)

Figure 30-22 and Table 30-15 describe this register.

Figure 30-22. I2C Mode Register (I2CMDR) [offset = 24h]

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	nIRS	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 30-15. I2C Mode Register (I2CMDR) Field Descriptions

Bit	Field	Value	Description
15	NACKMOD		No-acknowledge (NACK) mode
			This bit is used to send an acknowledge (ACK) or a no-acknowledge (NACK) to the transmitter. This bit is only applicable when the I2C is in receiver mode. In master receiver mode, when the internal data counter decrements to zero, the I2C sends a NACK. The master receiver I2C finishes a transfer when it sends a NACK. The I2C ignores ICCNT when NACKMOD is 1. The NACKMOD bit should be set before the rising edge of the last data bit if a NACK must be sent, and this bit is cleared once a NACK has been sent.
		0	The I2C sends an ACK to the transmitter during the acknowledge cycle.
		1	The I2C sends a NACK to the transmitter during the acknowledge cycle.
14	FREE		Free running bit
			This bit is used to determine the state of the I2C when a breakpoint is encountered in the high level language (HLL) debugger.
		0	The I2C stops immediately if SCL is low and keeps driving SCL low if the I2C master is a transmitter/receiver. If SCL is high, I2C waits until SCL becomes low and then stops. If the I2C is a slave, it will stop when the transmission/reception completes.
		1	The I2C runs free.
13	STT		Start condition
			The start condition bit works with the STP bit (master only mode). The STT and STP bits are configured to generate different transfer formats (see Table 30-16). The STT and STP bits can be used to terminate the repeat mode. This bit takes one I2C module clock cycle to set.
		0	The STT is reset to 0 by the hardware after the START condition has been generated.
		1	STT is set to 1 by the device to generate a START condition. In master mode, setting STT to 1 generates a START condition.
12	Reserved	0	Reads return 0. Writes have no effect.
11	STP		Stop condition
			1(Master mode only) This bit can be set to a 1 by the CPU to generate a stop condition. It is reset to 0 by the hardware after the stop condition has been generated. The stop condition is generated when ICCNT passes 0 when the I2C is in non-repeat mode (RM=0). In repeat mode (RM=1), the stop condition is generated if STP bit is 1. In transmitter mode, I2CTXRDY needs to be 1 (that is, you have to set STP bit unless you write data into I2CDXR).
		0	STP is reset to 0 by the hardware after the STOP condition has been generated.
		1	STP is set to 1 by the device to generate a STOP condition.
10	MST		Master/slave mode bit
			This bit determines whether the module will operate in master or slave mode; see Table 30-17. This bit is cleared after generating a STOP condition. The BB bit is cleared first, and MST bit is cleared second. Before starting the next transaction in master mode, this bit must be confirmed to be cleared.
		0	The module is in the slave mode and the clock is received from the master device.
		1	The module is in the master mode and it generates the clock. This bit is cleared when the transfer has completed.

Table 30-15. I2C Mode Register (I2CMDR) Field Descriptions (continued)

Bit	Field	Value	Description
9	TRX		Transmit/receive bit This bit determines the direction of data transmission of the I2C module. See Table 30-17.
		0	The module is in the receive mode and data on the SDA line is shifted into the data register I2CDRR.
		1	The module is in the transmit mode and the data in the I2CDXR is shifted out on the SDA line.
8	XA		Expand address enable bit This bit controls the addressing mode. When XA is set to 1, the I2C does not support the combined format in master mode operations. However, the I2C will acknowledge and support the formats when configured as a slave. This bit needs to be configured even if the I2C is in slave mode.
		0	The mode is set to 7-bit addressing mode (normal address mode).
		1	The mode is set to 10-bit addressing mode (expanded address mode).
7	RM		Repeat mode enable bit (Master mode only) This bit is a ‘don’t care’ if the module is configured in slave mode (MST = 0); see Table 30-16. Each time a byte of data is received, the user should decide whether or not to continue receiving more data. See Figure 30-23 for a diagram of this function.
		0	The mode is not in repeat mode.
		1	In repeat mode, data is continuously transmitted out of the ICDXR or received into the ICDRR until the STP bit is set to 1 regardless of ICCNT value. See Table 30-16 for module conditions.
6	DLB		Digital loop back enable bit This bit disables or enables the digital loopback mode of the I2C. This bit only applies in Master transmitter mode.
		0	Digital loop back mode is disabled.
		1	Digital loop back mode is enabled. In digital loop back mode, data transmitted out of the I2CDXR will be received in the I2CDRR. The address of the I2COAR is output on SDA.
5	nIRS		I2C reset enable bit When set to 0, this bit will place all status registers in this module to their default state. Resetting nIRS during a data transfer can hang the I2C bus.
		0	I2C is in reset.
		1	I2C is out of reset.
4	STB		Start byte mode enable bit (Master mode only) The Start byte mode bit is set to 1 by the CPU to configure the I2C in Start byte mode. The I2C sends “00000001” regardless of the I2CSAR value. Refer to the Philips I2C specification for more details.
		0	The module is not in START byte mode.
		1	The module is in START byte mode.
3	FDF		Free data format enable bit This bit configures the module to operate in free data format mode (see Table 30-17) in both master and slave modes. When FDF is 0, ARDY is asserted after ACK for the slave address. When FDF is 1, there is no slave address. Therefore, ARDY is asserted after sending the start condition. FDF mode is not supported in digital loop back mode.
		0	The module is not in free data format mode.
		1	The module is in free data format mode.
2-0	BC		Bit count This bit defines the number of bits starting from the LSB (excluding the acknowledge bit) that are sent on the bus when data is written to the data transmit register. If the bits BC0, BC1, and BC2 are all 0, then the number of bits sent on the bus is 8. If the bit count bits are a non-zero value, then the number of bits sent on the bus is that value. The value “001” is reserved. When performing a transfer using the bit count of, for example, n (where n is nonzero), only the n least significant bits in the data receive register are valid and correct. The rest of the bits should be disregarded. See Table 30-18 for more information.

Figure 30-23. Typical Timing Diagram of Repeat Mode

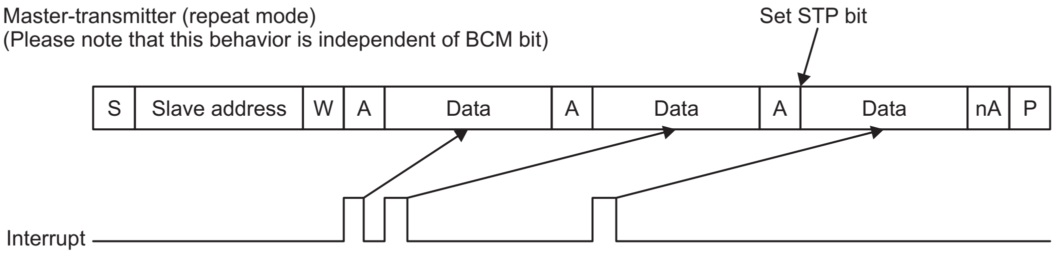

Master-transmitter (repeat mode)
(Please note that this behavior is independent of BCM bit)
Set STP bit
S
Slave address
W
A
Data
A
Data
A
Data
nA
P
Interrupt

Table 30-16. I2C Module Condition, Bus Activity, and Mode

RM	STT	STP	Condition	Bus Activities [1]	Mode
0	0	0	Idle	None	N/A
0	0	1	Stop	P	N/A
0	1	0	(Repeat) Start	S-A-D..(n)..D	Repeat n
0	1	1	(Repeat) Start-Stop	S-A-D..(n)..D-P	Repeat n
1	0	0	Idle	none	N/A
1	0	1	Stop	P	N/A
1	1	0	(Repeat) Start	S-A-D-D-D-....	Continuous
1	1	1	Reserved	none	N/A

[1] P = Stop condition; S = Start condition; A = Acknowledge bit; D = data

Table 30-17. I2C Module Operating Modes

FDF	MST	TRX	Operating Mode
0	0	x	Slave in non-FDF mode
0	1	0	Master receive in non-FDF mode
0	1	1	Master transmit in non-FDF mode
1	0	0	Slave receive in FDF mode
1	0	1	Slave transmit in FDF mode
1	1	0	Master receive in FDF mode
1	1	1	Master transmit in FDF mode

Table 30-18. Number of Bits Sent on Bus

BC2	BC1	BC0	Bits in FDF	Bits with ACK
0	0	0	8	9
0	0	1	NA (reserved)	NA (reserved)
0	1	0	2	3
0	1	1	3	4
1	0	0	4	5
1	0	1	5	6
1	1	0	6	7
1	1	1	7	8

30.6.11 I2C Interrupt Vector Register (I2CIVR)

The I2C interrupt vector register is a 16-bit memory-mapped register used to indicate the occurrence of an interrupt. Figure 30-24 and Table 30-19 describe this register.

Figure 30-24. I2C Interrupt Vector Register (I2CIVR) [offset = 28h]

15 12	11 8	7 3	2 0
Reserved	TESTMD	Reserved	INTCODE
R-0	R/W-0	R-0	R/WC-0

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

Table 30-19. I2C Interrupt Vector Register (I2CIVR) Field Descriptions

Bit	Field	Value	Description
15-12	Reserved	0	Reads return 0. Writes have no effect.
11-8	TESTMD	0-3h	Reserved for internal testing.
7-3	Reserved	0	Reads return 0. Writes have no effect.
2-0	INTCODE	0-3h	Interrupt Code Bits This binary coded interrupt vector indicates which interrupt has occurred. If there is more than one interrupt pending, reading I2CIVR provides the vector for the highest-priority interrupt that is pending. Reading the I2CIVR will clear the corresponding flags in I2CSTR for AL, NACK and SCD as long as those interrupts are enabled. A new interrupt will be generated for each pending source. Reading I2CIVR will clear the INTCODE for AL, NACK, SCD, AAS, RXRDY and TXRDY. Reading I2CIVR **will not** clear the INTCODE for ARDY. The INTCODE for certain codes can also be cleared by either writing a 1 to the corresponding interrupt flag bits in I2CSTR, or by reading and writing to the receive or transmit registers. See Section 30.6.3 for more details. Users must read (clear) the I2CIVR before doing another start otherwise the I2CIVR could contain incorrect (old interrupt flag) value.

Table 30-20. Interrupt Codes for INTCODE Bits

Code	INTCODE(2-0)	Interrupt Occurred
00h	000	none
01h	001 (highest priority)	Arbitration lost (AL)
02h	010	No acknowledgement (NACK)
03h	011	Register access ready (ARDY)
04h	100	Receive data ready (RXRDY)
05h	101	Transmit data ready (TXRDY)
06h	110	Stop condition detect (SCD)
07h	111 (lowest priority)	Address as slave (AAS)

30.6.12 I2C Extended Mode Register (I2CEMDR)

The I2C extended mode register is a 16-bit memory-mapped register that contains additional mode select bits. Figure 30-25 and Table 30-21 describe this register.

Figure 30-25. I2C Extended Mode Register (I2CEMDR) [offset = 2Ch]

15 2	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 30-21. I2C Extended Mode Register (I2CEMDR) Field Descriptions

Bit	Field	Value	Description
15-2	Reserved	0	Reads return 0. Writes have no effect.
1	IGNACK		Ignore NACK mode
		0	The master transmitter will operate normally, discontinue the data transfer, and set the ARDY and NACK status bits when a NACK signal is received from the slave.
		1	The master transmitter will ignore a NACK received from the slave.
0	BCM		Backwards compatibility mode When set to 1, the I2C is compatible with previous versions of the I2C. This means the TXRDY interrupt is generated in slave-transmit mode when TXRDY is set and the I2C needs more data to transmit. This behavior causes an extra TXRDY interrupt to be generated because the I2C recognizes the end of transfer after generating an interrupt for the next byte of data. When BCM is 0, the TXRDY interrupt in slave-transmit mode is generated when XSMT = 1. In this case, the I2C generates an interrupt for the next byte after receiving the ACK from previous data. The setting of this bit only applies to slave transmit mode.
		0	The I2C is **not** in compatibility mode.
		1	The I2C is in compatibility mode.

30.6.13 I2C Prescale Register (I2CPSC)

The I2C prescaler register is a 16-bit memory-mapped register used for dividing down the VBUS_CLK to obtain a module clock frequency between 6.7 MHz and 13.3 MHz. Figure 30-26 and Table 30-22 describe this register.

Figure 30-26. I2C Prescale Register (I2CPSC) [offset = 30h]

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

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

Table 30-22. I2C Prescale Register (I2CPSC) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	PSC	0-FFh	Prescale 8-bit prescaler to divide down the VBUS clock to obtain the I2C module clock. This register must be initialized while the I2C is still in reset (nIRS = 0). The value takes effect on the rising edge of nIRS. See Section 30.1.3 for more information.

30.6.14 I2C Peripheral ID Register 1 (I2CPID1)

Figure 30-27 and Table 30-23 describe this register.

Figure 30-27. I2C Peripheral ID Register 1 (I2CPID1) [offset = 34h]

15 8	7 0
CLASS	REVISION
R-1	R-46h

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

Table 30-23. I2C Peripheral ID Register 1 (I2CPID1) Field Descriptions

Bit	Field	Value	Description
15-8	CLASS	0-FFh	Peripheral class These bits identify the class of peripheral.
7-0	REVISION	0-FFh	Revision level of the I2C These bits identify the revision level of the I2C.

30.6.15 I2C Peripheral ID Register 2 (I2CPID2)

Figure 30-28 and Table 30-24 describe this register.

Figure 30-28. I2C Peripheral ID Register 2 (I2CPID2) [offset = 38h]

15 8	7 0
Reserved	TYPE
R-0	R-5h

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

Table 30-24. I2C Peripheral ID Register 2 (I2CPID2) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	TYPE	0-FFh	Peripheral type These bits identify the type of peripheral.

30.6.16 I2C DMA Control Register (I2CDMACR)

This register contains the transmit and receive DMA enable bits. Figure 30-29 and Table 30-25 describe this register.

Figure 30-29. I2C DMA Control Register (I2CDMACR) [offset = 3Ch]

15 … 2	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 30-25. I2C DMA Control Register (I2CDMACR) Field Descriptions

Bit	Field	Value	Description
15-2	Reserved	0	Reads return 0. Writes have no effect.
1	TXDMAEN		Transmitter DMA enable
			This bit controls the transmit DMA event pin to the system. When this bit is a 1, the DMA transmit event is enabled and the DMA can occur. When this bit is a 0, the DMA transmit event is disabled.
			Writing a 1 to this bit will send a TXDMA request to the DMA module if PINFUNC is also set to 0.
		0	The transmit DMA is disabled.
		1	The transmit DMA is enabled.
0	RXDMAEN		Receive DMA enable
			This bit controls the receive DMA event pin to the system. When this bit is 1, the DMA receive event is enabled and the DMA can occur. When this bit is a 0, the DMA receive event is disabled.
		0	The receive DMA is disabled.
		1	The receive DMA is enabled.

30.6.17 I2C Pin Function Register (I2CPFNC)

Figure 30-30 and Table 30-26 describe this register.

Figure 30-30. I2C Pin Function Register (I2CPFNC) [offset = 48h]

15 … 1	0
Reserved	PINFUNC
R-0	R/W-0

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

Table 30-26. I2C Pin Function Register (I2CPFNC) Field Descriptions

Bit	Field	Value	Description
15-1	Reserved	0	Reads return 0. Writes have no effect.
0	PINFUNC		SDA and SCL pin function
			This bit controls whether the SDA and SCL pins function as I2C pins or as I/O pins.
		0	SDA and SCL pins function as I2C pins.
		1	SDA and SCL pins function as I/O pins.

30.6.18 I2C Pin Direction Register (I2CPDIR)

This register is used to independently configure each I2C pin, when configured as a general-purpose I/O, as either an input or output. Figure 30-31 and Table 30-27 describe this register.

Figure 30-31. I2C Pin Direction Register (I2CPDIR) [offset = 4Ch]

15 2	1	0
Reserved	SDADIR	SCLDIR
R-0	R/W-0	R/W-0

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

Table 30-27. I2C Pin Direction Register (I2CPDIR) Field Descriptions

Bit	Field	Value	Description
15-2	Reserved	0	Reads return 0. Writes have no effect.
1	SDADIR		SDA direction This bit controls the direction of the I2C SDA pin when configured as a GPIO.
		0	SDA pin functions as an input.
		1	SDA pin functions as an output.
0	SCLDIR		SCL direction This bit controls the direction of the I2C SCL pin when configured as a GPIO.
		0	SCL pin functions as an input.
		1	SCL pin functions as an output.

30.6.19 I2C Data Input Register (I2CDIN)

Figure 30-32 and Table 30-28 describe this register.

Figure 30-32. I2C Data Input Register (I2CDIN) [offset = 50h]

15 2	1	0
Reserved	SDAIN	SCLIN
R-0	R-x	R-x

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

Table 30-28. I2C Data Input Register (I2CDIN) Field Descriptions

Bit	Field	Value	Description
15-2	Reserved	0	Reads return 0. Writes have no effect.
1	SDAIN		Serial data in The value of this bit reflects the value on the SDA pin.
0	SCLIN		Serial clock data in The value of this bit reflects the value on the SCL pin.

30.6.20 I2C Data Output Register (I2CDOUT)

This register contains the values sent to the I2C pins. Figure 30-33 and Table 30-29 describe this register.

Figure 30-33. I2C Data Output Register (I2CDOUT) [offset 0x54]

15 - 2	1	0
Reserved	SDAOUT	SCLOUT
R-0	R/W-0	R/W-0

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

Table 30-29. I2C Data Output Register (I2CDOUT) Field Descriptions

Bit	Field	Value	Description
15-2	Reserved	0	Reads return 0. Writes have no effect.
1	SDAOUT		SDA data output This function is only active if the SDA pin is configured as an I/O pin with PINFUNC = 1. This bit contains the value sent to the SDA pin.
		0	SDA pin is driven low.
		1	SDA pin is driven high.
0	SCLOUT		SCL data output This function is only active if the SCL pin is configured as an I/O pin with PINFUNC = 1. This bit contains the value sent to the SCL pin.
		0	SCL pin is driven low.
		1	SCL pin is driven high.

30.6.21 I2C Data Set Register (I2CDSET)

The I2CDSET register is an alias of the I2CDOUT register. Figure 30-34 and Table 30-30 describe this register.

Figure 30-34. I2C Data Set Register (I2CDSET) [offset = 58h]

15 - 2	1	0
Reserved	SDASET	SCLSET
R-0	R/W-0	R/W-0

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

Table 30-30. I2C Data Set Register (I2CDSET) Field Description

Bit	Field	Value	Description
15-2	Reserved	0	Reads return 0. Writes have no effect.
1	SDASET	0	Serial data set This bit is used to set the SDA GPIO pin. Read: Reads return value of SDAOUT. Write: Writing a 0 to this bit has no effect.
		1	Read: Reads return value of SDAOUT. Write: SDAOUT is set to logic high (1).
0	SCLSET	0	Serial clock set This bit is used to set the SCL GPIO pin. Read: Reads return value of SCLOUT. Write: Writing a 0 to this bit has no effect.
		1	Read: Reads return value of SCLOUT. Write: SCLOUT is set to logic high (1).

30.6.22 I2C Data Clear Register (I2CDCLR)

The I2CDCLR register is an alias of the I2CDOUT register. Figure 30-35 and Table 30-31 describe this register.

Figure 30-35. I2C Data Clear Register (I2CDCLR) [offset = 5Ch]

15	2	1	0
Reserved		SDACLR	SCLCLR
R-0		R/W-0	R/W-0

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

Table 30-31. I2C Data Clear Register (I2CDSET) Field Descriptions

Bit	Field	Value	Description
15-2	Reserved	0	Reads return 0. Writes have no effect.
1	SDACLR		Serial data clear This bit is used to clear the SDA GPIO pin.
		0	Read: Reads return value of SDAOUT. Write: Writing a 0 to this bit has no effect.
		1	Read: Reads return value of SDAOUT. Write: SDAOUT is cleared to logic low (0).
0	SCLCLR		Serial clock clear This bit is used to clear the SCL GPIO pin.
		0	Read: Reads return value of SCLOUT. Write: Writing a 0 to this bit has no effect.
		1	Read: Reads return value of SCLOUT. Write: SCLOUT is cleared to logic low (0).

30.6.23 I2C Pin Open Drain Register (I2CPDR)

Figure 30-36 and Table 30-32 describe this register.

Figure 30-36. I2C Pin Open Drain Register (I2CPDR) [offset = 60h]

15	2	1	0
Reserved		SDAPDR	SCLPDR
R-0		R/W-0	R/W-0

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

Table 30-32. I2C Pin Open Drain Register (I2CPDR) Field Descriptions

Bit	Field	Value	Description
15-2	Reserved	0	Reads return 0. Writes have no effect.
1	SDAPDR		SDA pin open drain enable
		0	The open drain function is enabled (the output voltage is V_{OL} or lower if SDAOUT = 0 and high-impedance if SDAOUT = 1).
		1	The open drain function is disabled (output voltage is V_{OL} or lower if SDAOUT = 0; V_{OH} or higher if SDAOUT = 1).
0	SCLPDR		SCL pin open drain enable
		0	The open drain function is enabled (the output voltage is V_{OL} or lower if SCLOUT = 0 and high-impedance if SCLOUT = 1).
		1	The open drain function is disabled (output voltage is V_{OL} or lower if SCLOUT = 0; V_{OH} or higher if SCLOUT = 1).

30.6.24 I2C Pull Disable Register (I2CPDIS)

Values in the I2CPDIS register enable or disable the pull control capability of the pins. Figure 30-37 and Table 30-33 describe this register.

Figure 30-37. I2C Pull Disable Register (I2CPDIS) [offset = 64h]

15	2	1	0
Reserved		SDAPDIS	SCLPDIS
R-0		R/W-1	R/W-1

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

Table 30-33. I2C Pull Disable Register (I2CPDIS) Field Descriptions

Bit	Field	Value	Description
15-2	Reserved	0	Reads return 0. Writes have no effect.
1	SDAPDIS		SDA pull disable
		0	The pull function is enabled.
		1	The pull function is disabled.
0	SCLPDIS		SCL pull disable
		0	The pull function is enabled.
		1	The pull function is disabled.

30.6.25 I2C Pull Select Register (I2CPSEL)

Values in the I2CPSEL register select the pull up or pull down functions of the corresponding pins. Figure 30-38 and Table 30-34 describe this register.

Figure 30-38. I2C Pull Select Register (I2CPSEL) [offset = 68h]

15	2	1	0
Reserved		SDAPSEL	SCLPSEL
R-0		R/W-0	R/W-0

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

Table 30-34. I2C Pull Select Register (I2CPSEL) Field Descriptions

Bit	Field	Value	Description
15-2	Reserved	0	Reads return 0. Writes have no effect.
1	SDAPSEL		SDA pull select
		0	The pull down function is enabled.
		1	The pull up function is enabled.
0	SCLPSEL		SCL pull select
		0	The pull down function is enabled.
		1	The pull up function is enabled.

30.6.25.1 Summary

The behavior of the input buffer, output buffer, and the pull control is summarized in Table 30-35.

Table 30-35. Input Buffer, Output Buffer, and Pull Control Behavior as GPIO Pins

Device under Reset?	Pin Direction (DIR) [1][2]	Pull Disable (PULDIS) [1][3]	Pull Select (PULSEL) [1][4]	Pull Control	Output Buffer	Input Buffer
Yes	X	X	X	Enabled	Disabled	Enabled
No	0	0	0	Pull down	Disabled	Enabled
No	0	0	1	Pull up	Disabled	Enabled
No	0	1	0	Disabled	Disabled	Enabled
No	0	1	1	Disabled	Disabled	Enabled
No	1	X	X	Disabled	Enabled	Enabled

[1] X = Don’t care

[2] DIR = 0 for input, = 1 for output

[3] PULDIS = 0 for enabling pull control
= 1 for disabling pull control

[4] PULSEL= 0 for pull-down functionality
= 1 for pull-up functionality

30.6.26 I2C Pins Slew Rate Select Register (I2CSRS)

This register controls the slew rate of the signal on the I2C pins. Figure 30-39 and Table 30-36 describe this register.

Figure 30-39. I2C Pins Slew Rate Select Register (I2CSRS) [offset = 6Ch]

15 … 2	1	0
Reserved	SDASRS	SCLSRS
R-0	R/W-0	R/W-0

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

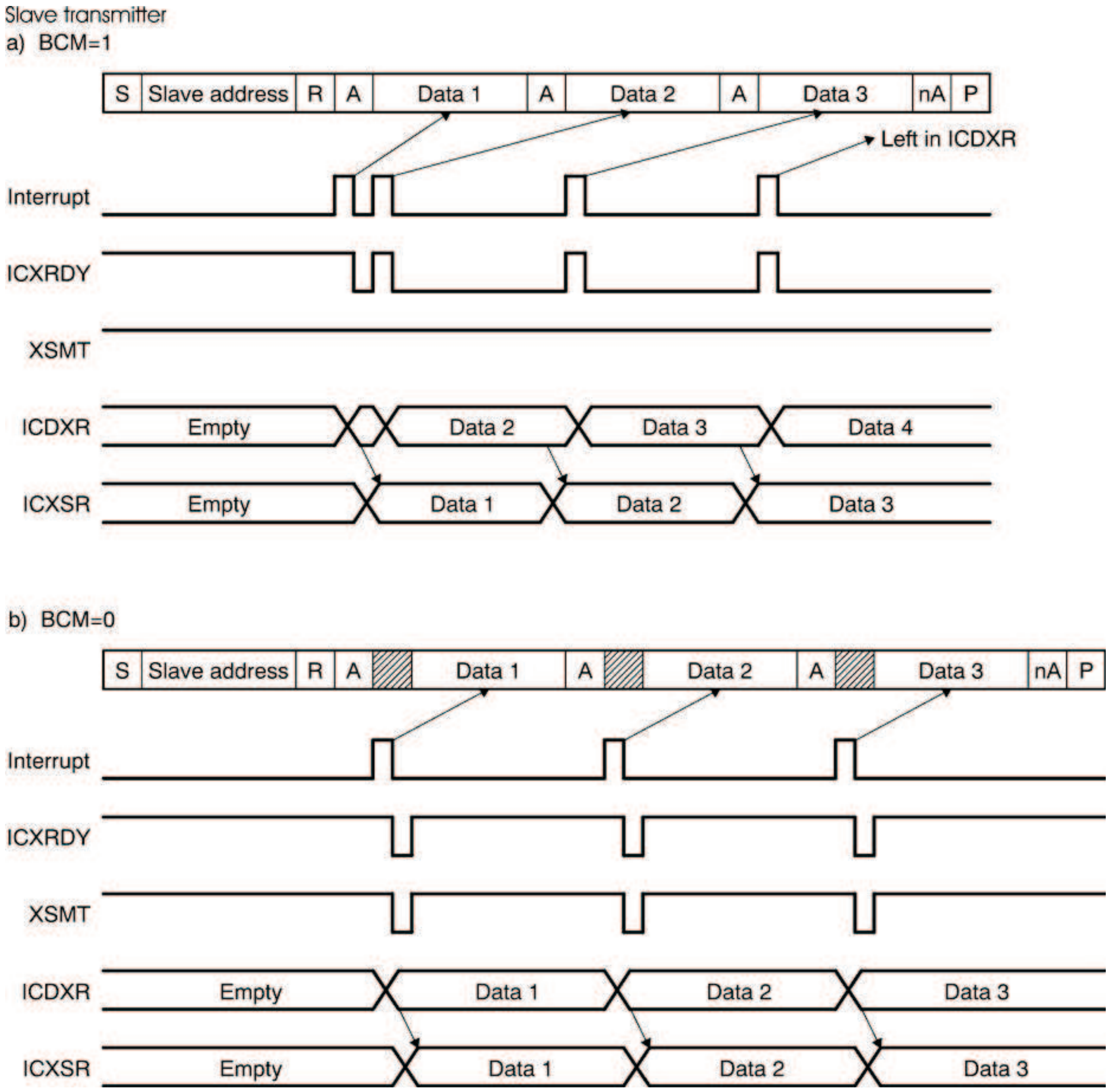
Table 30-36. I2C Pins Slew Rate Select Register (I2CSRS) Field Descriptions

Bit	Field	Value	Description
15-2	Reserved	0	Reads return 0. Writes have no effect.
1	SDASRS		SDA slew rate select
		0	The slow buffer is selected.
		1	The normal buffer is selected.
0	SCLSRS		SCL slew rate select
		0	The slow buffer is selected.
		1	The normal buffer is selected.

30.7 Sample Waveforms

Figure 30-40 provides waveforms to illustrate the difference between normal operation and backward compatibility mode.

Figure 30-40. Difference between Normal Operation and Backward Compatibility Mode



Chapter 31
SPNU514C–March 2018

EMAC/MDIO Module

This chapter describes the Ethernet Media Access Controller (EMAC) and physical layer (PHY) device Management Data Input/Output (MDIO) module.

Topic **Page**

31.1 Introduction 1694
31.2 Architecture 1696
31.3 EMAC Control Module Registers.. 1743
31.4 MDIO Registers.. 1756
31.5 EMAC Module Registers 1769

31.1 Introduction

This document provides a functional description of the Ethernet Media Access Controller (EMAC) and physical layer (PHY) device Management Data Input/Output (MDIO) module integrated in the microcontroller. Included are the features of the EMAC and MDIO modules, a discussion of their architecture and operation, how these modules connect to the outside world, and a description of the registers for each module.

The EMAC controls the flow of packet data from the system to the PHY. The MDIO module controls PHY configuration and status monitoring.

Both the EMAC and the MDIO modules interface to the system core through a custom interface that allows efficient data transmission and reception. This custom interface is referred to as the EMAC control module and is considered integral to the EMAC/MDIO peripheral.

31.1.1 Purpose of the Peripheral

The EMAC module is used to move data between the device and another host connected to the same network, in compliance with the Ethernet protocol.

31.1.2 Features

The EMAC/MDIO has the following features:

- Synchronous 10/100 Mbps operation.
- Standard Media Independent Interface (MII) and/or Reduced Media Independent Interface (RMII) to physical layer device (PHY).
- EMAC acts as DMA master to either internal or external device memory space.
- Eight receive channels with VLAN tag discrimination for receive quality-of-service (QOS) support.
- Eight transmit channels with round-robin or fixed priority for transmit quality-of-service (QOS) support.
- Ether-Stats and 802.3-Stats statistics gathering.
- Transmit CRC generation selectable on a per channel basis.
- Broadcast frames selection for reception on a single channel.
- Multicast frames selection for reception on a single channel.
- Promiscuous receive mode frames selection for reception on a single channel (all frames, all good frames, short frames, error frames).
- Hardware flow control.
- 8k-byte local EMAC descriptor memory that allows the peripheral to operate on descriptors without affecting the CPU. The descriptor memory holds enough information to transfer up to 512 Ethernet packets without CPU intervention. (This memory is also known as CPPI RAM.)
- Programmable interrupt logic permits the software driver to restrict the generation of back-to-back interrupts, which allows more work to be performed in a single call to the interrupt service routine.

31.1.3 Functional Block Diagram

Figure 31-1 shows the three main functional modules of the EMAC/MDIO peripheral:

- EMAC control module
- EMAC module
- MDIO module

The EMAC control module is the main interface between the device core processor to the EMAC and MDIO modules. The EMAC control module controls device interrupts and incorporates an 8k-byte internal RAM to hold EMAC buffer descriptors (also known as CPPI RAM).

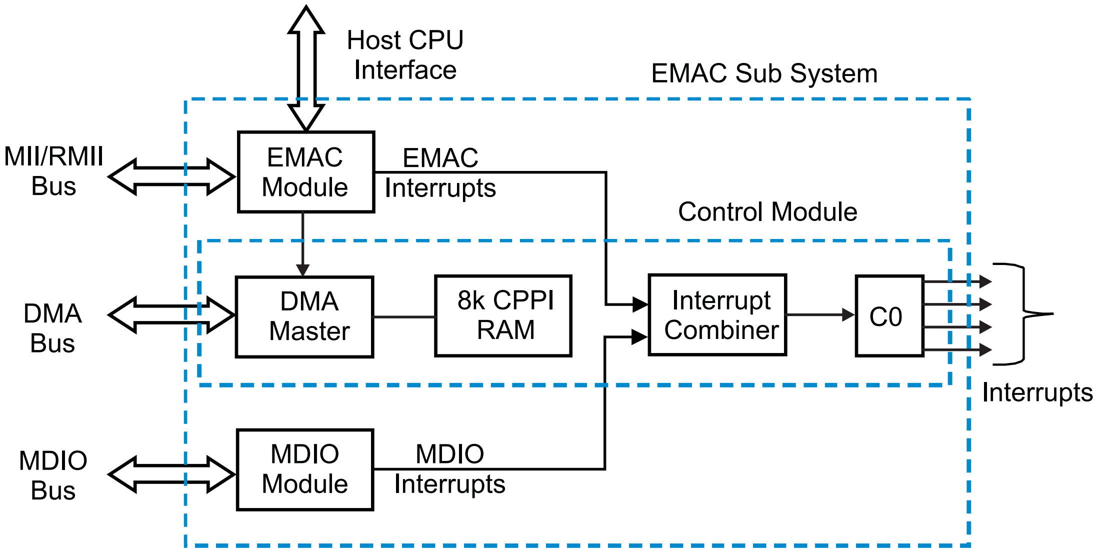
The MDIO module implements the 802.3 serial management interface to interrogate and control up to 32 Ethernet PHYs connected to the device by using a shared two-wire bus. Host software uses the MDIO module to configure the autonegotiation parameters of each PHY attached to the EMAC, retrieve the negotiation results, and configure required parameters in the EMAC module for correct operation. The module is designed to allow almost transparent operation of the MDIO interface, with very little maintenance from the core processor.

The EMAC module provides an efficient interface between the processor and the network. The EMAC on this device supports 10Base-T (10 Mbits/sec) and 100BaseTX (100 Mbits/sec), half-duplex and full-duplex mode, and hardware flow control and quality-of-service (QOS) support.

Figure 31-1 shows the main interface between the EMAC control module and the CPU. The following connections are made to the device core:

- The DMA bus connection from the EMAC control module allows the EMAC module to read and write both internal and external memory through the DMA memory transfer controller.
- The EMAC control, EMAC, and MDIO modules all have control registers. These registers are memory-mapped into device memory space. Along with these registers, the control module's internal CPPI RAM is mapped into this same range.
- The EMAC and MDIO interrupts are combined into four interrupt signals within the control module. The Vectored Interrupt Manager (VIM) receives all four interrupt signals from the combiner and submits these interrupt requests to the CPU.

Figure 31-1. EMAC and MDIO Block Diagram



31.1.4 Industry Standard(s) Compliance Statement

The EMAC peripheral conforms to the IEEE 802.3 standard, describing the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer specifications. The IEEE 802.3 standard has also been adopted by ISO/IEC and re-designated as ISO/IEC 8802-3:2000(E).

However, the EMAC deviates from the standard in the way it handles transmit underflow errors. The EMAC MII interface does not use the Transmit Coding Error signal MTXER. Instead of driving the error pin when an underflow condition occurs on a transmitted frame, the EMAC intentionally generates an incorrect checksum by inverting the frame CRC, so that the transmitted frame is detected as an error by the network.

31.2 Architecture

This section discusses the architecture and basic function of the EMAC peripheral.

31.2.1 Clock Control

All internal EMAC logic is clocked synchronously on the VCLKA4 domain. Please refer to the *Architecture* chapter for more details.

The MDIO clock is based on a divide-down of the VCLK3 peripheral bus clock and is specified to run up to 2.5 MHz (although typical operation would be 1.0 MHz). Because the VCLK3 peripheral clock frequency is configurable, the application software or driver must control the divide-down value.

The transmit and receive clock sources are provided by the external PHY to the MII_TXCLK and MII_RXCLK pins or to the RMII reference clock pin. Data is transmitted and received with respect to the reference clocks of the interface pins.

The MII interface frequencies for the transmit and receive clocks are fixed by the IEEE 802.3 specification as:

- 2.5 MHz at 10 Mbps
- 25 MHz at 100 Mbps

The RMII interface frequency for the transmit and receive clocks are fixed at 50 MHz for both 10 Mbps and 100 Mbps.

31.2.2 Memory Map

The EMAC peripheral includes internal memory that is used to hold buffer descriptions of the Ethernet packets to be received and transmitted. This internal RAM is 2K × 32 bits in size. Data can be written to and read from the EMAC internal memory by either the EMAC or the CPU. It is used to store buffer descriptors that are 4-words (16-bytes) deep. This 8K local memory holds enough information to transfer up to 512 Ethernet packets without CPU intervention. This EMAC RAM is also referred to as the CPPI buffer descriptor memory because it complies with the Communications Port Programming Interface (CPPI) v3.0 standard.

The packet buffer descriptors can also be placed in other on- and off-chip memories such as the CPU RAM. There are some tradeoffs in terms of interconnect bandwidth when descriptors are placed in the CPU RAM, versus when they are placed in the EMAC’s internal memory. In general, the EMAC throughput is better when the descriptors are placed in the local EMAC CPPI RAM.

31.2.3 *Signal Descriptions*

The microcontrollers support both the MII and the RMII interfaces. Only one of these two interfaces can be used at a time. A separate control register in the I/O Multiplexing Module (IOMM) allows the application to indicate the actual interface being used. This is the bit 24 of the PINMMR29 control register. This bit is set by default and selects the RMII interface. The application can select the MII interface by clearing this bit. Please refer to the *I/O Multiplexing and Control Module (IOMM)* chapter for more details on the procedure to configure the PINMMR registers.

Each of the EMAC and MDIO signals for the MII and RMII interfaces are multiplexed with other I/O functions on this microcontroller. Please refer to Section 31.2.4 for information on configuration of the multiplexing control registers to enable the MII / RMII connections to these I/Os.

31.2.3.1 Media Independent Interface (MII) Connections

Figure 31-2 shows a device with integrated EMAC and MDIO interfaced via a MII connection in a typical system. The EMAC module does not include a transmit error (MTXER) pin. In the case of transmit error, CRC inversion is used to negate the validity of the transmitted frame.

The individual EMAC and MDIO signals for the MII interface are summarized in Table 31-1. For more information, refer to either the IEEE 802.3 standard or ISO/IEC 8802-3:2000(E).

Figure 31-2. Ethernet Configuration—MII Connections

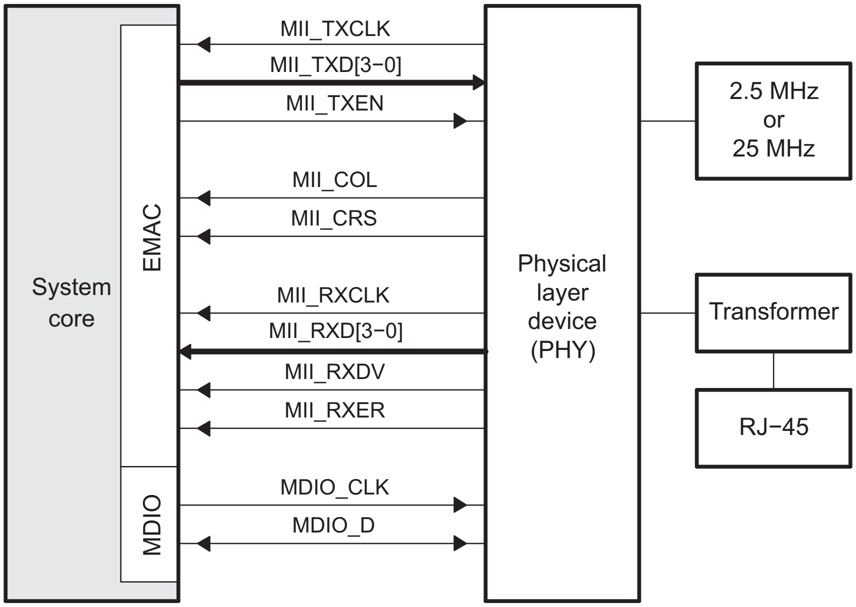


Table 31-1. EMAC and MDIO Signals for MII Interface

Signal	Type	Description
MII_TXCLK	I	Transmit clock (MII_TXCLK). The transmit clock is a continuous clock that provides the timing reference for transmit operations. The MII_TXD and MII_TXEN signals are tied to this clock. The clock is generated by the PHY and is 2.5 MHz at 10 Mbps operation and 25 MHz at 100 Mbps operation.
MII_TXD[3-0]	O	Transmit data (MII_TXD). The transmit data pins are a collection of 4 data signals comprising 4 bits of data. MTDX0 is the least-significant bit (LSB). The signals are synchronized by MII_TXCLK and valid only when MII_TXEN is asserted.
MII_TXEN	O	Transmit enable (MII_TXEN). The transmit enable signal indicates that the MII_TXD pins are generating nibble data for use by the PHY. It is driven synchronously to MII_TXCLK.
MII_COL	I	Collision detected (MII_COL). In half-duplex operation, the MII_COL pin is asserted by the PHY when it detects a collision on the network. It remains asserted while the collision condition persists. This signal is not necessarily synchronous to MII_TXCLK nor MII_RXCLK. In full-duplex operation, the MII_COL pin is used for hardware transmit flow control. Asserting the MII_COL pin will stop packet transmissions; packets in the process of being transmitted when MII_COL is asserted will complete transmission. The MII_COL pin should be held low if hardware transmit flow control is not used.
MII_CRS	I	Carrier sense (MII_CRS). In half-duplex operation, the MII_CRS pin is asserted by the PHY when the network is not idle in either transmit or receive. The pin is deasserted when both transmit and receive are idle. This signal is not necessarily synchronous to MII_TXCLK nor MII_RXCLK. In full-duplex operation, the MII_CRS pin should be held low.
MII_RXCLK	I	Receive clock (MII_RXCLK). The receive clock is a continuous clock that provides the timing reference for receive operations. The MII_RXD, MII_RXDV, and MII_RXER signals are tied to this clock. The clock is generated by the PHY and is 2.5 MHz at 10 Mbps operation and 25 MHz at 100 Mbps operation.
MII_RXD[3-0]	I	Receive data (MII_RXD). The receive data pins are a collection of 4 data signals comprising 4 bits of data. MRDX0 is the least-significant bit (LSB). The signals are synchronized by MII_RXCLK and valid only when MII_RXDV is asserted.
MII_RXDV	I	Receive data valid (MII_RXDV). The receive data valid signal indicates that the MII_RXD pins are generating nibble data for use by the EMAC. It is driven synchronously to MII_RXCLK.
MII_RXER	I	Receive error (MII_RXER). The receive error signal is asserted for one or more MII_RXCLK periods to indicate that an error was detected in the received frame. This is meaningful only during data reception when MII_RXDV is active.
MDIO_CLK	O	Management data clock (MDIO_CLK). The MDIO data clock is sourced by the MDIO module on the system. It is used to synchronize MDIO data access operations done on the MDIO pin. The frequency of this clock is controlled by the CLKDIV bits in the MDIO control register (CONTROL).
MDIO_D	I/O	Management data input output (MDIO_D). The MDIO data pin drives PHY management data into and out of the PHY by way of an access frame consisting of start of frame, read/write indication, PHY address, register address, and data bit cycles. The MDIO_D pin acts as an output for all but the data bit cycles at which time it is an input for read operations.

31.2.3.2 Reduced Media Independent Interface (RMII) Connections

Figure 31-3 shows a device with integrated EMAC and MDIO interfaced via a RMII connection in a typical system.

The individual EMAC and MDIO signals for the RMII interface are summarized in Table 31-2. For more information, refer to either the IEEE 802.3 standard or ISO/IEC 8802-3:2000(E).

Figure 31-3. Ethernet Configuration—RMII Connections

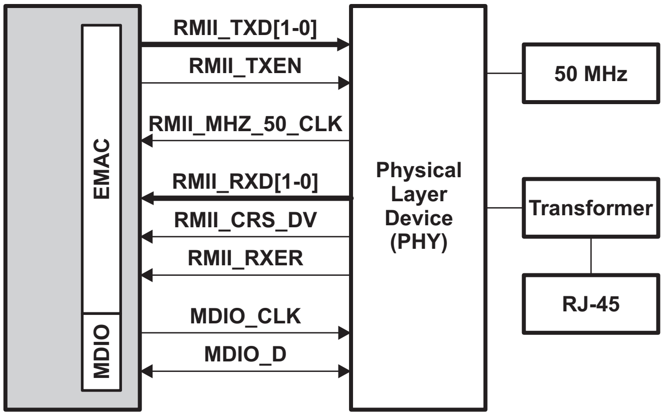


Table 31-2. EMAC and MDIO Signals for RMII Interface

Signal	Type	Description
RMII_TXD[1-0]	O	Transmit data (RMII_TXD). The transmit data pins are a collection of 2 bits of data. RMTDX0 is the least-significant bit (LSB). The signals are synchronized by RMII_MHZ_50_CLK and valid only when RMII_TXEN is asserted.
RMII_TXEN	O	Transmit enable (RMII_TXEN). The transmit enable signal indicates that the RMII_TXD pins are generating data for use by the PHY. RMII_TXEN is synchronous to RMII_MHZ_50_CLK.
RMII_MHZ_50_CLK	I	RMII reference clock (RMII_MHZ_50_CLK). The reference clock is used to synchronize all RMII signals. RMII_MHZ_50_CLK must be continuous and fixed at 50 MHz.
RMII_RXD[1-0]	I	Receive data (RMII_RXD). The receive data pins are a collection of 2 bits of data. RMRDX0 is the least-significant bit (LSB). The signals are synchronized by RMII_MHZ_50_CLK and valid only when RMII_CRS_DV is asserted and RMII_RXER is deasserted.
RMII_CRS_DV	I	Carrier sense/receive data valid (RMII_CRS_DV). Multiplexed signal between carrier sense and receive data valid.
RMII_RXER	I	Receive error (RMII_RXER). The receive error signal is asserted to indicate that an error was detected in the received frame.
MDIO_CLK	O	Management data clock (MDIO_CLK). The MDIO data clock is sourced by the MDIO module on the system. It is used to synchronize MDIO data access operations done on the MDIO pin. The frequency of this clock is controlled by the CLKDIV bits in the MDIO control register (CONTROL).
MDIO_D	I/O	Management data input output (MDIO_D). The MDIO data pin drives PHY management data into and out of the PHY by way of an access frame consisting of start of frame, read/write indication, PHY address, register address, and data bit cycles. The MDIO_D pin acts as an output for all but the data bit cycles at which time it is an input for read operations.

31.2.4 MII / RMII Signal Multiplexing Control

Each of the MII and RMII interface signals are multiplexed with other functions on this microcontroller. The application must configure the control registers in the I/O multiplexing module in order to enable the MII/RMII functionality on the corresponding I/Os. Table 31-3 shows the byte to be configured to enable the MDIO functions. Table 31-4 shows the byte to be configured to enable the MII or RMII functions. Please refer to the *I/O Multiplexing and Control Module (IOMM)* chapter for more details on the procedure to configure the PINMMR registers.

Table 31-3. MDIO Multiplexing Control

MDIO Signal Name	Control for Selecting EMAC / MDIO Signal
MDIO_CLK	PINMMR7 [15:8] = 0b00000100
MDIO_D	PINMMR8 [15:8] = 0b00000100

Table 31-4. MII / RMII Multiplexing Control

MII / RMII Signal Name	Control for Selecting MII Signal	Control for Selecting RMII Signal
MII_TXCLK	PINMMR17 [7:0] = 0b00000010	-
MII_TXD[3]	PINMMR14 [7:0] = 0b00000100	-
MII_TXD[2]	PINMMR13 [31:24] = 0b00000100	-
MII_TXD[1] / RMII_TXD[1]	PINMMR13 [15:8] = 0b00000100	PINMMR13 [15:8] = 0b00001000
MII_TXD[0] / RMII_TXD[0]	PINMMR13 [7:0] = 0b00000100	PINMMR13 [7:0] = 0b00001000
MII_TXEN / RMII_TXEN	PINMMR13 [23:16] = 0b00000100	PINMMR13 [23:16] = 0b00001000
MII_COL	PINMMR20 [23:16] = 0b00000100	-
MII_CRS / RMII_CRSDV	PINMMR17 [23:16] = 0b00000010	PINMMR17 [23:16] = 0b00000100
MII_RXCLK / RMII_50MHZ_CLK	PINMMR14 [15:8] = 0b00000010	PINMMR14 [15:8] = 0b00000100
MII_RXD[3]	PINMMR12 [31:24] = 0b00000100	-
MII_RXD[2]	PINMMR12 [23:16] = 0b00000100	-
MII_RXD[1] / RMII_RXD[1]	PINMMR12 [7:0] = 0b00000010	PINMMR12 [7:0] = 0b00000100
MII_RXD[0] / RMII_RXD[0]	PINMMR11 [31:24] = 0b00000100	PINMMR11 [31:24] = 0b00001000
MII_RXDV	PINMMR19 [15:8] = 0b00000010	-
MII_RXER / RMII_RXER	PINMMR10 [7:0] = 0b00000010	PINMMR10 [7:0] = 0b00000100

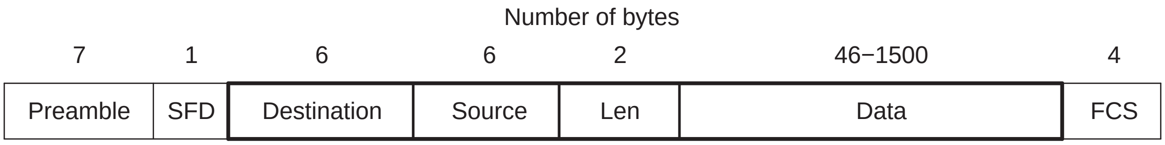
31.2.5 Ethernet Protocol Overview

A brief overview of the Ethernet protocol is given in the following subsections. See the IEEE 802.3 standard document for in-depth information on the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method.

31.2.5.1 Ethernet Frame Format

All the Ethernet technologies use the same frame structure. The format of an Ethernet frame is shown in Figure 31-4 and described in Table 31-5. The Ethernet packet, which is the collection of bytes representing the data portion of a single Ethernet frame on the wire, is shown outlined in bold. The Ethernet frames are of variable lengths, with no frame smaller than 64 bytes or larger than RXMAXLEN bytes (header, data, and CRC).

Figure 31-4. Ethernet Frame Format



Legend: SFD=Start Frame Delimeter; FCS=Frame Check Sequence (CRC)

Table 31-5. Ethernet Frame Description

Field	Bytes	Description
Preamble	7	Preamble. These 7 bytes have a fixed value of 55h and serve to wake up the receiving EMAC ports and to synchronize their clocks to that of the sender's clock.
SFD	1	Start of Frame Delimiter. This field with a value of 5Dh immediately follows the preamble pattern and indicates the start of important data.
Destination	6	Destination address. This field contains the Ethernet MAC address of the EMAC port for which the frame is intended. It may be an individual or multicast (including broadcast) address. When the destination EMAC port receives an Ethernet frame with a destination address that does not match any of its MAC physical addresses, and no promiscuous, multicast or broadcast channel is enabled, it discards the frame.
Source	6	Source address. This field contains the MAC address of the Ethernet port that transmits the frame to the Local Area Network.
Len	2	Length/Type field. The length field indicates the number of EMAC client data bytes contained in the subsequent data field of the frame. This field can also be used to identify the type of data the frame is carrying.
Data	46 to (RXMAXLEN - 18)	Data field. This field carries the datagram containing the upper layer protocol frame, that is, IP layer datagram. The maximum transfer unit (MTU) of Ethernet is (RXMAXLEN - 18) bytes. This means that if the upper layer protocol datagram exceeds (RXMAXLEN - 18) bytes, then the host has to fragment the datagram and send it in multiple Ethernet packets. The minimum size of the data field is 46 bytes. This means that if the upper layer datagram is less then 46 bytes, the data field has to be extended to 46 bytes by appending extra bits after the data field, but prior to calculating and appending the FCS.
FCS	4	Frame Check Sequence. A cyclic redundancy check (CRC) is used by the transmit and receive algorithms to generate a CRC value for the FCS field. The frame check sequence covers the 60 to 1514 bytes of the packet data. Note that this 4-byte field may or may not be included as part of the packet data, depending on how the EMAC is configured.

31.2.5.2 Ethernet's Multiple Access Protocol

Nodes in an Ethernet Local Area Network are interconnected by a broadcast channel -- when an EMAC port transmits a frame, all the adapters on the local network receive the frame. Carrier Sense Multiple Access with Collision Detection (CSMA/CD) algorithms are used when the EMAC operates in half-duplex mode. When operating in full-duplex mode, there is no contention for use of a shared medium because there are exactly two ports on the local network.

Each port runs the CSMA/CD protocol without explicit coordination with the other ports on the Ethernet network. Within a specific port, the CSMA/CD protocol works as follows:

1. The port obtains data from upper layer protocols at its node, prepares an Ethernet frame, and puts the frame in a buffer.
2. If the port senses that the medium is idle, it starts to transmit the frame. If the port senses that the transmission medium is busy, it waits until it no longer senses energy (plus an Inter-Packet Gap time) and then starts to transmit the frame.
3. While transmitting, the port monitors for the presence of signal energy coming from other ports. If the port transmits the entire frame without detecting signal energy from other Ethernet devices, the port is done with the frame.
4. If the port detects signal energy from other ports while transmitting, it stops transmitting its frame and instead transmits a 48-bit jam signal.
5. After transmitting the jam signal, the port enters an exponential backoff phase. If a data frame encounters back-to-back collisions, the port chooses a random value that is dependent on the number of collisions. The port then waits an amount of time that is a multiple of this random value and returns to step 2.

31.2.6 Programming Interface

31.2.6.1 Packet Buffer Descriptors

The buffer descriptor is a central part of the EMAC module and is how the application software describes Ethernet packets to be sent and empty buffers to be filled with incoming packet data. The basic descriptor format is shown in Figure 31-5 and described in Table 31-6.

For example, consider three packets to be transmitted: Packet A is a single fragment (60 bytes), Packet B is fragmented over three buffers (1514 bytes total), and Packet C is a single fragment (1514 bytes). The linked list of descriptors to describe these three packets is shown in Figure 31-6.

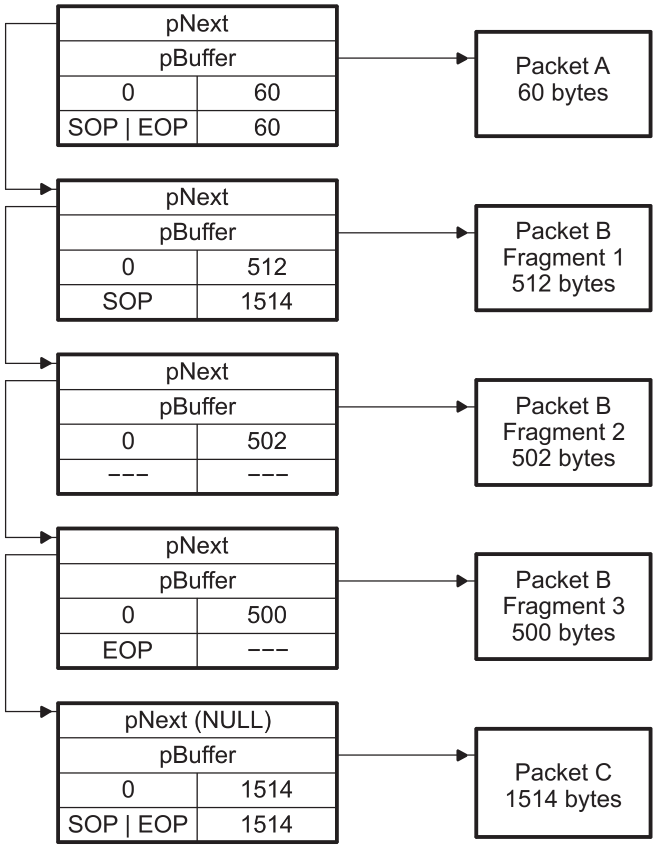
Figure 31-5. Basic Descriptor Format

	Bit Fields	
Word Offset	**31 16**	**15 0**
0	Next Descriptor Pointer	
1	Buffer Pointer	
2	Buffer Offset	Buffer Length
3	Flags	Packet Length

Table 31-6. Basic Descriptor Description

Word Offset	Field	Field Description
0	Next Descriptor Pointer	The next descriptor pointer is used to create a single linked list of descriptors. Each descriptor describes a packet or a packet fragment. When a descriptor points to a single buffer packet or the first fragment of a packet, the start of packet (SOP) flag is set in the flags field. When a descriptor points to a single buffer packet or the last fragment of a packet, the end of packet (EOP) flag is set. When a packet is fragmented, each fragment must have its own descriptor and appear sequentially in the descriptor linked list.
1	Buffer Pointer	The buffer pointer refers to the actual memory buffer that contains packet data during transmit operations, or is an empty buffer ready to receive packet data during receive operations.
2	Buffer Offset	The buffer offset is the offset from the start of the packet buffer to the first byte of valid data. This field only has meaning when the buffer descriptor points to a buffer that actually contains data.
	Buffer Length	The buffer length is the actual number of valid packet data bytes stored in the buffer. If the buffer is empty and waiting to receive data, this field represents the size of the empty buffer.
3	Flags	The flags field contains more information about the buffer, such as, is it the first fragment in a packet (SOP), the last fragment in a packet (EOP), or contains an entire contiguous Ethernet packet (both SOP and EOP). The flags are described in Section 31.2.6.4 and Section 31.2.6.5.
	Packet Length	The packet length only has meaning for buffers that both contain data and are the start of a new packet (SOP). In the case of SOP descriptors, the packet length field contains the length of the entire Ethernet packet, regardless if it is contained in a single buffer or fragmented over several buffers.

Figure 31-6. Typical Descriptor Linked List



31.2.6.2 Transmit and Receive Descriptor Queues

The EMAC module processes descriptors in linked lists as discussed in Section 31.2.6.1. The lists used by the EMAC are maintained by the application software through the use of the head descriptor pointer registers (HDP). The EMAC supports eight channels for transmit and eight channels for receive. The corresponding head descriptor pointers are:

- TX*n*HDP - Transmit Channel *n* DMA Head Descriptor Pointer Register
- RX*n*HDP - Receive Channel *n* DMA Head Descriptor Pointer Register

After an EMAC reset and before enabling the EMAC for send and receive, all 16 head descriptor pointer registers must be initialized to 0.

The EMAC uses a simple system to determine if a descriptor is currently owned by the EMAC or by the application software. There is a flag in the buffer descriptor flags called OWNER. When this flag is set, the packet that is referenced by the descriptor is considered to be owned by the EMAC. Note that ownership is done on a packet based granularity, not on descriptor granularity, so only SOP descriptors make use of the OWNER flag. As packets are processed, the EMAC patches the SOP descriptor of the corresponding packet and clears the OWNER flag. This is an indication that the EMAC has finished processing all descriptors up to and including the first with the EOP flag set, indicating the end of the packet (note this may only be one descriptor with both the SOP and EOP flags set).

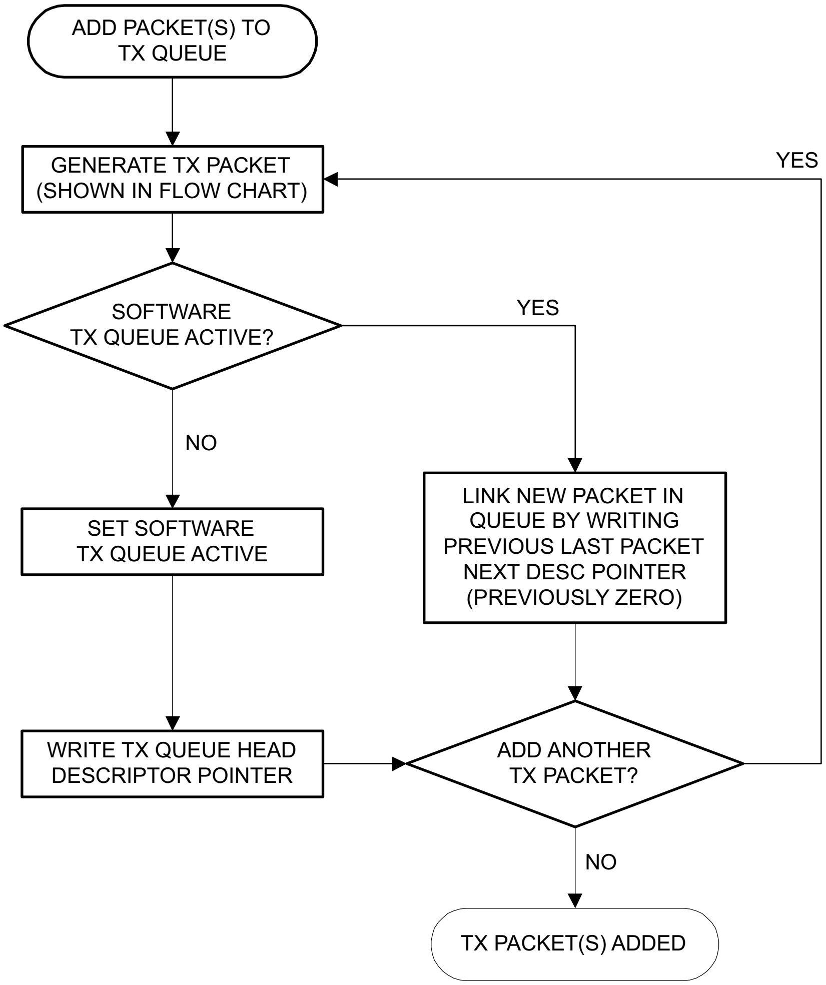
To add a descriptor or a linked list of descriptors to an EMAC descriptor queue for the first time, the software application simply writes the pointer to the descriptor or first descriptor of a list to the corresponding HDP register. Note that the last descriptor in the list must have its “next” pointer cleared to 0. This is the only way the EMAC has of detecting the end of the list. Therefore, in the case where only a single descriptor is added, its “next descriptor” pointer must be initialized to 0.

The HDP must never be written to while a list is active. To add additional descriptors to a descriptor list already owned by the EMAC, the NULL “next” pointer of the last descriptor of the previous list is patched with a pointer to the first descriptor of the new list. The list of new descriptors to be appended to the existing list must itself be NULL terminated before the pointer patch is performed.

There is a potential race condition where the EMAC may read the “next” pointer of a descriptor as NULL in the instant before an application appends additional descriptors to the list by patching the pointer. This case is handled by the software application always examining the buffer descriptor flags of all EOP packets, looking for a special flag called end of queue (EOQ). The EOQ flag is set by the EMAC on the last descriptor of a packet when the descriptor’s “next” pointer is NULL. This is the way the EMAC indicates to the software application that it believes it has reached the end of the list. When the software application sees the EOQ flag set, the application may at that time submit the new list, or the portion of the appended list that was missed by writing the new list pointer to the same HDP that started the process.

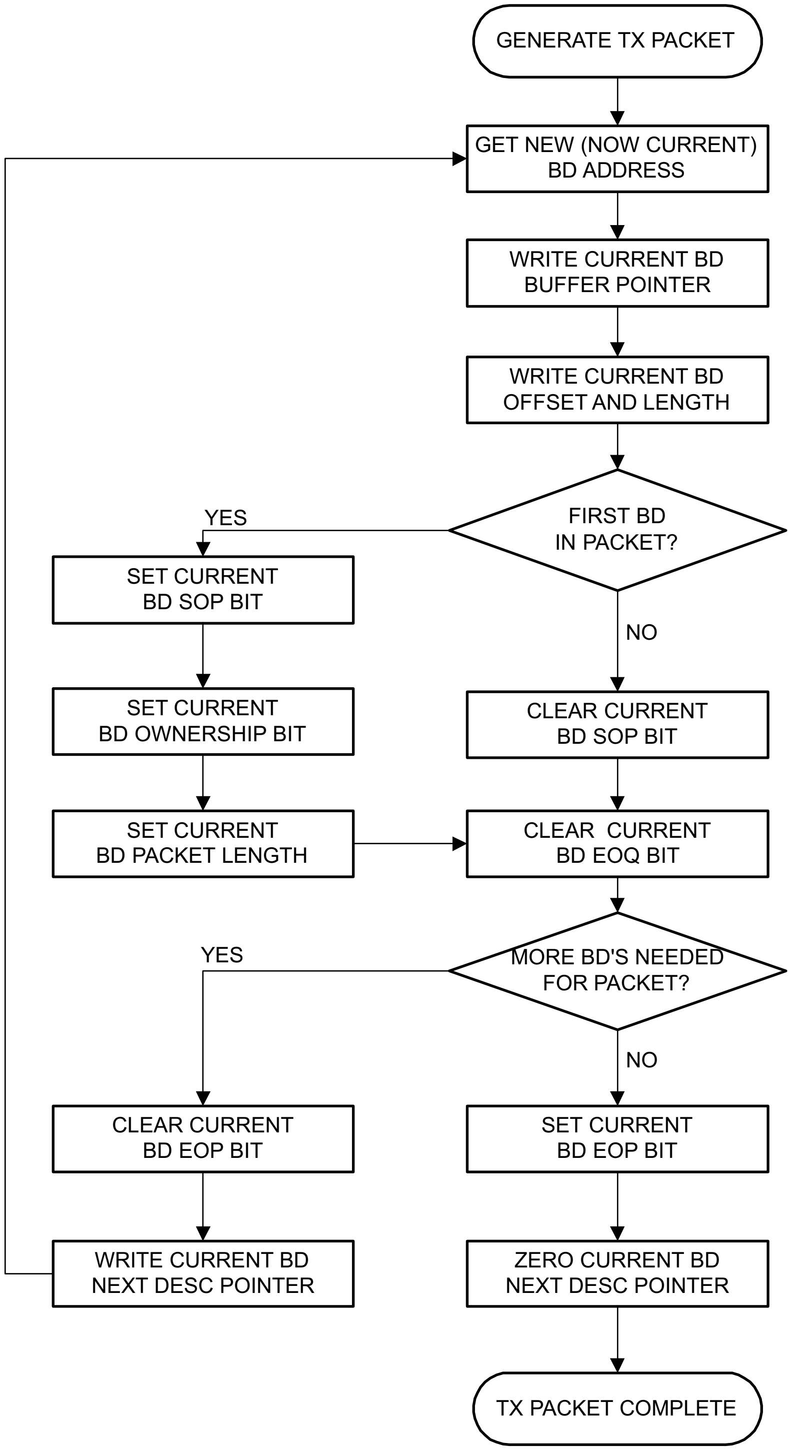
This process applies when adding packets to a transmit list, and empty buffers to a receive list. Figure 31-7, Figure 31-8, and Figure 31-9 illustrate transmit operations.

Figure 31-7. Transmit Packet Add Flow Chart


ADD PACKET(S) TO TX QUEUE
GENERATE TX PACKET (SHOWN IN FLOW CHART)
YES
SOFTWARE TX QUEUE ACTIVE?
YES
NO
SET SOFTWARE TX QUEUE ACTIVE
LINK NEW PACKET IN QUEUE BY WRITING PREVIOUS LAST PACKET NEXT DESC POINTER (PREVIOUSLY ZERO)
WRITE TX QUEUE HEAD DESCRIPTOR POINTER
ADD ANOTHER TX PACKET?
NO
TX PACKET(S) ADDED

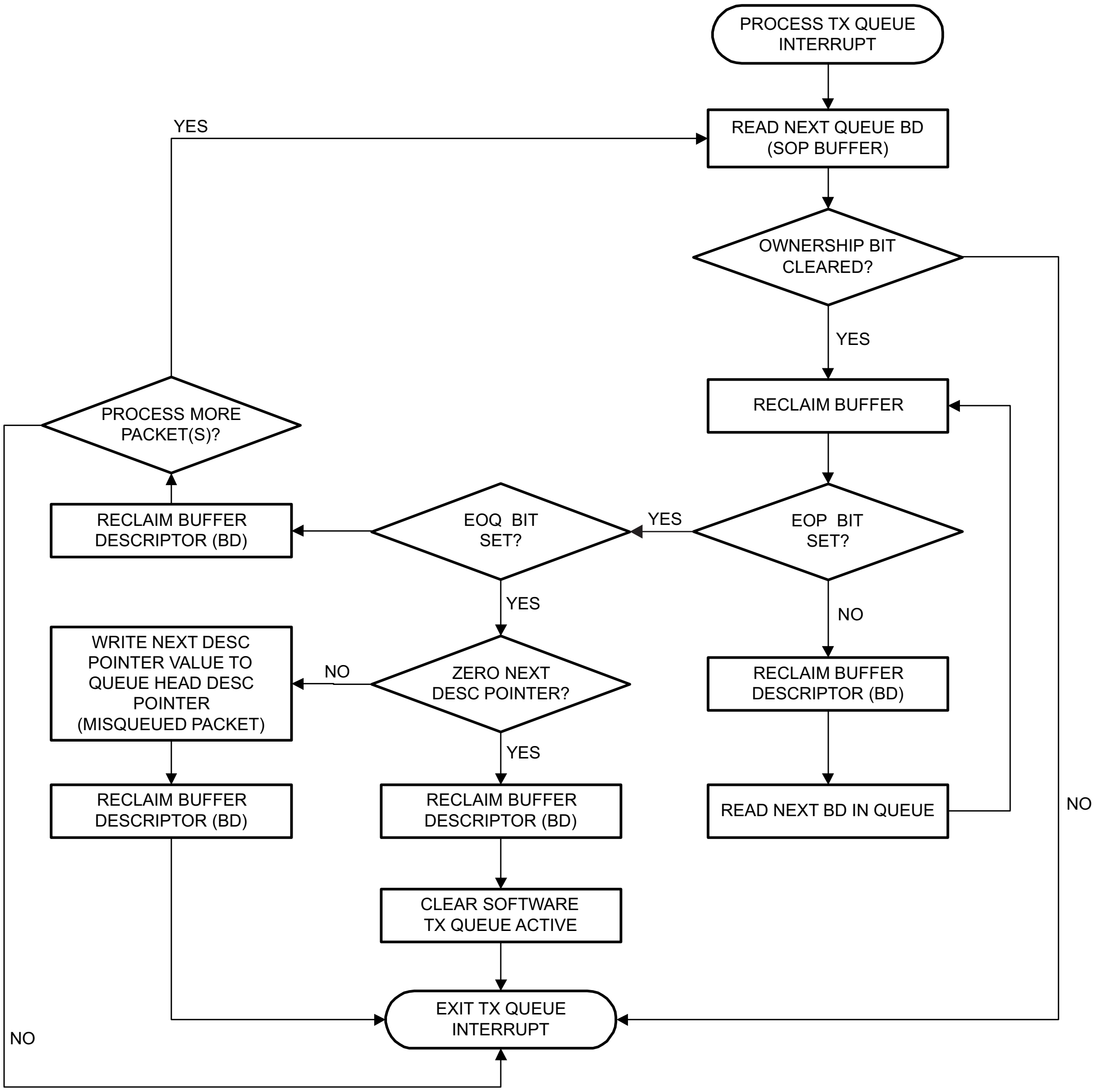
Note: Software TX QUEUE ACTIVE is an indication that at least one packet is in the TX queue (from the software viewpoint).

Figure 31-8. Generate Transmit Packet Flow Chart


GENERATE TX PACKET
GET NEW (NOW CURRENT) BD ADDRESS
WRITE CURRENT BD BUFFER POINTER
WRITE CURRENT BD OFFSET AND LENGTH
FIRST BD IN PACKET?
YES
SET CURRENT BD SOP BIT
SET CURRENT BD OWNERSHIP BIT
SET CURRENT BD PACKET LENGTH
NO
CLEAR CURRENT BD SOP BIT
CLEAR CURRENT BD EOQ BIT
MORE BD'S NEEDED FOR PACKET?
YES
CLEAR CURRENT BD EOP BIT
WRITE CURRENT BD NEXT DESC POINTER
NO
SET CURRENT BD EOP BIT
ZERO CURRENT BD NEXT DESC POINTER
TX PACKET COMPLETE

BD = Buffer Descriptor

Figure 31-9. Transmit Queue Interrupt Processing Flow Chart


PROCESS TX QUEUE INTERRUPT
READ NEXT QUEUE BD (SOP BUFFER)
YES
OWNERSHIP BIT CLEARED?
YES
RECLAIM BUFFER
PROCESS MORE PACKET(S)?
RECLAIM BUFFER DESCRIPTOR (BD)
EOQ BIT SET?
YES
EOP BIT SET?
YES
NO
WRITE NEXT DESC POINTER VALUE TO QUEUE HEAD DESC POINTER (MISQUEUED PACKET)
NO
ZERO NEXT DESC POINTER?
RECLAIM BUFFER DESCRIPTOR (BD)
YES
RECLAIM BUFFER DESCRIPTOR (BD)
RECLAIM BUFFER DESCRIPTOR (BD)
READ NEXT BD IN QUEUE
NO
CLEAR SOFTWARE TX QUEUE ACTIVE
EXIT TX QUEUE INTERRUPT
NO

BD = Buffer Descriptor

Note: Whether or not to process more than one packet is a software decision.

31.2.6.3 Transmit and Receive EMAC Interrupts

The EMAC processes descriptors in linked list chains as discussed in Section 31.2.6.1, using the linked list queue mechanism discussed in Section 31.2.6.2.

The EMAC synchronizes descriptor list processing through the use of interrupts to the software application. The interrupts are controlled by the application using the interrupt masks, global interrupt enable, and the completion pointer register (CP). The CP is also called the interrupt acknowledge register.

The EMAC supports eight channels for transmit and eight channels for receive. The corresponding completion pointer registers are:

- TX*n*CP - Transmit Channel *n* Completion Pointer (Interrupt Acknowledge) Register
- RX*n*CP - Receive Channel *n* Completion Pointer (Interrupt Acknowledge) Register

These registers serve two purposes. When read, they return the pointer to the last descriptor that the EMAC has processed. When written by the software application, the value represents the last descriptor processed by the software application. When these two values do not match, the interrupt is active.

Interrupts in the EMAC control module are routed to the Vectored Interrupt Manager (VIM) as four separate interrupt requests. The interrupt configuration determines whether or not an active interrupt request actually interrupts the CPU. In general the following settings are required for basic EMAC transmit and receive interrupts:

1. EMAC transmit and receive interrupts are enabled by setting the mask registers RXINTMASKSET and TXINTMASKSET
2. Global interrupts are set in the EMAC control module: C0RXEN and C0TXEN
3. The VIM is configured to accept C0_RX_PULSE and C0_TX_PULSE interrupts from the EMAC control module
4. The normal mode (IRQ) interrupts are enabled in the Cortex-R4F CPU

Whether or not the interrupt is enabled, the current state of the receive or transmit channel interrupt can be examined directly by the software application reading the EMAC receive interrupt status (unmasked) register (RXINTSTATRAW) and transmit interrupt status (unmasked) register (TXINTSTATRAW).

After servicing transmit or receive interrupts, the application software must acknowledge both the EMAC and EMAC control module interrupts.

EMAC interrupts are acknowledged when the application software updates the value of TX*n*CP or RX*n*CP with a value that matches the internal value kept by the EMAC. This mechanism ensures that the application software never misses an EMAC interrupt because the interrupt acknowledgment is tied directly to the buffer descriptor processing.

EMAC control module interrupts are acknowledged when the application software writes the appropriate C0TX or C0RX key to the EMAC End-Of-Interrupt Vector register (MACEOIVECTOR). The MACEOIVECTOR behaves as an interrupt pulse interlock -- once the EMAC control module has issued an interrupt pulse to the CPU, it will not generate further pulses of the same type until the original pulse has been acknowledged.

31.2.6.4 Transmit Buffer Descriptor Format

A transmit (TX) buffer descriptor (Figure 31-10) is a contiguous block of four 32-bit data words aligned on a 32-bit boundary that describes a packet or a packet fragment. Example 31-1 shows the transmit buffer descriptor described by a C structure.

Figure 31-10. Transmit Buffer Descriptor Format

Word 0

31 — 0
Next Descriptor Pointer

Word 1

31 — 0
Buffer Pointer

Word 2

31 — 16	15 — 0
Buffer Offset	Buffer Length

Word 3

31	30	29	28	27	26	25 — 16
SOP	EOP	OWNER	EOQ	TDOWNCMPLT	PASSCRC	Reserved

15 — 0
Packet Length

Example 31-1. Transmit Buffer Descriptor in C Structure Format

```
/*
// EMAC Descriptor
//
// The following is the format of a single buffer descriptor
// on the EMAC.
*/
typedef struct _EMAC_Desc {
 struct _EMAC_Desc *pNext; /* Pointer to next descriptor in chain */
 Uint8 *pBuffer; /* Pointer to data buffer */
 Uint32 BufOffLen; /* Buffer Offset(MSW) and Length(LSW) */
 Uint32 PktFlgLen; /* Packet Flags(MSW) and Length(LSW) */
} EMAC_Desc;

/* Packet Flags */
#define EMAC_DSC_FLAG_SOP 0x80000000u
#define EMAC_DSC_FLAG_EOP 0x40000000u
#define EMAC_DSC_FLAG_OWNER 0x20000000u
#define EMAC_DSC_FLAG_EOQ 0x10000000u
#define EMAC_DSC_FLAG_TDOWNCMPLT 0x08000000u
#define EMAC_DSC_FLAG_PASSCRC 0x04000000u
```

31.2.6.4.1 Next Descriptor Pointer

The next descriptor pointer points to the 32-bit word aligned memory address of the next buffer descriptor in the transmit queue. This pointer is used to create a linked list of buffer descriptors. If the value of this pointer is zero, then the current buffer is the last buffer in the queue. The software application must set this value prior to adding the descriptor to the active transmit list. This pointer is not altered by the EMAC.

The value of pNext should never be altered once the descriptor is in an active transmit queue, unless its current value is NULL. If the pNext pointer is initially NULL, and more packets need to be queued for transmit, the software application may alter this pointer to point to a newly appended descriptor. The EMAC will use the new pointer value and proceed to the next descriptor unless the pNext value has already been read. In this latter case, the transmitter will halt on the transmit channel in question, and the software application may restart it at that time. The software can detect this case by checking for an end of queue (EOQ) condition flag on the updated packet descriptor when it is returned by the EMAC.

31.2.6.4.2 Buffer Pointer

The buffer pointer is the byte-aligned memory address of the memory buffer associated with the buffer descriptor. The software application must set this value prior to adding the descriptor to the active transmit list. This pointer is not altered by the EMAC.

31.2.6.4.3 Buffer Offset

This 16-bit field indicates how many unused bytes are at the start of the buffer. For example, a value of 0000h indicates that no unused bytes are at the start of the buffer and that valid data begins on the first byte of the buffer, while a value of 000Fh indicates that the first 15 bytes of the buffer are to be ignored by the EMAC and that valid buffer data starts on byte 16 of the buffer. The software application must set this value prior to adding the descriptor to the active transmit list. This field is not altered by the EMAC.

Note that this value is only checked on the first descriptor of a given packet (where the start of packet (SOP) flag is set). It can not be used to specify the offset of subsequent packet fragments. Also, since the buffer pointer may point to any byte–aligned address, this field may be entirely superfluous, depending on the device driver architecture.

The range of legal values for this field is 0 to (Buffer Length – 1).

31.2.6.4.4 Buffer Length

This 16-bit field indicates how many valid data bytes are in the buffer. On single fragment packets, this value is also the total length of the packet data to be transmitted. If the buffer offset field is used, the offset bytes are not counted as part of this length. This length counts only valid data bytes. The software application must set this value prior to adding the descriptor to the active transmit list. This field is not altered by the EMAC.

31.2.6.4.5 Packet Length

This 16-bit field specifies the number of data bytes in the entire packet. Any leading buffer offset bytes are not included. The sum of the buffer length fields of each of the packet’s fragments (if more than one) must be equal to the packet length. The software application must set this value prior to adding the descriptor to the active transmit list. This field is not altered by the EMAC. This value is only checked on the first descriptor of a given packet (where the start of packet (SOP) flag is set).

31.2.6.4.6 Start of Packet (SOP) Flag

When set, this flag indicates that the descriptor points to a packet buffer that is the start of a new packet. In the case of a single fragment packet, both the SOP and end of packet (EOP) flags are set. Otherwise, the descriptor pointing to the last packet buffer for the packet sets the EOP flag. This bit is set by the software application and is not altered by the EMAC.

31.2.6.4.7 End of Packet (EOP) Flag

When set, this flag indicates that the descriptor points to a packet buffer that is last for a given packet. In the case of a single fragment packet, both the start of packet (SOP) and EOP flags are set. Otherwise, the descriptor pointing to the last packet buffer for the packet sets the EOP flag. This bit is set by the software application and is not altered by the EMAC.

31.2.6.4.8 Ownership (OWNER) Flag

When set, this flag indicates that all the descriptors for the given packet (from SOP to EOP) are currently owned by the EMAC. This flag is set by the software application on the SOP packet descriptor before adding the descriptor to the transmit descriptor queue. For a single fragment packet, the SOP, EOP, and OWNER flags are all set. The OWNER flag is cleared by the EMAC once it is finished with all the descriptors for the given packet. Note that this flag is valid on SOP descriptors only.

31.2.6.4.9 End of Queue (EOQ) Flag

When set, this flag indicates that the descriptor in question was the last descriptor in the transmit queue for a given transmit channel, and that the transmitter has halted. This flag is initially cleared by the software application prior to adding the descriptor to the transmit queue. This bit is set by the EMAC when the EMAC identifies that a descriptor is the last for a given packet (the EOP flag is set), and there are no more descriptors in the transmit list (next descriptor pointer is NULL).

The software application can use this bit to detect when the EMAC transmitter for the corresponding channel has halted. This is useful when the application appends additional packet descriptors to a transmit queue list that is already owned by the EMAC. Note that this flag is valid on EOP descriptors only.

31.2.6.4.10 Teardown Complete (TDOWNCMPLT) Flag

This flag is used when a transmit queue is being torn down, or aborted, instead of allowing it to be transmitted. This would happen under device driver reset or shutdown conditions. The EMAC sets this bit in the SOP descriptor of each packet as it is aborted from transmission.

Note that this flag is valid on SOP descriptors only. Also note that only the first packet in an unsent list has the TDOWNCMPLT flag set. Subsequent descriptors are not processed by the EMAC.

31.2.6.4.11 Pass CRC (PASSCRC) Flag

This flag is set by the software application in the SOP packet descriptor before it adds the descriptor to the transmit queue. Setting this bit indicates to the EMAC that the 4 byte Ethernet CRC is already present in the packet data, and that the EMAC should not generate its own version of the CRC.

When the CRC flag is cleared, the EMAC generates and appends the 4-byte CRC. The buffer length and packet length fields do not include the CRC bytes. When the CRC flag is set, the 4-byte CRC is supplied by the software application and is already appended to the end of the packet data. The buffer length and packet length fields include the CRC bytes, as they are part of the valid packet data. Note that this flag is valid on SOP descriptors only.

31.2.6.5 Receive Buffer Descriptor Format

A receive (RX) buffer descriptor (Figure 31-11) is a contiguous block of four 32-bit data words aligned on a 32-bit boundary that describes a packet or a packet fragment. Example 31-2 shows the receive buffer descriptor described by a C structure.

31.2.6.5.1 Next Descriptor Pointer

This pointer points to the 32–bit word aligned memory address of the next buffer descriptor in the receive queue. This pointer is used to create a linked list of buffer descriptors. If the value of this pointer is zero, then the current buffer is the last buffer in the queue. The software application must set this value prior to adding the descriptor to the active receive list. This pointer is not altered by the EMAC.

The value of pNext should never be altered once the descriptor is in an active receive queue, unless its current value is NULL. If the pNext pointer is initially NULL, and more empty buffers can be added to the pool, the software application may alter this pointer to point to a newly appended descriptor. The EMAC will use the new pointer value and proceed to the next descriptor unless the pNext value has already been read. In this latter case, the receiver will halt the receive channel in question, and the software application may restart it at that time. The software can detect this case by checking for an end of queue (EOQ) condition flag on the updated packet descriptor when it is returned by the EMAC.

31.2.6.5.2 Buffer Pointer

The buffer pointer is the byte-aligned memory address of the memory buffer associated with the buffer descriptor. The software application must set this value prior to adding the descriptor to the active receive list. This pointer is not altered by the EMAC.

Figure 31-11. Receive Buffer Descriptor Format

Word 0

31-0
Next Descriptor Pointer

Word 1

31-0
Buffer Pointer

Word 2

31-16	15-0
Buffer Offset	Buffer Length

Word 3

31	30	29	28	27	26	25	24
SOP	EOP	OWNER	EOQ	TDOWNCMPLT	PASSCRC	JABBER	OVERSIZE

23	22	21	20	19	18	17	16
FRAGMENT	UNDERSIZED	CONTROL	OVERRUN	CODEERROR	ALIGNERROR	CRCERROR	NOMATCH

15-0
Packet Length

Example 31-2. Receive Buffer Descriptor in C Structure Format

```
/*
// EMAC Descriptor
//
// The following is the format of a single buffer descriptor
// on the EMAC.
*/
typedef struct _EMAC_Desc {
 struct _EMAC_Desc *pNext; /* Pointer to next descriptor in chain */
 Uint8 *pBuffer; /* Pointer to data buffer */
 Uint32 BufOffLen; /* Buffer Offset(MSW) and Length(LSW) */
 Uint32 PktFlgLen; /* Packet Flags(MSW) and Length(LSW) */
} EMAC_Desc;

/* Packet Flags */
#define EMAC_DSC_FLAG_SOP 0x80000000u
#define EMAC_DSC_FLAG_EOP 0x40000000u
#define EMAC_DSC_FLAG_OWNER 0x20000000u
#define EMAC_DSC_FLAG_EOQ 0x10000000u
#define EMAC_DSC_FLAG_TDOWNCMPLT 0x08000000u
#define EMAC_DSC_FLAG_PASSCRC 0x04000000u
#define EMAC_DSC_FLAG_JABBER 0x02000000u
#define EMAC_DSC_FLAG_OVERSIZE 0x01000000u
#define EMAC_DSC_FLAG_FRAGMENT 0x00800000u
#define EMAC_DSC_FLAG_UNDERSIZED 0x00400000u
#define EMAC_DSC_FLAG_CONTROL 0x00200000u
#define EMAC_DSC_FLAG_OVERRUN 0x00100000u
#define EMAC_DSC_FLAG_CODEERROR 0x00080000u
#define EMAC_DSC_FLAG_ALIGNERROR 0x00040000u
#define EMAC_DSC_FLAG_CRCERROR 0x00020000u
#define EMAC_DSC_FLAG_NOMATCH 0x00010000u
```

31.2.6.5.3 Buffer Offset

This 16-bit field must be initialized to zero by the software application before adding the descriptor to a receive queue.

Whether or not this field is updated depends on the setting of the RXBUFFEROFFSET register. When the offset register is set to a non-zero value, the received packet is written to the packet buffer at an offset given by the value of the register, and this value is also written to the buffer offset field of the descriptor.

When a packet is fragmented over multiple buffers because it does not fit in the first buffer supplied, the buffer offset only applies to the first buffer in the list, which is where the start of packet (SOP) flag is set in the corresponding buffer descriptor. In other words, the buffer offset field is only updated by the EMAC on SOP descriptors.

The range of legal values for the BUFFEROFFSET register is 0 to (Buffer Length – 1) for the smallest value of buffer length for all descriptors in the list.

31.2.6.5.4 Buffer Length

This 16-bit field is used for two purposes:

- Before the descriptor is first placed on the receive queue by the application software, the buffer length field is first initialized by the software to have the physical size of the empty data buffer pointed to by the buffer pointer field.
- After the empty buffer has been processed by the EMAC and filled with received data bytes, the buffer length field is updated by the EMAC to reflect the actual number of valid data bytes written to the buffer.

31.2.6.5.5 Packet Length

This 16-bit field specifies the number of data bytes in the entire packet. This value is initialized to zero by the software application for empty packet buffers. The value is filled in by the EMAC on the first buffer used for a given packet. This is signified by the EMAC setting a start of packet (SOP) flag. The packet length is set by the EMAC on all SOP buffer descriptors.

31.2.6.5.6 Start of Packet (SOP) Flag

When set, this flag indicates that the descriptor points to a packet buffer that is the start of a new packet. In the case of a single fragment packet, both the SOP and end of packet (EOP) flags are set. Otherwise, the descriptor pointing to the last packet buffer for the packet has the EOP flag set. This flag is initially cleared by the software application before adding the descriptor to the receive queue. This bit is set by the EMAC on SOP descriptors.

31.2.6.5.7 End of Packet (EOP) Flag

When set, this flag indicates that the descriptor points to a packet buffer that is last for a given packet. In the case of a single fragment packet, both the start of packet (SOP) and EOP flags are set. Otherwise, the descriptor pointing to the last packet buffer for the packet has the EOP flag set. This flag is initially cleared by the software application before adding the descriptor to the receive queue. This bit is set by the EMAC on EOP descriptors.

31.2.6.5.8 Ownership (OWNER) Flag

When set, this flag indicates that the descriptor is currently owned by the EMAC. This flag is set by the software application before adding the descriptor to the receive descriptor queue. This flag is cleared by the EMAC once it is finished with a given set of descriptors, associated with a received packet. The flag is updated by the EMAC on SOP descriptor only. So when the application identifies that the OWNER flag is cleared on an SOP descriptor, it may assume that all descriptors up to and including the first with the EOP flag set have been released by the EMAC. (Note that in the case of single buffer packets, the same descriptor will have both the SOP and EOP flags set.)

31.2.6.5.9 End of Queue (EOQ) Flag

When set, this flag indicates that the descriptor in question was the last descriptor in the receive queue for a given receive channel, and that the corresponding receiver channel has halted. This flag is initially cleared by the software application prior to adding the descriptor to the receive queue. This bit is set by the EMAC when the EMAC identifies that a descriptor is the last for a given packet received (also sets the EOP flag), and there are no more descriptors in the receive list (next descriptor pointer is NULL).

The software application can use this bit to detect when the EMAC receiver for the corresponding channel has halted. This is useful when the application appends additional free buffer descriptors to an active receive queue. Note that this flag is valid on EOP descriptors only.

31.2.6.5.10 Teardown Complete (TDOWNCMPLT) Flag

This flag is used when a receive queue is being torn down, or aborted, instead of being filled with received data. This would happen under device driver reset or shutdown conditions. The EMAC sets this bit in the descriptor of the first free buffer when the tear down occurs. No additional queue processing is performed.

31.2.6.5.11 Pass CRC (PASSCRC) Flag

This flag is set by the EMAC in the SOP buffer descriptor if the received packet includes the 4-byte CRC. This flag should be cleared by the software application before submitting the descriptor to the receive queue.

31.2.6.5.12 Jabber Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet is a jabber frame and was not discarded because the RXCEFEN bit was set in the RXMBPENABLE. Jabber frames are frames that exceed the RXMAXLEN in length, and have CRC, code, or alignment errors.

31.2.6.5.13 Oversize Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet is an oversized frame and was not discarded because the RXCEFEN bit was set in the RXMBPENABLE.

31.2.6.5.14 Fragment Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet is only a packet fragment and was not discarded because the RXCEFEN bit was set in the RXMBPENABLE.

31.2.6.5.15 Undersized Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet is undersized and was not discarded because the RXCSFEN bit was set in the RXMBPENABLE.

31.2.6.5.16 Control Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet is an EMAC control frame and was not discarded because the RXCMFEN bit was set in the RXMBPENABLE.

31.2.6.5.17 Overrun Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet was aborted due to a receive overrun.

31.2.6.5.18 Code Error (CODEERROR) Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet contained a code error and was not discarded because the RXCEFEN bit was set in the RXMBPENABLE.

31.2.6.5.19 Alignment Error (ALIGNERROR) Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet contained an alignment error and was not discarded because the RXCEFEN bit was set in the RXMBPENABLE.

31.2.6.5.20 CRC Error (CRCERROR) Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet contained a CRC error and was not discarded because the RXCEFEN bit was set in the RXMBPENABLE.

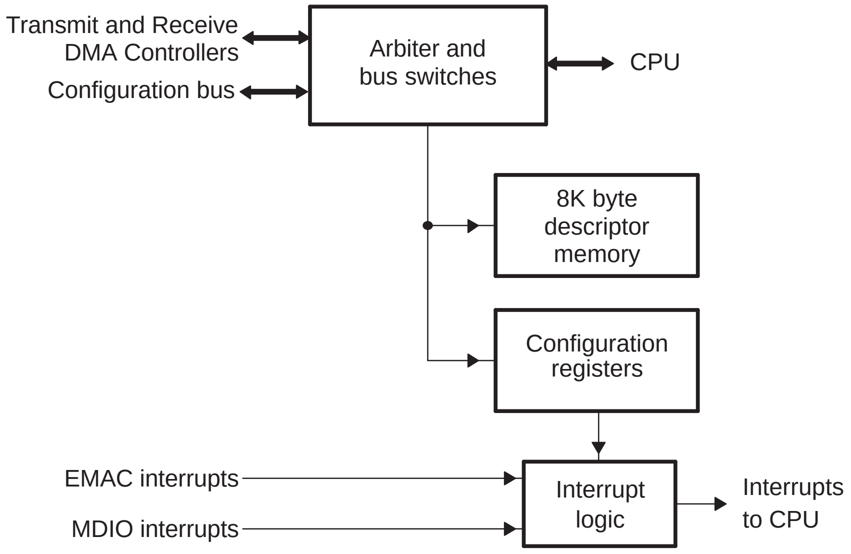
31.2.6.5.21 No Match (NOMATCH) Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet did not pass any of the EMAC's address match criteria and was not discarded because the RXCAFEN bit was set in the RXMBPENABLE. Although the packet is a valid Ethernet data packet, it was only received because the EMAC is in promiscuous mode.

31.2.7 EMAC Control Module

The EMAC control module (Figure 31-12) interfaces the EMAC and MDIO modules to the rest of the system, and also provides a local memory space to hold EMAC packet buffer descriptors. Local memory is used to help avoid contention with device memory spaces. Other functions include the bus arbiter, and interrupt logic control.

Figure 31-12. EMAC Control Module Block Diagram



31.2.7.1 Internal Memory

The EMAC control module includes 8K bytes of internal memory (CPPI buffer descriptor memory). The internal memory block is essential for allowing the EMAC to operate more independently of the CPU. It also prevents memory underflow conditions when the EMAC issues read or write requests to descriptor memory. (Memory accesses to read or write the actual Ethernet packet data are protected by the EMAC's internal FIFOs).

A descriptor is a 16-byte memory structure that holds information about a single Ethernet packet buffer, which may contain a full or partial Ethernet packet. Thus with the 8K memory block provided for descriptor storage, the EMAC module can send and received up to a combined 512 packets before it needs to be serviced by application or driver software.

31.2.7.2 Bus Arbiter

The EMAC control module bus arbiter operates transparently to the rest of the system. It is used:

- To arbitrate between the CPU and EMAC buses for access to internal descriptor memory.
- To arbitrate between internal EMAC buses for access to system memory.

31.2.7.3 Interrupt Control

Interrupt conditions generated by the EMAC and MDIO modules are combined into four interrupt signals that are routed to the Vectored Interrupt Manager (VIM); the VIM then relays the interrupt signals to the CPU. The EMAC control module uses two sets of registers to control the interrupt signals to the CPU:

- C0RXTHRESHEN, C0RXEN, C0TXEN, and C0MISCEN registers enable the pulse signals that are mapped to the VIM
- INTCONTROL, C0RXIMAX, and C0TXIMAX registers enable interrupt pacing to limit the number of interrupt pulses generated per millisecond

Interrupts must be acknowledged by writing the appropriate value to the EMAC End-Of-Interrupt Vector (MACEOIVECTOR). The MACEOIVECTOR behaves as an interrupt pulse interlock -- once the EMAC control module has issued an interrupt pulse to the CPU, it will not generate further pulses of the same type until the original pulse has been acknowledged.

31.2.8 MDIO Module

The MDIO module is used to manage up to 32 physical layer (PHY) devices connected to the Ethernet Media Access Controller (EMAC). The device supports a single PHY being connected to the EMAC at any given time. The MDIO module is designed to allow almost transparent operation of the MDIO interface with little maintenance from the CPU.

The MDIO module continuously polls 32 MDIO addresses in order to enumerate all PHY devices in the system. Once a PHY device has been detected, the MDIO module reads the MDIO PHY link status register (LINK) to monitor the PHY link state. Link change events are stored in the MDIO module, which can interrupt the CPU. This storing of the events allows the CPU to poll the link status of the PHY device without continuously performing MDIO module accesses. However, when the CPU must access the MDIO module for configuration and negotiation, the MDIO module performs the MDIO read or write operation independent of the CPU. This independent operation allows the processor to poll for completion or interrupt the CPU once the operation has completed.

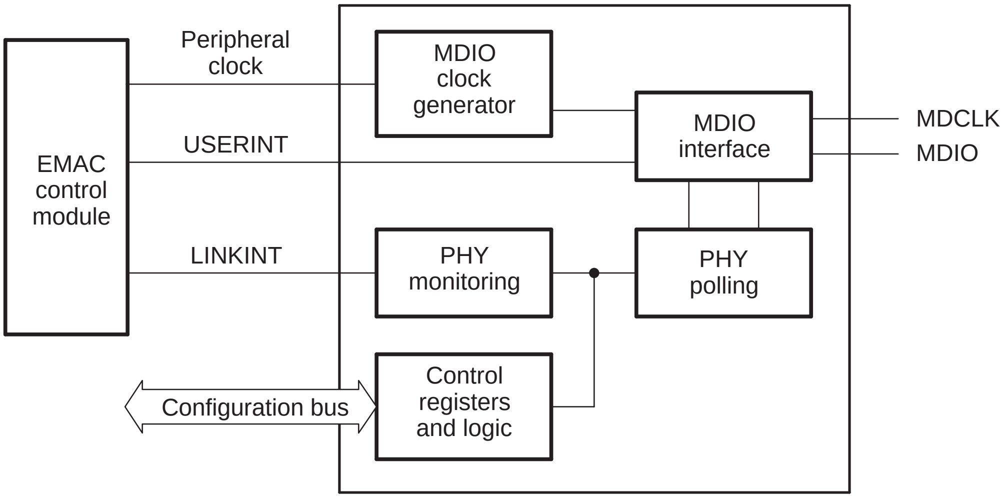
The MDIO module does not support the "Clause 45" interface.

31.2.8.1 MDIO Module Components

The MDIO module (Figure 31-13) interfaces to the PHY components through two MDIO pins (MDIO_CLK and MDIO), and to the CPU through the EMAC control module and the configuration bus. The MDIO module consists of the following logical components:

- MDIO clock generator
- Global PHY detection and link state monitoring
- Active PHY monitoring
- PHY register user access

Figure 31-13. MDIO Module Block Diagram



31.2.8.1.1 MDIO Clock Generator

The MDIO clock generator controls the MDIO clock based on a divide-down of the VCLK3 peripheral clock in the EMAC control module. The MDIO clock is specified to run up to 2.5 MHz, although typical operation would be 1.0 MHz. Since the VCLK3 peripheral clock frequency is configurable, the application software or driver controls the divide-down amount. See the device datasheet for peripheral clock speed specifications.

31.2.8.1.2 Global PHY Detection and Link State Monitoring

The MDIO module continuously polls all 32 MDIO addresses in order to enumerate the PHY devices in the system. The module tracks whether or not a PHY on a particular address has responded, and whether or not the PHY currently has a link. Using this information allows the software application to quickly determine which MDIO address the PHY is using.

31.2.8.1.3 Active PHY Monitoring

Once a PHY candidate has been selected for use, the MDIO module transparently monitors its link state by reading the MDIO PHY link status register (LINK). Link change events are stored on the MDIO device and can optionally interrupt the CPU. This allows the system to poll the link status of the PHY device without continuously performing costly MDIO accesses.

31.2.8.1.4 PHY Register User Access

When the CPU must access MDIO for configuration and negotiation, the PHY access module performs the actual MDIO read or write operation independent of the CPU. This allows the CPU to poll for completion or receive an interrupt when the read or write operation has been performed. The user access registers USERACCESS*n* allows the software to submit the access requests for the PHY connected to the device.

31.2.8.2 MDIO Module Operational Overview

The MDIO module implements the 802.3 serial management interface to interrogate and control an Ethernet PHY, using a shared two-wired bus. It separately performs autodetection and records the current link status of up to 32 PHYs, polling all 32 MDIO addresses.

Application software uses the MDIO module to configure the autonegotiation parameters of the PHY attached to the EMAC, retrieve the negotiation results, and configure required parameters in the EMAC.

In this device, the Ethernet PHY attached to the system can be directly controlled and queried. The Media Independent Interface (MII) address of this PHY device is specified in one of the PHYADRMON bits in the MDIO user PHY select register (USERPHYSEL*n*). The MDIO module can be programmed to trigger a CPU interrupt on a PHY link change event, by setting the LINKINTENB bit in USERPHYSEL*n*. Reads and writes to registers in this PHY device are performed using the MDIO user access register (USERACCESS*n*).

The MDIO module powers-up in an idle state until specifically enabled by setting the ENABLE bit in the MDIO control register (CONTROL). At this time, the MDIO clock divider and preamble mode selection are also configured. The MDIO preamble is enabled by default, but can be disabled when the connected PHY does not require it. Once the MDIO module is enabled, the MDIO interface state machine continuously polls the PHY link status (by reading the generic status register) of all possible 32 PHY addresses and records the results in the MDIO PHY alive status register (ALIVE) and MDIO PHY link status register (LINK). The corresponding bit for the connected PHY (0-31) is set in ALIVE, if the PHY responded to the read request. The corresponding bit is set in LINK, if the PHY responded and also is currently linked. In addition, any PHY register read transactions initiated by the application software using USERACCESS*n* causes ALIVE to be updated.

The USERPHYSEL*n* is used to track the link status of the connected PHY address. A change in the link status of the PHY being monitored sets the appropriate bit in the MDIO link status change interrupt registers (LINKINTRAW and LINKINTMASKED), if enabled by the LINKINTENB bit in USERPHYSEL*n*.

While the MDIO module is enabled, the host issues a read or write transaction over the MII management interface using the DATA, PHYADR, REGADR, and WRITE bits in USERACCESS*n*. When the application sets the GO bit in USERACCESS*n*, the MDIO module begins the transaction without any further intervention from the CPU. Upon completion, the MDIO module clears the GO bit and sets the corresponding USERINTRAW bit (0 or 1) in the MDIO user command complete interrupt register (USERINTRAW) corresponding to USERACCESS*n* used. The corresponding USERINTMASKED bit (0 or 1) in the MDIO user command complete interrupt register (USERINTMASKED) may also be set, depending on the mask setting configured in the MDIO user command complete interrupt mask set register (USERINTMASKSET) and the MDIO user interrupt mask clear register (USERINTMASKCLEAR).

A round-robin arbitration scheme is used to schedule transactions that may be queued using both USERACCESS0 and USERACCESS1. The application software must check the status of the GO bit in USERACCESS*n* before initiating a new transaction, to ensure that the previous transaction has completed. The application software can use the ACK bit in USERACCESS*n* to determine the status of a read transaction.

31.2.8.2.1 Initializing the MDIO Module

The following steps are performed by the application software or device driver to initialize the MDIO device:

1. Configure the PREAMBLE and CLKDIV bits in the MDIO control register (CONTROL).
2. Enable the MDIO module by setting the ENABLE bit in CONTROL.
3. The MDIO PHY alive status register (ALIVE) can be read in polling fashion until a PHY connected to the system responded, and the MDIO PHY link status register (LINK) can determine whether this PHY already has a link.
4. Setup the appropriate PHY addresses in the MDIO user PHY select register (USERPHYSEL*n*), and set the LINKINTENB bit to enable a link change event interrupt if desirable.
5. If an interrupt on general MDIO register access is desired, set the corresponding bit in the MDIO user command complete interrupt mask set register (USERINTMASKSET) to use the MDIO user access register (USERACCESS*n*). Since only one PHY is used in this device, the application software can use one USERACCESS*n* to trigger a completion interrupt; the other USERACCESS*n* is not setup.

31.2.8.2.2 Writing Data To a PHY Register

The MDIO module includes a user access register (USERACCESS*n*) to directly access a specified PHY device. To write a PHY register, perform the following:

1. Check to ensure that the GO bit in the MDIO user access register (USERACCESS*n*) is cleared.
2. Write to the GO, WRITE, REGADR, PHYADR, and DATA bits in USERACCESS*n* corresponding to the PHY and PHY register you want to write.
3. The write operation to the PHY is scheduled and completed by the MDIO module. Completion of the write operation can be determined by polling the GO bit in USERACCESS*n* for a 0.
4. Completion of the operation sets the corresponding USERINTRAW bit (0 or 1) in the MDIO user command complete interrupt register (USERINTRAW) corresponding to USERACCESS*n* used. If interrupts have been enabled on this bit using the MDIO user command complete interrupt mask set register (USERINTMASKSET), then the bit is also set in the MDIO user command complete interrupt register (USERINTMASKED) and an interrupt is triggered on the CPU.

31.2.8.2.3 Reading Data From a PHY Register

The MDIO module includes a user access register (USERACCESS*n*) to directly access a specified PHY device. To read a PHY register, perform the following:

1. Check to ensure that the GO bit in the MDIO user access register (USERACCESS*n*) is cleared.
2. Write to the GO, REGADR, and PHYADR bits in USERACCESS*n* corresponding to the PHY and PHY register you want to read.
3. The read data value is available in the DATA bits in USERACCESS*n* after the module completes the read operation on the serial bus. Completion of the read operation can be determined by polling the GO and ACK bits in USERACCESS*n*. Once the GO bit has cleared, the ACK bit is set on a successful read.
4. Completion of the operation sets the corresponding USERINTRAW bit (0 or 1) in the MDIO user command complete interrupt register (USERINTRAW) corresponding to USERACCESS*n* used. If interrupts have been enabled on this bit using the MDIO user command complete interrupt mask set register (USERINTMASKSET), then the bit is also set in the MDIO user command complete interrupt register (USERINTMASKED) and an interrupt is triggered on the CPU.

31.2.8.2.4 Example of MDIO Register Access Code

The MDIO module uses the MDIO user access register (USERACCESS*n*) to access the PHY control registers. Software functions that implement the access process may simply be the following four macros:

• PHYREG_read(regadr, phyadr)	Start the process of reading a PHY register
• PHYREG_write(regadr, phyadr, data)	Start the process of writing a PHY register
• PHYREG_wait( )	Synchronize operation (make sure read/write is idle)
• PHYREG_waitResults(results)	Wait for read to complete and return data read

Note that it is not necessary to wait after a write operation, as long as the status is checked before every operation to make sure the MDIO hardware is idle. An alternative approach is to call PHYREG_wait() after every write, and PHYREG_waitResults( ) after every read, then the hardware can be assumed to be idle when starting a new operation.

The implementation of these macros using the chip support library (CSL) is shown in Example 31-3 (USERACCESS0 is assumed).

Note that this implementation does not check the ACK bit in USERACCESS*n* on PHY register reads (does not follow the procedure outlined in Section 31.2.8.2.3). Since the MDIO PHY alive status register (ALIVE) is used to initially select a PHY, it is assumed that the PHY is acknowledging read operations. It is possible that a PHY could become inactive at a future point in time. An example of this would be a PHY that can have its MDIO addresses changed while the system is running. It is not very likely, but this condition can be tested by periodically checking the PHY state in ALIVE.

Example 31-3. MDIO Register Access Macros

```
#define PHYREG_read(regadr, phyadr)
    MDIO_REGS->USERACCESS0 =
        CSL_FMK(MDIO_USERACCESS0_GO,1u)                        | /
        CSL_FMK(MDIO_USERACCESS0_REGADR,regadr)                | /
        CSL_FMK(MDIO_USERACCESS0_PHYADR,phyadr)
#define PHYREG_write(regadr, phyadr, data)
    MDIO_REGS->USERACCESS0 =
        CSL_FMK(MDIO_USERACCESS0_GO,1u)                        | /
        CSL_FMK(MDIO_USERACCESS0_WRITE,1)                      | /
        CSL_FMK(MDIO_USERACCESS0_REGADR,regadr)                | /
        CSL_FMK(MDIO_USERACCESS0_PHYADR,phyadr)                | /
        CSL_FMK(MDIO_USERACCESS0_DATA, data)
#define PHYREG_wait()
    while( CSL_FEXT(MDIO_REGS->USERACCESS0,MDIO_USERACCESS0_GO) )
#define PHYREG_waitResults( results ) {
    while( CSL_FEXT(MDIO_REGS->USERACCESS0,MDIO_USERACCESS0_GO) );
results = CSL_FEXT(MDIO_REGS->USERACCESS0, MDIO_USERACCESS0_DATA); }
```

31.2.9 EMAC Module

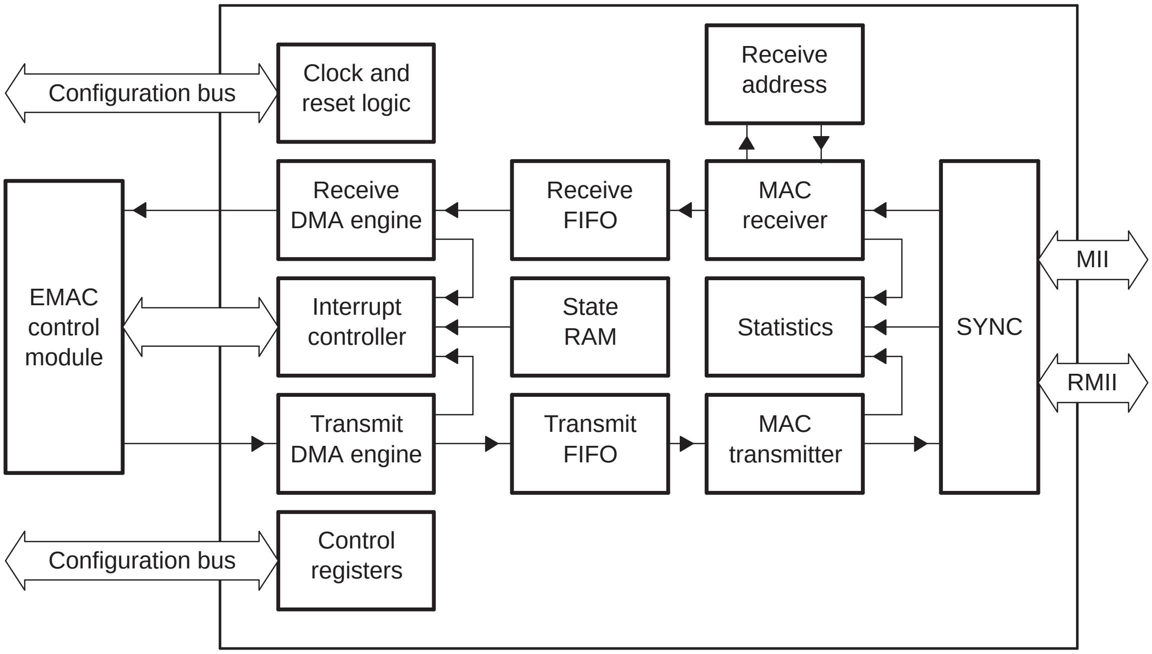
This section discusses the architecture and basic function of the EMAC module.

31.2.9.1 EMAC Module Components

The EMAC module (Figure 31-14) interfaces to the outside world through the Media Independent Interface (MII) or Reduced Media Independent Interface (RMII). The interface between the EMAC module and the system core is provided through the EMAC control module. The EMAC consists of the following logical components:

- The receive path includes: receive DMA engine, receive FIFO, and MAC receiver
- The transmit path includes: transmit DMA engine, transmit FIFO, and MAC transmitter
- Statistics logic
- State RAM
- Interrupt controller
- Control registers and logic
- Clock and reset logic

Figure 31-14. EMAC Module Block Diagram



31.2.9.1.1 Receive DMA Engine

The receive DMA engine is the interface between the receive FIFO and the system core. It interfaces to the CPU through the bus arbiter in the EMAC control module. This DMA engine is totally independent of the device DMA.

31.2.9.1.2 Receive FIFO

The receive FIFO consists of three cells of 64-bytes each and associated control logic. The FIFO buffers receive data in preparation for writing into packet buffers in device memory.

31.2.9.1.3 MAC Receiver

The MAC receiver detects and processes incoming network frames, de-frames them, and puts them into the receive FIFO. The MAC receiver also detects errors and passes statistics to the statistics RAM.

31.2.9.1.4 Transmit DMA Engine

The transmit DMA engine is the interface between the transmit FIFO and the CPU. It interfaces to the CPU through the bus arbiter in the EMAC control module.

31.2.9.1.5 Transmit FIFO

The transmit FIFO consists of three cells of 64-bytes each and associated control logic. The FIFO buffers data in preparation for transmission.

31.2.9.1.6 MAC Transmitter

The MAC transmitter formats frame data from the transmit FIFO and transmits the data using the CSMA/CD access protocol. The frame CRC can be automatically appended, if required. The MAC transmitter also detects transmission errors and passes statistics to the statistics registers.

31.2.9.1.7 Statistics Logic

The Ethernet statistics are counted and stored in the statistics logic RAM. This statistics RAM keeps track of 36 different Ethernet packet statistics.

31.2.9.1.8 State RAM

State RAM contains the head descriptor pointers and completion pointers registers for both transmit and receive channels.

31.2.9.1.9 EMAC Interrupt Controller

The interrupt controller contains the interrupt related registers and logic. The 26 raw EMAC interrupts are input to this submodule and masked module interrupts are output.

31.2.9.1.10 Control Registers and Logic

The EMAC is controlled by a set of memory-mapped registers. The control logic also signals transmit, receive, and status related interrupts to the CPU through the EMAC control module.

31.2.9.1.11 Clock and Reset Logic

The clock and reset submodule generates all the EMAC clocks and resets. For more details on reset capabilities, see Section 31.2.15.1.

31.2.9.2 EMAC Module Operational Overview

After reset, initialization, and configuration, the host may initiate transmit operations. Transmit operations are initiated by host writes to the appropriate transmit channel head descriptor pointer contained in the state RAM block. The transmit DMA controller then fetches the first packet in the packet chain from memory. The DMA controller writes the packet into the transmit FIFO in bursts of 64-byte cells. When the threshold number of cells, configurable using the TXCELLTHRESH bit in the FIFO control register (FIFOCONTROL), have been written to the transmit FIFO, or a complete packet, whichever is smaller, the MAC transmitter then initiates the packet transmission. The SYNC block transmits the packet over the MII or RMII interfaces in accordance with the 802.3 protocol. Transmit statistics are counted by the statistics block.

Receive operations are initiated by host writes to the appropriate receive channel head descriptor pointer after host initialization and configuration. The SYNC submodule receives packets and strips off the Ethernet related protocol. The packet data is input to the MAC receiver, which checks for address match and processes errors. Accepted packets are then written to the receive FIFO in bursts of 64-byte cells. The receive DMA controller then writes the packet data to memory. Receive statistics are counted by the statistics block.

The EMAC module operates independently of the CPU. It is configured and controlled by its register set mapped into device memory. Information about data packets is communicated by use of 16-byte descriptors that are placed in an 8K-byte block of RAM in the EMAC control module (CPPI buffer descriptor memory).

For transmit operations, each 16-byte descriptor describes a packet or packet fragment in the system's internal or external memory. For receive operations, each 16-byte descriptor represents a free packet buffer or buffer fragment. On both transmit and receive, an Ethernet packet is allowed to span one or more memory fragments, represented by one 16-byte descriptor per fragment. In typical operation, there is only one descriptor per receive buffer, but transmit packets may be fragmented, depending on the software architecture.

An interrupt is issued to the CPU whenever a transmit or receive operation has completed. However, it is not necessary for the CPU to service the interrupt while there are additional resources available. In other words, the EMAC continues to receive Ethernet packets until its receive descriptor list has been exhausted. On transmit operations, the transmit descriptors need only be serviced to recover their associated memory buffer. Thus, it is possible to delay servicing of the EMAC interrupt if there are real-time tasks to perform.

Eight channels are supplied for both transmit and receive operations. On transmit, the eight channels represent eight independent transmit queues. The EMAC can be configured to treat these channels as an equal priority "round-robin" queue or as a set of eight fixed-priority queues. On receive, the eight channels represent eight independent receive queues with packet classification. Packets are classified based on the destination MAC address. Each of the eight channels is assigned its own MAC address, enabling the EMAC module to act like eight virtual MAC adapters. Also, specific types of frames can be sent to specific channels. For example, multicast, broadcast, or other (promiscuous, error, etc.), can each be received on a specific receive channel queue.

The EMAC keeps track of 36 different statistics, plus keeps the status of each individual packet in its corresponding packet descriptor.

31.2.10 MAC Interface

The following sections discuss the operation of the Media Independent Interface (MII) and Reduced Media Independent Interface (RMII) in 10 Mbps and 100 Mbps mode. An IEEE 802.3 compliant Ethernet MAC controls the interface.

31.2.10.1 Data Reception

31.2.10.1.1 Receive Control

Data received from the PHY is interpreted and output to the EMAC receive FIFO. Interpretation involves detection and removal of the preamble and start-of-frame delimiter, extraction of the address and frame length, data handling, error checking and reporting, cyclic redundancy checking (CRC), and statistics control signal generation. Address detection and frame filtering is performed outside the MAC interface.

31.2.10.1.2 Receive Inter-Frame Interval

The 802.3 standard requires an interpacket gap (IPG), which is 96 bit times. However, the EMAC can tolerate a reduced IPG of 8 bit times with a correct preamble and start frame delimiter. This interval between frames must comprise (in the following order):

1. An Interpacket Gap (IPG).
2. A 7-byte preamble (all bytes 55h).
3. A 1-byte start of frame delimiter (5Dh).

31.2.10.1.3 Receive Flow Control

When enabled and triggered, receive flow control is initiated to limit the EMAC from further frame reception. Two forms of receive buffer flow control are available:

- Collision-based flow control for half-duplex mode
- IEEE 802.3x pause frames flow control for full-duplex mode

In either case, receive flow control prevents frame reception by issuing the flow control appropriate for the current mode of operation. Receive flow control prevents reception of frames on the EMAC until all of the triggering conditions clear, at which time frames may again be received by the EMAC.

Receive flow control is enabled by the RXBUFFERFLOWEN bit in the MAC control register (MACCONTROL). The EMAC is configured for collision or IEEE 802.3X flow control using the FULLDUPLEX bit in MACCONTROL. Receive flow control is triggered when the number of free buffers in any enabled receive channel free buffer count register (RX*n*FREEBUFFER) is less than or equal to the receive channel flow control threshold register (RX*n*FLOWTHRESH) value. Receive flow control is independent of receive QOS, except that both use the free buffer values.

31.2.10.1.3.1 Collision-Based Receive Buffer Flow Control

Collision-based receive buffer flow control provides a means of preventing frame reception when the EMAC is operating in half-duplex mode (the FULLDUPLEX bit is cleared in MACCONTROL). When receive flow control is enabled and triggered, the EMAC generates collisions for received frames. The jam sequence transmitted is the 12-byte sequence C3.C3.C3.C3.C3.C3.C3.C3.C3.C3.C3.C3h. The jam sequence begins no later than approximately as the source address starts to be received. Note that these forced collisions are not limited to a maximum of 16 consecutive collisions, and are independent of the normal back-off algorithm.

Receive flow control does not depend on the value of the incoming frame destination address. A collision is generated for any incoming packet, regardless of the destination address, if any EMAC enabled channel’s free buffer register value is less than or equal to the channel’s flow threshold value.

31.2.10.1.3.2 IEEE 802.3x-Based Receive Buffer Flow Control

IEEE 802.3x-based receive buffer flow control provides a means of preventing frame reception when the EMAC is operating in full-duplex mode (the FULLDUPLEX bit is set in MACCONTROL). When receive flow control is enabled and triggered, the EMAC transmits a pause frame to request that the sending station stop transmitting for the period indicated within the transmitted pause frame.

The EMAC transmits a pause frame to the reserved multicast address at the first available opportunity (immediately if currently idle or following the completion of the frame currently being transmitted). The pause frame contains the maximum possible value for the pause time (FFFFh). The EMAC counts the receive pause frame time (decrements FF00h to 0) and retransmits an outgoing pause frame, if the count reaches 0. When the flow control request is removed, the EMAC transmits a pause frame with a zero pause time to cancel the pause request.

Note that transmitted pause frames are only a request to the other end station to stop transmitting. Frames that are received during the pause interval are received normally (provided the receive FIFO is not full).

Pause frames are transmitted if enabled and triggered, regardless of whether or not the EMAC is observing the pause time period from an incoming pause frame.

The EMAC transmits pause frames as described below:

- The 48-bit reserved multicast destination address 01.80.C2.00.00.01h.
- The 48-bit source address (set using the MACSRCADDRLO and MACSRCADDRHI registers).
- The 16-bit length/type field containing the value 88.08h.
- The 16-bit pause opcode equal to 00.01h.
- The 16-bit pause time value of FF.FFh. A pause-quantum is 512 bit-times. Pause frames sent to cancel a pause request have a pause time value of 00.00h.
- Zero padding to 64-byte data length (EMAC transmits only 64-byte pause frames).

- The 32-bit frame-check sequence (CRC word).

All quantities are hexadecimal and are transmitted most-significant byte first. The least-significant bit (LSB) is transferred first in each byte.

If the RXBUFFERFLOWEN bit in MACCONTROL is cleared to 0 while the pause time is nonzero, then the pause time is cleared to 0 and a zero count pause frame is sent.

31.2.10.2 Data Transmission

The EMAC passes data to the PHY from the transmit FIFO (when enabled). Data is synchronized to the transmit clock rate. Transmission begins when there are TXCELLTHRESH cells of 64 bytes each, or a complete packet, in the FIFO.

31.2.10.2.1 Transmit Control

A jam sequence is output if a collision is detected on a transmit packet. If the collision was late (after the first 64 bytes have been transmitted), the collision is ignored. If the collision is not late, the controller will back off before retrying the frame transmission. When operating in full-duplex mode, the carrier sense (MII_CRS) and collision-sensing (MII_COL) modes are disabled.

31.2.10.2.2 CRC Insertion

If the SOP buffer descriptor PASSCRC flag is cleared, the EMAC generates and appends a 32-bit Ethernet CRC onto the transmitted data. For the EMAC-generated CRC case, a CRC (or placeholder) at the end of the data is allowed but not required. The buffer byte count value should not include the CRC bytes, if they are present.

If the SOP buffer descriptor PASSCRC flag is set, then the last four bytes of the transmit data are transmitted as the frame CRC. The four CRC data bytes should be the last four bytes of the frame and should be included in the buffer byte count value. The MAC performs no error checking on the outgoing CRC.

31.2.10.2.3 Adaptive Performance Optimization (APO)

The EMAC incorporates adaptive performance optimization (APO) logic that may be enabled by setting the TXPACE bit in the MAC control register (MACCONTROL). Transmission pacing to enhance performance is enabled when the TXPACE bit is set. Adaptive performance pacing introduces delays into the normal transmission of frames, delaying transmission attempts between stations, reducing the probability of collisions occurring during heavy traffic (as indicated by frame deferrals and collisions), thereby, increasing the chance of successful transmission.

When a frame is deferred, suffers a single collision, multiple collisions, or excessive collisions, the pacing counter is loaded with an initial value of 31. When a frame is transmitted successfully (without experiencing a deferral, single collision, multiple collision, or excessive collision), the pacing counter is decremented by 1, down to 0.

With pacing enabled, a new frame is permitted to immediately (after one interpacket gap) attempt transmission only if the pacing counter is 0. If the pacing counter is nonzero, the frame is delayed by the pacing delay of approximately four interpacket gap (IPG)delays. APO only affects the IPG preceding the first attempt at transmitting a frame; APO does not affect the back-off algorithm for retransmitted frames.

31.2.10.2.4 Interpacket-Gap (IPG) Enforcement

The measurement reference for the IPG of 96 bit times is changed depending on frame traffic conditions. If a frame is successfully transmitted without collision and MII_CRS is deasserted within approximately 48 bit times of MII_TXEN being deasserted, then 96 bit times is measured from MII_TXEN. If the frame suffered a collision or MII_CRS is not deasserted until more than approximately 48 bit times after MII_TXEN is deasserted, then 96 bit times (approximately, but not less) is measured from MII_CRS.

31.2.10.2.5 Back Off

The EMAC implements the 802.3 binary exponential back-off algorithm.

31.2.10.2.6 Transmit Flow Control

Incoming pause frames are acted upon, when enabled, to prevent the EMAC from transmitting any further frames. Incoming pause frames are only acted upon when the FULLDUPLEX and TXFLOWEN bits in the MAC control register (MACCONTROL) are set. Pause frames are not acted upon in half-duplex mode. Pause frame action is taken if enabled, but normally the frame is filtered and not transferred to memory. MAC control frames are transferred to memory, if the RXCMFEN bit in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE) is set. The TXFLOWEN and FULLDUPLEX bits affect whether or not MAC control frames are acted upon, but they have no affect upon whether or not MAC control frames are transferred to memory or filtered.

Pause frames are a subset of MAC control frames with an opcode field of 0001h. Incoming pause frames are only acted upon by the EMAC if:

- TXFLOWEN bit is set in MACCONTROL
- The frame’s length is 64 to RXMAXLEN bytes inclusive
- The frame contains no CRC error or align/code errors

The pause time value from valid frames is extracted from the two bytes following the opcode. The pause time is loaded into the EMAC transmit pause timer and the transmit pause time period begins. If a valid pause frame is received during the transmit pause time period of a previous transmit pause frame then:

- If the destination address is not equal to the reserved multicast address or any enabled or disabled unicast address, then the transmit pause timer immediately expires, or
- If the new pause time value is 0, then the transmit pause timer immediately expires, else
- The EMAC transmit pause timer immediately is set to the new pause frame pause time value. (Any remaining pause time from the previous pause frame is discarded).

If the TXFLOWEN bit in MACCONTROL is cleared, then the pause timer immediately expires.

The EMAC does not start the transmission of a new data frame any sooner than 512 bit-times after a pause frame with a nonzero pause time has finished being received (MII_RXDV going inactive). No transmission begins until the pause timer has expired (the EMAC may transmit pause frames in order to initiate outgoing flow control). Any frame already in transmission when a pause frame is received is completed and unaffected.

Incoming pause frames consist of:

- A 48-bit destination address equal to one of the following:
  - The reserved multicast destination address 01.80.C2.00.00.01h
  - Any EMAC 48-bit unicast address. Pause frames are accepted, regardless of whether the channel is enabled or not.
- The 16-bit length/type field containing the value 88.08h.
- The 48-bit source address of the transmitting device.
- The 16-bit pause opcode equal to 00.01h.
- The 16-bit pause time. A pause-quantum is 512 bit-times.
- Padding to 64-byte data length.
- The 32-bit frame-check sequence (CRC word).

All quantities are hexadecimal and are transmitted most-significant byte first. The least-significant bit (LSB) is transferred first in each byte.

The padding is required to make up the frame to a minimum of 64 bytes. The standard allows pause frames longer than 64 bytes to be discarded or interpreted as valid pause frames. The EMAC recognizes any pause frame between 64 bytes and RXMAXLEN bytes in length.

31.2.10.2.7 Speed, Duplex, and Pause Frame Support

The MAC operates at 10 Mbps or 100 Mbps, in half-duplex or full-duplex mode, and with or without pause frame support as configured by the host.

31.2.11 Packet Receive Operation

31.2.11.1 Receive DMA Host Configuration

To configure the receive DMA for operation the host must:

- Initialize the receive addresses.
- Initialize the receive channel *n* DMA head descriptor pointer registers (RX*n*HDP) to 0.
- Write the MAC address hash *n* registers (MACHASH1 and MACHASH2), if multicast addressing is desired.
- If flow control is to be enabled, initialize:
  - the receive channel *n* free buffer count registers (RX*n*FREEBUFFER)
  - the receive channel *n* flow control threshold register (RX*n*FLOWTHRESH)
  - the receive filter low priority frame threshold register (RXFILTERLOWTHRESH)
- Enable the desired receive interrupts using the receive interrupt mask set register (RXINTMASKSET) and the receive interrupt mask clear register (RXINTMASKCLEAR).
- Set the appropriate configuration bits in the MAC control register (MACCONTROL).
- Write the receive buffer offset register (RXBUFFEROFFSET) value (typically zero).
- Setup the receive channel(s) buffer descriptors and initialize RX*n*HDP.
- Enable the receive DMA controller by setting the RXEN bit in the receive control register (RXCONTROL).
- Configure and enable the receive operation, as desired, in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE) and by using the receive unicast set register (RXUNICASTSET) and the receive unicast clear register (RXUNICASTCLEAR).

31.2.11.2 Receive Channel Enabling

Each of the eight receive channels has an enable bit (RXCH*n*EN) in the receive unicast set register (RXUNICASTSET) that is controlled using RXUNICASTSET and the receive unicast clear register (RXUNICASTCLEAR). The RXCH*n*EN bits determine whether the given channel is enabled (when set to 1) to receive frames with a matching unicast or multicast destination address.

The RXBROADEN bit in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE) determines if broadcast frames are enabled or filtered. If broadcast frames are enabled (when set to 1), then they are copied to only a single channel selected by the RXBROADCH bit in RXMBPENABLE.

The RXMULTEN bit in RXMBPENABLE determines if hash matching multicast frames are enabled or filtered. Incoming multicast addresses (group addresses) are hashed into an index in the hash table. If the indexed bit is set then the frame hash matches and will be transferred to the channel selected by the RXMULTCH bit in RXMBPENABLE when multicast frames are enabled. The multicast hash bits are set in the MAC address hash *n* registers (MACHASH1 and MACHASH2).

The RXPROMCH bit in RXMBPENABLE selects the promiscuous channel to receive frames selected by the RXCMFEN, RXCSFEN, RXCEFEN, and RXCAFEN bits. These four bits allow reception of MAC control frames, short frames, error frames, and all frames (promiscuous), respectively.

31.2.11.3 Receive Address Matching

All eight MAC addresses corresponding to the eight receive channels share the upper 40 bits. Only the lower byte is unique for each address. All eight receive addresses should be initialized, because pause frames are acted upon regardless of whether a channel is enabled or not.

A MAC address is written by first writing the address number (channel) to be written into the MAC index register (MACINDEX). The upper 32 bits of address are then written to the MAC address high bytes register (MACADDRHI), which is followed by writing the lower 16 bits of address to the MAC address low bytes register (MACADDRLO). Since all eight MAC addresses share the upper 40 bits of address, MACADDRHI needs to be written only the first time (for the first channel configured).

31.2.11.4 Hardware Receive QOS Support

Hardware receive quality of service (QOS) is supported, when enabled, by the Tag Protocol Identifier format and the associated Tag Control Information (TCI) format priority field. When the incoming frame length/type value is equal to 81.00h, the EMAC recognizes the frame as an Ethernet Encoded Tag Protocol Type. The two octets immediately following the protocol type contain the 16-bit TCI field. Bits 15-13 of the TCI field contain the received frames priority (0 to 7). The received frame is a low-priority frame, if the priority value is 0 to 3; the received frame is a high-priority frame, if the priority value is 4 to 7. All frames that have a length/type field value not equal to 81.00h are low-priority frames. Received frames that contain priority information are determined by the EMAC as:

- A 48-bit (6 bytes) destination address equal to:
  – The destination station's individual unicast address.
  – The destination station's multicast address (MACHASH1 and MACHASH2).
  – The broadcast address of all ones.
- A 48-byte (6 bytes) source address.
- The 16-bit (2 bytes) length/type field containing the value 81.00h.
- The 16-bit (2 bytes) TCI field with the priority field in the upper 3 bits.
- Data bytes
- The 4 bytes CRC.

The receive filter low priority frame threshold register (RXFILTERLOWTHRESH) and the receive channel *n* free buffer count registers (RX*n*FREEBUFFER) are used in conjunction with the priority information to implement receive hardware QOS. Low-priority frames are filtered if the number of free buffers (RX*n*FREEBUFFER) for the frame channel is less than or equal to the filter low threshold (RXFILTERLOWTHRESH) value. Hardware QOS is enabled by the RXQOSEN bit in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE).

31.2.11.5 Host Free Buffer Tracking

The host must track free buffers for each enabled channel (including unicast, multicast, broadcast, and promiscuous), if receive QOS or receive flow control is used. Disabled channel free buffer values are do not cares. During initialization, the host should write the number of free buffers for each enabled channel to the appropriate receive channel *n* free buffer count registers (RX*n*FREEBUFFER). The EMAC decrements the appropriate channel’s free buffer value for each buffer used. When the host reclaims the frame buffers, the host should write the channel free buffer register with the number of reclaimed buffers (write to increment). There are a maximum of 65,535 free buffers available. RX*n*FREEBUFFER only needs to be updated by the host if receive QOS or flow control is used.

31.2.11.6 Receive Channel Teardown

The host commands a receive channel teardown by writing the channel number to the receive teardown register (RXTEARDOWN). When a teardown command is issued to an enabled receive channel, the following occurs:

- Any current frame in reception completes normally.
- The TDOWNCMPLT flag is set in the next buffer descriptor in the chain, if there is one.
- The channel head descriptor pointer is cleared to 0.
- A receive interrupt for the channel is issued to the host.
- The corresponding receive channel *n* completion pointer register (RX*n*CP) contains the value FFFF FFFCh.

Channel teardown may be commanded on any channel at any time. The host is informed of the teardown completion by the set teardown complete (TDOWNCMPLT) buffer descriptor bit. The EMAC does not clear any channel enables due to a teardown command. A teardown command to an inactive channel issues an interrupt that software should acknowledge with an FFFF FFFCh acknowledge value to RX*n*CP (note that there is no buffer descriptor in this case). Software may read RX*n*CP to determine if the interrupt was due to a commanded teardown. The read value is FFFF FFFCh, if the interrupt was due to a teardown command.

31.2.11.7 Receive Frame Classification

Received frames are proper (good) frames, if they are between 64 bytes and the value in the receive maximum length register (RXMAXLEN) bytes in length (inclusive) and contain no code, align, or CRC errors.

Received frames are long frames, if their frame count exceeds the value in RXMAXLEN. The RXMAXLEN reset (default) value is 5EEh (1518 in decimal). Long received frames are either oversized or jabber frames. Long frames with no errors are oversized frames; long frames with CRC, code, or alignment errors are jabber frames.

Received frames are short frames, if their frame count is less than 64 bytes. Short frames that address match and contain no errors are undersized frames; short frames with CRC, code, or alignment errors are fragment frames. If the frame length is less than or equal to 20, then the frame CRC is passed, regardless of whether the RXPASSCRC bit is set or cleared in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE).

A received long packet always contains RXMAXLEN number of bytes transferred to memory (if the RXCEFEN bit is set in RXMBPENABLE), regardless of the value of the RXPASSCRC bit. Following is an example with RXMAXLEN set to 1518:

- If the frame length is 1518, then the packet is not a long packet and there are 1514 or 1518 bytes transferred to memory depending on the value of the RXPASSCRC bit.
- If the frame length is 1519, there are 1518 bytes transferred to memory regardless of the RXPASSCRC bit value. The last three bytes are the first three CRC bytes.
- If the frame length is 1520, there are 1518 bytes transferred to memory regardless of the RXPASSCRC bit value. The last two bytes are the first two CRC bytes.
- If the frame length is 1521, there are 1518 bytes transferred to memory regardless of the RXPASSCRC bit value. The last byte is the first CRC byte.
- If the frame length is 1522, there are 1518 bytes transferred to memory. The last byte is the last data byte.

31.2.11.8 Promiscuous Receive Mode

When the promiscuous receive mode is enabled by setting the RXCAFEN bit in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE), nonaddress matching frames that would normally be filtered are transferred to the promiscuous channel. Address matching frames that would normally be filtered due to errors are transferred to the address match channel when the RXCAFEN and RXCEFEN bits in RXMBPENABLE are set. A frame is considered to be an address matching frame only if it is enabled to be received on a unicast, multicast, or broadcast channel. Frames received to disabled unicast, multicast, or broadcast channels are considered nonaddress matching.

MAC control frames address match only if the RXCMFEN bit in RXMBPENABLE is set. The RXCEFEN and RXCSFEN bits in RXMBPENABLE determine whether error frames are transferred to memory or not, but they do not determine whether error frames are address matching or not. Short frames are a special type of error frames.

A single channel is selected as the promiscuous channel by the RXPROMCH bit in RXMBPENABLE. The promiscuous receive mode is enabled by the RXCMFEN, RXCEFEN, RXCSFEN, and RXCAFEN bits in RXMBPENABLE. Table 31-7 shows the effects of the promiscuous enable bits. Proper frames are frames that are between 64 bytes and the value in the receive maximum length register (RXMAXLEN) bytes in length inclusive and contain no code, align, or CRC errors.

Table 31-7. Receive Frame Treatment Summary

Address Match	RXCAFEN	RXCEFEN	RXCMFEN	RXCSFEN	Receive Frame Treatment
0	0	X	X	X	No frames transferred.
0	1	0	0	0	Proper frames transferred to promiscuous channel.
0	1	0	0	1	Proper/undersized data frames transferred to promiscuous channel.
0	1	0	1	0	Proper data and control frames transferred to promiscuous channel.
0	1	0	1	1	Proper/undersized data and control frames transferred to promiscuous channel.
0	1	1	0	0	Proper/oversize/jabber/code/align/CRC data frames transferred to promiscuous channel. No control or undersized/fragment frames are transferred.
0	1	1	0	1	Proper/undersized/fragment/oversize/jabber/code/align/CRC data frames transferred to promiscuous channel. No control frames are transferred.
0	1	1	1	0	Proper/oversize/jabber/code/align/CRC data and control frames transferred to promiscuous channel. No undersized frames are transferred.
0	1	1	1	1	All nonaddress matching frames with and without errors transferred to promiscuous channel.
1	X	0	0	0	Proper data frames transferred to address match channel.
1	X	0	0	1	Proper/undersized data frames transferred to address match channel.
1	X	0	1	0	Proper data and control frames transferred to address match channel.
1	X	0	1	1	Proper/undersized data and control frames transferred to address match channel.
1	X	1	0	0	Proper/oversize/jabber/code/align/CRC data frames transferred to address match channel. No control or undersized frames are transferred.
1	X	1	0	1	Proper/oversize/jabber/fragment/undersized/code/align/CRC data frames transferred to address match channel. No control frames are transferred.
1	X	1	1	0	Proper/oversize/jabber/code/align/CRC data and control frames transferred to address match channel. No undersized/fragment frames are transferred.
1	X	1	1	1	All address matching frames with and without errors transferred to the address match channel

31.2.11.9 Receive Overrun

The types of receive overrun are:

- FIFO start of frame overrun (FIFO_SOF)
- FIFO middle of frame overrun (FIFO_MOF)
- DMA start of frame overrun (DMA_SOF)
- DMA middle of frame overrun (DMA_MOF)

The statistics counters used to track these types of receive overrun are:

- Receive start of frame overruns register (RXSOFOVERRUNS)
- Receive middle of frame overruns register (RXMOFOVERRUNS)
- Receive DMA overruns register (RXDMAOVERRUNS)

Start of frame overruns happen when there are no resources available when frame reception begins. Start of frame overruns increment the appropriate overrun statistic(s) and the frame is filtered.

Middle of frame overruns happen when there are some resources to start the frame reception, but the resources run out during frame reception. In normal operation, a frame that overruns after starting the frame reception is filtered and the appropriate statistic(s) are incremented; however, the RXCEFEN bit in the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE) affects overrun frame treatment. Table 31-8 shows how the overrun condition is handled for the middle of frame overrun.

Table 31-8. Middle of Frame Overrun Treatment

Address Match	RXCAFEN	RXCEFEN	Middle of Frame Overrun Treatment
0	0	X	Overrun frame filtered.
0	1	0	Overrun frame filtered.
0	1	1	As much frame data as possible is transferred to the promiscuous channel until overrun. The appropriate overrun statistic(s) is incremented and the OVERRUN and NOMATCH flags are set in the SOP buffer descriptor. Note that the RXMAXLEN number of bytes cannot be reached for an overrun to occur (it would be truncated and be a jabber or oversize).
1	X	0	Overrun frame filtered with the appropriate overrun statistic(s) incremented.
1	X	1	As much frame data as possible is transferred to the address match channel until overrun. The appropriate overrun statistic(s) is incremented and the OVERRUN flag is set in the SOP buffer descriptor. Note that the RXMAXLEN number of bytes cannot be reached for an overrun to occur (it would be truncated).

31.2.12 Packet Transmit Operation

The transmit DMA is an eight channel interface. Priority between the eight queues may be either fixed or round-robin as selected by the TXPTYPE bit in the MAC control register (MACCONTROL). If the priority type is fixed, then channel 7 has the highest priority and channel 0 has the lowest priority. Round-robin priority proceeds from channel 0 to channel 7.

31.2.12.1 Transmit DMA Host Configuration

To configure the transmit DMA for operation the host must perform:

- Write the MAC source address low bytes register (MACSRCADDRLO) and the MAC source address high bytes register (MACSRCADDRHI) (used for pause frames on transmit).
- Initialize the transmit channel *n* DMA head descriptor pointer registers (TX*n*HDP) to 0.
- Enable the desired transmit interrupts using the transmit interrupt mask set register (TXINTMASKSET) and the transmit interrupt mask clear register (TXINTMASKCLEAR).
- Set the appropriate configuration bits in the MAC control register (MACCONTROL).
- Setup the transmit channel(s) buffer descriptors in host memory.
- Enable the transmit DMA controller by setting the TXEN bit in the transmit control register (TXCONTROL).
- Write the appropriate TX*n*HDP with the pointer to the first descriptor to start transmit operations.

31.2.12.2 Transmit Channel Teardown

The host commands a transmit channel teardown by writing the channel number to the transmit teardown register (TXTEARDOWN). When a teardown command is issued to an enabled transmit channel, the following occurs:

- Any frame currently in transmission completes normally.
- The TDOWNCMPLT flag is set in the next SOP buffer descriptor in the chain, if there is one.
- The channel head descriptor pointer is cleared to 0.
- A transmit interrupt is issued to inform the host of the channel teardown.
- The corresponding transmit channel *n* completion pointer register (TX*n*CP) contains the value FFFF FFFCh.
- The host should acknowledge a teardown interrupt with an FFFF FFFCh acknowledge value.

Channel teardown may be commanded on any channel at any time. The host is informed of the teardown completion by the set teardown complete (TDOWNCMPLT) buffer descriptor bit. The EMAC does not clear any channel enables due to a teardown command. A teardown command to an inactive channel issues an interrupt that software should acknowledge with an FFFF FFFCh acknowledge value to TX*n*CP (note that there is no buffer descriptor in this case). Software may read the interrupt acknowledge location (TX*n*CP) to determine if the interrupt was due to a commanded teardown. The read value is FFFF FFFCh, if the interrupt was due to a teardown command.

31.2.13 Receive and Transmit Latency

The transmit and receive FIFOs each contain three 64-byte cells. The EMAC begins transmission of a packet on the wire after TXCELLTHRESH (configurable through the FIFO control register) cells, or a complete packet, are available in the FIFO.

Transmit underrun cannot occur for packet sizes of TXCELLTHRESH times 64 bytes (or less). For larger packet sizes, transmit underrun occurs if the memory latency is greater than the time required to transmit a 64-byte cell on the wire; this is 5.12 μs in 100 Mbps mode and 51.2 μs in 10 Mbps mode. The memory latency time includes all buffer descriptor reads for the entire cell data.

Receive overrun is prevented if the receive memory cell latency is less than the time required to transmit a 64-byte cell on the wire: 5.12 μs in 100 Mbps mode, or 51.2 μs in 10 Mbps mode. The latency time includes any required buffer descriptor reads for the cell data.

Latency to system’s internal and external RAM can be controlled through the use of the transfer node priority allocation register available at the device level. Latency to descriptor RAM is low because RAM is local to the EMAC, as it is part of the EMAC control module.

31.2.14 Transfer Node Priority

The device contains a chip-level master priority register that is used to set the priority of the transfer node used in issuing memory transfer requests to system memory.

Although the EMAC has internal FIFOs to help alleviate memory transfer arbitration problems, the average transfer rate of data read and written by the EMAC to internal or external processor memory must be at least that of the Ethernet wire rate. In addition, the internal FIFO system can not withstand a single memory latency event greater than the time it takes to fill or empty a TXCELLTHRESH number of internal 64 byte FIFO cells.

For 100 Mbps operation, these restrictions translate into the following rules:

- The short-term average, each 64-byte memory read/write request from the EMAC must be serviced in no more than 5.12 μs.
- Any single latency event in request servicing can be no longer than (5.12 × TXCELLTHRESH) μs.

31.2.15 Reset Considerations

31.2.15.1 Software Reset Considerations

The peripheral clock is controlled by the Global Clock Module (GCM), while the reset to the EMAC, MDIO and EMAC control module is controlled by the system module. See the Architecture chapter for more information on how to enable or disable the peripheral clock to the EMAC, MDIO and EMAC control module. For more on how the EMAC, MDIO, and EMAC control module are disabled or placed in reset at runtime, see Section 31.2.18.

Within the peripheral itself, the EMAC component of the Ethernet MAC peripheral can be placed in a reset state by writing to the soft reset register (SOFTRESET). Writing a 1 to the SOFTRESET bit causes the EMAC logic to be reset and the register values to be set to their default values. Software reset occurs when the receive and transmit DMA controllers are in an idle state to avoid locking up the configuration bus; it is the responsibility of the software to verify that there are no pending frames to be transferred. After writing a 1 to the SOFTRESET bit, it may be polled to determine if the reset has occurred. If a 1 is read, the reset has not yet occurred; if a 0 is read, then a reset has occurred.

After a software reset operation, all the EMAC registers need to be reinitialized for proper data transmission, including the FULLDUPLEX bit setting in the MAC control register (MACCONTROL).

Unlike the EMAC module, the MDIO and EMAC control modules cannot be placed in reset from a register inside their memory map.

31.2.15.2 Hardware Reset Considerations

When a hardware reset occurs, the EMAC peripheral has its register values reset and all the components return to their default state. After the hardware reset, the EMAC needs to be initialized before being able to resume its data transmission, as described in Section 31.2.16.

A hardware reset is the only means of recovering from the error interrupts (HOSTPEND), which are triggered by errors in packet buffer descriptors. Before doing a hardware reset, you should inspect the error codes in the MAC status register (MACSTATUS) that gives information about the type of software error that needs to be corrected. For detailed information on error interrupts, see Section 31.2.17.1.4.

31.2.16 Initialization

31.2.16.1 Enabling the EMAC/MDIO Peripheral

When the device is powered on, the EMAC peripheral becomes enabled as soon as the system reset is released, and the EMAC peripheral registers are set to their default values. The application software can configure the EMAC peripheral registers as required.

31.2.16.2 EMAC Control Module Initialization

The EMAC control module is used for global interrupt enables and to pace interrupts using 1ms time windows. There is also an 8K block of CPPI RAM local to the EMAC that is used to hold packet buffer descriptors.

Note that although the EMAC control module and the EMAC module have slightly different functions, in practice, the type of maintenance performed on the EMAC control module is more commonly conducted from the EMAC module software (as opposed to the MDIO module).

The initialization of the EMAC control module consists of two parts:

1. Configuration of the interrupt to the CPU.
2. Initialization of the EMAC control module:
   - Setting the interrupt pace counts using the EMAC control module registers INTCONTROL, C0RXIMAX, and C0TXIMAX
   - Initializing the EMAC and MDIO modules
   - Enabling interrupts in the EMAC control module using the EMAC control module interrupt control registers C0RXTHRESHEN, C0RXEN, C0TXEN, and C0MISCEN.

The process of mapping the EMAC interrupts to the CPU is done through the Vectored Interrupt Manager (VIM). Once the interrupt is mapped to a CPU interrupt, general masking and unmasking of interrupts (to control reentrancy) should be done at the chip level by manipulating the interrupt core enable mask registers.

31.2.16.3 MDIO Module Initialization

The MDIO module is used to initially configure and monitor one or more external PHY devices. Other than initializing the software state machine (details on this state machine can be found in the IEEE 802.3 standard), all that needs to be done for the MDIO module is to enable the MDIO engine and to configure the clock divider. To set the clock divider, supply an MDIO clock of 1 MHz. For example, if the peripheral clock is 50 MHz, the divider can be set to 50.

Both the state machine enable and the MDIO clock divider are controlled through the MDIO control register (CONTROL). If none of the potentially connected PHYs require the access preamble, the PREAMBLE bit in CONTROL can also be set to speed up PHY register access.

If the MDIO module is to operate on an interrupt basis, the interrupts can be enabled at this time using the MDIO user command complete interrupt mask set register (USERINTMASKSET) for register access and the MDIO user PHY select register (USERPHYSEL*n*) if a target PHY is already known.

Once the MDIO state machine has been initialized and enabled, it starts polling all 32 PHY addresses on the MDIO bus, looking for an active PHY. Since it can take up to 50 μs to read one register, it can be some time before the MDIO module provides an accurate representation of whether a PHY is available. Also, a PHY can take up to 3 seconds to negotiate a link. Thus, it is advisable to run the MDIO software off a time-based event rather than polling.

For more information on PHY control registers, see your PHY device documentation.

31.2.16.4 EMAC Module Initialization

The EMAC module is used to send and receive data packets over the network. This is done by maintaining up to eight transmit and receive descriptor queues. The EMAC module configuration must also be kept up-to-date based on PHY negotiation results returned from the MDIO module. Most of the work in developing an application or device driver for Ethernet is programming this module.

The following is the initialization procedure a device driver would follow to get the EMAC to the state where it is ready to receive and send Ethernet packets. Some of these steps are not necessary when performed immediately after device reset.

1. If enabled, clear the device interrupt enable bits in the EMAC control module interrupt control registers C0RXTHRESHEN, C0RXEN, C0TXEN, and C0MISCEN.
2. Clear the MAC control register (MACCONTROL), receive control register (RXCONTROL), and transmit control register (TXCONTROL) (not necessary immediately after reset).
3. Initialize all 16 header descriptor pointer registers (RX*n*HDP and TX*n*HDP) to 0.
4. Clear all 36 statistics registers by writing 0 (not necessary immediately after reset).
5. Setup the local Ethernet MAC address by programming the MAC index register (MACINDEX), MAC address high bytes register (MACADDRHI), and MAC address low bytes register (MACADDRLO). Be sure to program all eight MAC address registers - whether the receive channel is to be enabled or not. Duplicate the same MAC address across all unused channels. When using more than one receive channel, start with channel 0 and progress upwards.
6. If buffer flow control is to be enabled, initialize the receive channel *n* free buffer count registers (RX*n*FREEBUFFER), receive channel *n* flow control threshold register (RX*n*FLOWTHRESH), and receive filter low priority frame threshold register (RXFILTERLOWTHRESH).
7. Most device drivers open with no multicast addresses, so clear the MAC address hash registers (MACHASH1 and MACHASH2) to 0.
8. Write the receive buffer offset register (RXBUFFEROFFSET) value (typically zero).
9. Initially clear all unicast channels by writing FFh to the receive unicast clear register (RXUNICASTCLEAR). If unicast is desired, it can be enabled now by writing the receive unicast set register (RXUNICASTSET). Some drivers will default to unicast on device open while others will not.
10. Setup the receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE) with an initial configuration. The configuration is based on the current receive filter settings of the device driver. Some drivers may enable things like broadcast and multicast packets immediately, while others may not.
11. Set the appropriate configuration bits in MACCONTROL (do not set the GMIIEN bit yet).
12. Clear all unused channel interrupt bits by writing the receive interrupt mask clear register (RXINTMASKCLEAR) and the transmit interrupt mask clear register (TXINTMASKCLEAR).
13. Enable the receive and transmit channel interrupt bits in the receive interrupt mask set register (RXINTMASKSET) and the transmit interrupt mask set register (TXINTMASKSET) for the channels to be used, and enable the HOSTMASK and STATMASK bits using the MAC interrupt mask set register (MACINTMASKSET).
14. Initialize the receive and transmit descriptor list queues.
15. Prepare receive by writing a pointer to the head of the receive buffer descriptor list to RX*n*HDP.
16. Enable the receive and transmit DMA controllers by setting the RXEN bit in RXCONTROL and the TXEN bit in TXCONTROL. Then set the GMIIEN bit in MACCONTROL.
17. Enable the device interrupt in EMAC control module registers C0RXTHRESHEN, C0RXEN, C0TXEN, and C0MISCEN.

31.2.17 Interrupt Support

31.2.17.1 EMAC Module Interrupt Events and Requests

The EMAC module generates 26 interrupt events:

- TXPEND*n*: Transmit packet completion interrupt for transmit channels 0 through 7
- RXPEND*n*: Receive packet completion interrupt for receive channels 0 through 7
- RXTHRESHPEND*n*: Receive packet completion interrupt for receive channels 0 through 7 when flow control is enabled and the number of free buffers is below the threshold
- STATPEND: Statistics interrupt
- HOSTPEND: Host error interrupt

31.2.17.1.1 Transmit Packet Completion Interrupts

The transmit DMA engine has eight channels, with each channel having a corresponding interrupt (TXPEND*n*). The transmit interrupts are level interrupts that remain asserted until cleared by the CPU.

Each of the eight transmit channel interrupts may be individually enabled by setting the appropriate bit in the transmit interrupt mask set register (TXINTMASKSET) to 1. Each of the eight transmit channel interrupts may be individually disabled by clearing the appropriate bit by writing a 1 to the transmit interrupt mask clear register (TXINTMASKCLEAR). The raw and masked transmit interrupt status may be read by reading the transmit interrupt status (unmasked) register (TXINTSTATRAW) and the transmit interrupt status (masked) register (TXINTSTATMASKED), respectively.

When the EMAC completes the transmission of a packet, the EMAC issues an interrupt to the CPU (via the EMAC control module) when it writes the packet's last buffer descriptor address to the appropriate channel queue's transmit completion pointer located in the state RAM block. The interrupt is generated by the write when enabled by the interrupt mask, regardless of the value written.

Upon interrupt reception, the CPU processes one or more packets from the buffer chain and then acknowledges an interrupt by writing the address of the last buffer descriptor processed to the queue's associated transmit completion pointer in the transmit DMA state RAM.

The data written by the host (buffer descriptor address of the last processed buffer) is compared to the data in the register written by the EMAC port (address of last buffer descriptor used by the EMAC). If the two values are not equal (which means that the EMAC has transmitted more packets than the CPU has processed interrupts for), the transmit packet completion interrupt signal remains asserted. If the two values are equal (which means that the host has processed all packets that the EMAC has transferred), the pending interrupt is cleared. The value that the EMAC is expecting is found by reading the transmit channel *n* completion pointer register (TX*n*CP).

The EMAC write to the completion pointer actually stores the value in the state RAM. The CPU written value does not actually change the register value. The host written value is compared to the register content (which was written by the EMAC) and if the two values are equal then the interrupt is removed; otherwise, the interrupt remains asserted. The host may process multiple packets prior to acknowledging an interrupt, or the host may acknowledge interrupts for every packet.

The application software must acknowledge the EMAC control module after processing packets by writing the appropriate C0RX key to the EMAC End-Of-Interrupt Vector register (MACEOIVECTOR). See Section 31.5.12 for the acknowledge key values.

31.2.17.1.2 Receive Packet Completion Interrupts

The receive DMA engine has eight channels, which each channel having a corresponding interrupt (RXPEND*n*). The receive interrupts are level interrupts that remain asserted until cleared by the CPU.

Each of the eight receive channel interrupts may be individually enabled by setting the appropriate bit in the receive interrupt mask set register (RXINTMASKSET) to 1. Each of the eight receive channel interrupts may be individually disabled by clearing the appropriate bit by writing a 1 in the receive interrupt mask clear register (RXINTMASKCLEAR). The raw and masked receive interrupt status may be read by reading the receive interrupt status (unmasked) register (RXINTSTATRAW) and the receive interrupt status (masked) register (RXINTSTATMASKED), respectively.

When the EMAC completes a packet reception, the EMAC issues an interrupt to the CPU by writing the packet's last buffer descriptor address to the appropriate channel queue's receive completion pointer located in the state RAM block. The interrupt is generated by the write when enabled by the interrupt mask, regardless of the value written.

Upon interrupt reception, the CPU processes one or more packets from the buffer chain and then acknowledges one or more interrupt(s) by writing the address of the last buffer descriptor processed to the queue's associated receive completion pointer in the receive DMA state RAM.

The data written by the host (buffer descriptor address of the last processed buffer) is compared to the data in the register written by the EMAC (address of last buffer descriptor used by the EMAC). If the two values are not equal (which means that the EMAC has received more packets than the CPU has processed interrupts for), the receive packet completion interrupt signal remains asserted. If the two values are equal (which means that the host has processed all packets that the EMAC has received), the pending interrupt is de-asserted. The value that the EMAC is expecting is found by reading the receive channel *n* completion pointer register (RX*n*CP).

The EMAC write to the completion pointer actually stores the value in the state RAM. The CPU written value does not actually change the register value. The host written value is compared to the register content (which was written by the EMAC) and if the two values are equal then the interrupt is removed; otherwise, the interrupt remains asserted. The host may process multiple packets prior to acknowledging an interrupt, or the host may acknowledge interrupts for every packet.

The application software must acknowledge the EMAC control module after processing packets by writing the appropriate C0TX key to the EMAC End-Of-Interrupt Vector register (MACEOIVECTOR). See Section 31.5.12 for the acknowledge key values.

31.2.17.1.3 Statistics Interrupt

The statistics level interrupt (STATPEND) is issued when any statistics value is greater than or equal to 8000 0000h, if enabled by setting the STATMASK bit in the MAC interrupt mask set register (MACINTMASKSET) to 1. The statistics interrupt is removed by writing to decrement any statistics value greater than 8000 0000h. As long as the most-significant bit of any statistics value is set, the interrupt remains asserted.

The application software must acknowledge the EMAC control module after receiving statistics interrupts by writing the appropriate C0MISC key to the EMAC End-Of-Interrupt Vector register (MACEOIVECTOR). See Section 31.5.12 for the acknowledge key values.

31.2.17.1.4 Host Error Interrupt

The host error interrupt (HOSTPEND) is issued, if enabled, under error conditions dealing with the handling of buffer descriptors, detected during transmit or receive DMA transactions. The failure of the software application to supply properly formatted buffer descriptors results in this error. The error bit can only be cleared by resetting the EMAC module in hardware.

The host error interrupt is enabled by setting the HOSTMASK bit in the MAC interrupt mask set register (MACINTMASKSET) to 1. The host error interrupt is disabled by clearing the appropriate bit by writing a 1 in the MAC interrupt mask clear register (MACINTMASKCLEAR). The raw and masked host error interrupt status may be read by reading the MAC interrupt status (unmasked) register (MACINTSTATRAW) and the MAC interrupt status (masked) register (MACINTSTATMASKED), respectively.

The transmit host error conditions are:

- SOP error
- Ownership bit not set in SOP buffer
- Zero next buffer descriptor pointer with EOP
- Zero buffer pointer
- Zero buffer length
- Packet length error

The receive host error conditions are:

- Ownership bit not set in input buffer
- Zero buffer pointer

The application software must acknowledge the EMAC control module after receiving host error interrupts by writing the appropriate C0MISC key to the EMAC End-Of-Interrupt Vector (MACEOIVECTOR). See Section 31.5.12 for the acknowledge key values.

31.2.17.1.5 Receive Threshold Interrupts

Each of the eight receive channels have a corresponding receive threshold interrupt (RX*n*THRESHPEND). The receive threshold interrupts are level interrupts that remain asserted until the triggering condition is cleared by the host. Each of the eight threshold interrupts may be individually enabled by setting to 1 the appropriate bit in the RXINTMASKSET register. Each of the eight channel interrupts may be individually disabled by clearing to zero the appropriate bit by writing a 1 in the receive interrupt mask clear register (RXINTMASKCLEAR). The raw and masked interrupt receive interrupt status may be read by reading the receive interrupt status (unmasked) register (RXINTSTATRAW) and the receive interrupt status (masked) register (RXINTSTATMASKED), respectively.

An RX*n*THRESHPEND interrupt bit is asserted when enabled and when the channel's associated free buffer count (RX*n*FREEBUFFER) is less than or equal to the channel's associated flow control threshold register (RX*n*FLOWTHRESH). The receive threshold interrupts use the same free buffer count and threshold logic as does flow control, but the interrupts are independently enabled from flow control. The threshold interrupts are intended to give the host an indication that resources are running low for a particular channel(s).

The applications software must acknowledge the EMAC control module after receiving threshold interrupts by writing the appropriate C0RXTHRESH key to the EMAC End-Of-Interrupt Vector (MACEOIVECTOR). See Section 31.5.12 for the acknowledge key values.

31.2.17.2 MDIO Module Interrupt Events and Requests

The MDIO module generates two interrupt events:

- LINKINT0: Serial interface link change interrupt. Indicates a change in the state of the PHY link selected by the USERPHYSEL0 register
- USERINT0: Serial interface user command event complete interrupt selected by the USERACCESS0 register

31.2.17.2.1 Link Change Interrupt

The MDIO module asserts a link change interrupt (LINKINT0) if there is a change in the link state of the PHY corresponding to the address in the PHYADRMON bit in the MDIO register USERPHYSEL0, and if the LINKINTENB bit is also set in USERPHYSEL0. This interrupt event is also captured in the LINKINTRAW bit in the MDIO link status change interrupt register (LINKINTRAW). LINKINTRAW bits 0 and 1 correspond to USERPHYSEL0 and USERPHYSEL1, respectively.

When the interrupt is enabled and generated, the corresponding LINKINTMASKED bit is also set in the MDIO link status change interrupt register (LINKINTMASKED). The interrupt is cleared by writing back the same bit to LINKINTMASKED (write to clear).

The application software must acknowledge the EMAC control module after receiving MDIO interrupts by writing the appropriate C0MISC key to the EMAC End-Of-Interrupt Vector (MACEOIVECTOR). See Section 31.5.12 for the acknowledge key values.

31.2.17.2.2 User Access Completion Interrupt

When the GO bit in one of the MDIO register USERACCESS0 transitions from 1 to 0 (indicating completion of a user access) and the corresponding USERINTMASKSET bit in the MDIO user command complete interrupt mask set register (USERINTMASKSET) corresponding to USERACCESS0 is set, a user access completion interrupt (USERINT) is asserted. This interrupt event is also captured in the USERINTRAW bit in the MDIO user command complete interrupt register (USERINTRAW). USERINTRAW bits 0 and bit 1 correspond to USERACCESS0 and USERACCESS1, respectively.

When the interrupt is enabled and generated, the corresponding USERINTMASKED bit is also set in the MDIO user command complete interrupt register (USERINTMASKED). The interrupt is cleared by writing back the same bit to USERINTMASKED (write to clear).

The application software must acknowledge the EMAC control module after receiving MDIO interrupts by writing the appropriate C0MISC key to the EMAC End-Of-Interrupt Vector (MACEOIVECTOR). See Section 31.5.12 for the acknowledge key values.

31.2.17.3 Proper Interrupt Processing

All the interrupts signaled from the EMAC and MDIO modules are level driven, so if they remain active, their level remains constant; the CPU core may require edge- or pulse-triggered interrupts. In order to properly convert the level-driven interrupt signal to an edge- or pulse-triggered signal, the application software must make use of the interrupt control logic contained in the EMAC control module.

Section 31.2.7.3 discusses the interrupt control contained in the EMAC control module. For safe interrupt processing, upon entry to the ISR, the software application should disable interrupts using the EMAC control module registers C0RXTHRESHEN, C0RXEN, C0TXEN, C0MISCEN, and then reenable them upon leaving the ISR. If any interrupt signals are active at that time, this creates another rising edge on the interrupt signal going to the CPU interrupt controller, thus triggering another interrupt. The EMAC control module also uses the EMAC control module registers INTCONTROL, C0TXIMAX, and C0RXIMAX to implement interrupt pacing. The application software must acknowledge the EMAC control module by writing the appropriate key to the EMAC End-Of-Interrupt Vector (MACEOIVECTOR). See Section 31.5.12 for the acknowledge key values.

31.2.17.4 Interrupt Multiplexing

The EMAC control module combines different interrupt signals from both the EMAC and MDIO modules into four interrupt signals (C0RXTHRESHPULSE, C0RXPULSE, C0TXPULSE, C0MISCPULSE) that are routed to the Vectored Interrupt Manager (VIM). The VIM is capable of relaying all four interrupt signals to the CPU.

When an interrupt is generated, the reason for the interrupt can be read from the MAC input vector register (MACINVECTOR) located in the EMAC memory map. MACINVECTOR combines the status of the following 28 interrupt signals: TXPEND*n*, RXPEND*n*, RXTHRESHPEND*n*, STATPEND, HOSTPEND, LINKINT0, and USERINT0.

For more details on the interrupt mapping, see your device-specific datasheet and VIM chapter of the Technical Reference Manual.

31.2.18 Power Management

Each of the three main components of the EMAC peripheral can be placed in a reduced-power mode to conserve power during periods of low activity. The peripheral clock to the EMAC peripheral is controlled by the processor Global Clock Module (GCM). The GCM allows the application to enable or disable the peripheral clock to the EMAC peripheral.

The power conservation modes available for each of the three components of the EMAC/MDIO peripheral are:

- *Idle/Disabled state*. This mode stops the clocks going to the peripheral, and prevents all the register accesses. After reenabling the peripheral from this idle state, all the registers values prior to setting into the disabled state are restored, and data transmission can proceed. No reinitialization is required.
- *System reset.* The EMAC peripheral is reset by the system reset signal output from the System module. Refer to the "Architecture" chapter of the Technical Reference Manual to identify the causes of a system reset. Upon a system reset, the registers are reset to their default value. When powering-up after a system reset, all the EMAC submodules need to be reinitialized before any data transmission can happen.

For more information on the use of the GCM, see your device-specific Technical Reference Manual.

31.2.19 Emulation Considerations

EMAC emulation control is implemented for compatibility with other peripherals. The SOFT and FREE bits in the emulation control register (EMCONTROL) allow EMAC operation to be suspended.

When the emulation suspend state is entered, the EMAC stops processing receive and transmit frames at the next frame boundary. Any frame currently in reception or transmission is completed normally without suspension. For transmission, any complete or partial frame in the transmit cell FIFO is transmitted. For receive, frames that are detected by the EMAC after the suspend state is entered are ignored. No statistics are kept for ignored frames.

Table 31-9 shows how the SOFT and FREE bits affect the operation of the emulation suspend.

NOTE: Emulation suspend has not been tested.

Table 31-9. Emulation Control

SOFT	FREE	Description
0	0	Normal operation
1	0	Emulation suspend
X	1	Normal operation

31.3 EMAC Control Module Registers

Table 31-10 lists the memory-mapped registers for the EMAC control module. The base address for these registers is FCF7 8800h.

Table 31-10. EMAC Control Module Registers

Offset	Acronym	Register Description	Section
0h	REVID	EMAC Control Module Revision ID Register	Section 31.3.1
4h	SOFTRESET	EMAC Control Module Software Reset Register	Section 31.3.2
Ch	INTCONTROL	EMAC Control Module Interrupt Control Register	Section 31.3.3
10h	C0RXTHRESHEN	EMAC Control Module Receive Threshold Interrupt Enable Register	Section 31.3.4
14h	C0RXEN	EMAC Control Module Receive Interrupt Enable Register	Section 31.3.5
18h	C0TXEN	EMAC Control Module Transmit Interrupt Enable Register	Section 31.3.6
1Ch	C0MISCEN	EMAC Control Module Miscellaneous Interrupt Enable Register	Section 31.3.7
20h-3Ch	Reserved	Reserved	
40h	C0RXTHRESHSTAT	EMAC Control Module Receive Threshold Interrupt Status Register	Section 31.3.8
44h	C0RXSTAT	EMAC Control Module Receive Interrupt Status Register	Section 31.3.9
48h	C0TXSTAT	EMAC Control Module Transmit Interrupt Status Register	Section 31.3.10
4Ch	C0MISCSTAT	EMAC Control Module Miscellaneous Interrupt Status Register	Section 31.3.11
50h-6Ch	Reserved	Reserved	
70h	C0RXIMAX	EMAC Control Module Receive Interrupts Per Millisecond Register	Section 31.3.12
74h	C0TXIMAX	EMAC Control Module Transmit Interrupts Per Millisecond Register	Section 31.3.13
78h-84h	Reserved	Reserved	

31.3.1 EMAC Control Module Revision ID Register (REVID)

The EMAC control module revision ID register (REVID) is shown in Figure 31-15 and described in Table 31-11.

Figure 31-15. EMAC Control Module Revision ID Register (REVID)

31	0
REV	
R-4EC8 0100h	

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

Table 31-11. EMAC Control Module Revision ID Register (REVID) Field Descriptions

Bit	Field	Value	Description
31-0	REV		Identifies the EMAC Control Module revision.
		4EC8 0100h	Current revision of the EMAC Control Module.

31.3.2 EMAC Control Module Software Reset Register (SOFTRESET)

The EMAC Control Module Software Reset Register (SOFTRESET) is shown in Figure 31-16 and described in Table 31-12.

Figure 31-16. EMAC Control Module Software Reset Register (SOFTRESET)

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 31-12. EMAC Control Module Software Reset Register (SOFTRESET)

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	RESET		Software reset bit for the EMAC Control Module. Clears the interrupt status, control registers, and CPPI Ram on the clock cycle following a write of 1.
		0	No software reset.
		1	Perform a software reset.

31.3.3 EMAC Control Module Interrupt Control Register (INTCONTROL)

The EMAC control module interrupt control register (INTCONTROL) is shown in Figure 31-17 and described in Table 31-13. The settings in the INTCONTROL register are used in conjunction with the CnRXIMAX and CnTXIMAX registers.

Figure 31-17. EMAC Control Module Interrupt Control Register (INTCONTROL)

31-24
Reserved
R-0

23-18	17	16
Reserved	C0TXPACEEN	C0RXPACEEN
R-0	R/W-0	R/W-0

15-12	11-0
Reserved	INTPRESCALE
R-0	R/W-0

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

Table 31-13. EMAC Control Module Interrupt Control Register (INTCONTROL)

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17	C0TXPACEEN		Enable pacing for TX interrupt pulse generation
		0	Pacing for TX interrupts is disabled.
		1	Pacing for TX interrupts is enabled.
16	C0RXPACEEN		Enable pacing for RX interrupt pulse generation
		0	Pacing for RX interrupts is disabled.
		1	Pacing for RX interrupts is enabled.
15-12	Reserved	0	Reserved
11-0	INTPRESCALE	0-7FFh	Number of internal EMAC module reference clock periods within a 4 µs time window (see your device-specific data manual for information).

31.3.4 EMAC Control Module Receive Threshold Interrupt Enable Registers (C0RXTHRESHEN)

The EMAC control module receive threshold interrupt enable register (C0RXTHRESHEN) is shown in Figure 31-18 and described in Table 31-14.

Figure 31-18. EMAC Control Module Receive Threshold Interrupt Enable Register (C0RXTHRESHEN)

31–16
Reserved
R-0

15–8
Reserved
R-0

7	6	5	4	3	2	1	0
RXCH7 THRESHEN	RXCH6 THRESHEN	RXCH5 THRESHEN	RXCH4 THRESHEN	RXCH3 THRESHEN	RXCH2 THRESHEN	RXCH1 THRESHEN	RXCH0 THRESHEN
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 31-14. EMAC Control Module Receive Threshold Interrupt Enable Register (C0RXTHRESHEN)

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	RXCH7THRESHEN		Enable C0RXTHRESHPULSE interrupt generation for RX Channel 7
		0	C0RXTHRESHPULSE generation is disabled for RX Channel 7.
		1	C0RXTHRESHPULSE generation is enabled for RX Channel 7.
6	RXCH6THRESHEN		Enable C0RXTHRESHPULSE interrupt generation for RX Channel 6
		0	C0RXTHRESHPULSE generation is disabled for RX Channel 6.
		1	C0RXTHRESHPULSE generation is enabled for RX Channel 6.
5	RXCH5THRESHEN		Enable C0RXTHRESHPULSE interrupt generation for RX Channel 5
		0	C0RXTHRESHPULSE generation is disabled for RX Channel 5.
		1	C0RXTHRESHPULSE generation is enabled for RX Channel 5.
4	RXCH4THRESHEN		Enable C0RXTHRESHPULSE interrupt generation for RX Channel 4
		0	C0RXTHRESHPULSE generation is disabled for RX Channel 4.
		1	C0RXTHRESHPULSE generation is enabled for RX Channel 4.
3	RXCH3THRESHEN		Enable C0RXTHRESHPULSE interrupt generation for RX Channel 3
		0	C0RXTHRESHPULSE generation is disabled for RX Channel 3.
		1	C0RXTHRESHPULSE generation is enabled for RX Channel 3.
2	RXCH2THRESHEN		Enable C0RXTHRESHPULSE interrupt generation for RX Channel 2
		0	C0RXTHRESHPULSE generation is disabled for RX Channel 2.
		1	C0RXTHRESHPULSE generation is enabled for RX Channel 2.
1	RXCH1THRESHEN		Enable C0RXTHRESHPULSE interrupt generation for RX Channel 1
		0	C0RXTHRESHPULSE generation is disabled for RX Channel 1.
		1	C0RXTHRESHPULSE generation is enabled for RX Channel 1.
0	RXCH0THRESHEN		Enable C0RXTHRESHPULSE interrupt generation for RX Channel 0
		0	C0RXTHRESHPULSE generation is disabled for RX Channel 0.
		1	C0RXTHRESHPULSE generation is enabled for RX Channel 0.

31.3.5 EMAC Control Module Receive Interrupt Enable Registers (C0RXEN)

The EMAC control module receive interrupt enable register (C0RXEN) is shown in Figure 31-19 and described in Table 31-15

Figure 31-19. EMAC Control Module Receive Interrupt Enable Register (C0RXEN)

31	16
Reserved	
R-0	

15	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
RXCH7EN	RXCH6EN	RXCH5EN	RXCH4EN	RXCH3EN	RXCH2EN	RXCH1EN	RXCH0EN
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 31-15. EMAC Control Module Receive Interrupt Enable Register (C0RXEN)

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	RXCH7EN		Enable C0RXPULSE interrupt generation for RX Channel 7
		0	C0RXPULSE generation is disabled for RX Channel 7.
		1	C0RXPULSE generation is enabled for RX Channel 7.
6	RXCH6EN		Enable C0RXPULSE interrupt generation for RX Channel 6
		0	C0RXPULSE generation is disabled for RX Channel 6.
		1	C0RXPULSE generation is enabled for RX Channel 6.
5	RXCH5EN		Enable C0RXPULSE interrupt generation for RX Channel 5
		0	C0RXPULSE generation is disabled for RX Channel 5.
		1	C0RXPULSE generation is enabled for RX Channel 5.
4	RXCH4EN		Enable C0RXPULSE interrupt generation for RX Channel 4
		0	C0RXPULSE generation is disabled for RX Channel 4.
		1	C0RXPULSE generation is enabled for RX Channel 4.
3	RXCH3EN		Enable C0RXPULSE interrupt generation for RX Channel 3
		0	C0RXPULSE generation is disabled for RX Channel 3.
		1	C0RXPULSE generation is enabled for RX Channel 3.
2	RXCH2EN		Enable C0RXPULSE interrupt generation for RX Channel 2
		0	C0RXPULSE generation is disabled for RX Channel 2.
		1	C0RXPULSE generation is enabled for RX Channel 2.
1	RXCH1EN		Enable C0RXPULSE interrupt generation for RX Channel 1
		0	C0RXPULSE generation is disabled for RX Channel 1.
		1	C0RXPULSE generation is enabled for RX Channel 1.
0	RXCH0EN		Enable C0RXPULSE interrupt generation for RX Channel 0
		0	C0RXPULSE generation is disabled for RX Channel 0.
		1	C0RXPULSE generation is enabled for RX Channel 0.

31.3.6 EMAC Control Module Transmit Interrupt Enable Registers (C0TXEN)

The EMAC control module transmit interrupt enable register (C0TXEN) is shown in Figure 31-20 and described in Table 31-16

Figure 31-20. EMAC Control Module Transmit Interrupt Enable Register (C0TXEN)

31		16
	Reserved	
	R-0	

15		8
	Reserved	
	R-0	

7	6	5	4	3	2	1	0
TXCH7EN	TXCH6EN	TXCH5EN	TXCH4EN	TXCH3EN	TXCH2EN	TXCH1EN	TXCH0EN
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 31-16. EMAC Control Module Transmit Interrupt Enable Register (C0TXEN)

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	TXCH7EN		Enable C0TXPULSE interrupt generation for TX Channel 7
		0	C0TXPULSE generation is disabled for TX Channel 7.
		1	C0TXPULSE generation is enabled for TX Channel 7.
6	TXCH6EN		Enable C0TXPULSE interrupt generation for TX Channel 6
		0	C0TXPULSE generation is disabled for TX Channel 6.
		1	C0TXPULSE generation is enabled for TX Channel 6.
5	TXCH5EN		Enable C0TXPULSE interrupt generation for TX Channel 5
		0	C0TXPULSE generation is disabled for TX Channel 5.
		1	C0TXPULSE generation is enabled for TX Channel 5.
4	TXCH4EN		Enable C0TXPULSE interrupt generation for TX Channel 4
		0	C0TXPULSE generation is disabled for TX Channel 4.
		1	C0TXPULSE generation is enabled for TX Channel 4.
3	TXCH3EN		Enable C0TXPULSE interrupt generation for TX Channel 3
		0	C0TXPULSE generation is disabled for TX Channel 3.
		1	C0TXPULSE generation is enabled for TX Channel 3.
2	TXCH2EN		Enable C0TXPULSE interrupt generation for TX Channel 2
		0	C0TXPULSE generation is disabled for TX Channel 2.
		1	C0TXPULSE generation is enabled for TX Channel 2.
1	TXCH1EN		Enable C0TXPULSE interrupt generation for TX Channel 1
		0	C0TXPULSE generation is disabled for TX Channel 1.
		1	C0TXPULSE generation is enabled for TX Channel 1.
0	TXCH0EN		Enable C0TXPULSE interrupt generation for TX Channel 0
		0	C0TXPULSE generation is disabled for TX Channel 0.
		1	C0TXPULSE generation is enabled for TX Channel 0.

31.3.7 EMAC Control Module Miscellaneous Interrupt Enable Registers (C0MISCEN)

The EMAC control module miscellaneous interrupt enable register (C0MISCEN) is shown in Figure 31-21 and described in Table 31-17.

Figure 31-21. EMAC Control Module Miscellaneous Interrupt Enable Register (C0MISCEN)

31 - 16
Reserved
R-0

15 - 4	3	2	1	0
Reserved	STATPENDEN	HOSTPENDEN	LINKINT0EN	USERINT0EN
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 31-17. EMAC Control Module Miscellaneous Interrupt Enable Register (C0MISCEN)

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	STATPENDEN		Enable C0MISCPULSE interrupt generation when EMAC statistics interrupts are generated
		0	C0MISCPULSE generation is disabled for EMAC STATPEND interrupts.
		1	C0MISCPULSE generation is enabled for EMAC STATPEND interrupts.
2	HOSTPENDEN		Enable C0MISCPULSE interrupt generation when EMAC host interrupts are generated
		0	C0MISCPULSE generation is disabled for EMAC HOSTPEND interrupts.
		1	C0MISCPULSE generation is enabled for EMAC HOSTPEND interrupts.
1	LINKINT0EN		Enable C0MISCPULSE interrupt generation when MDIO LINKINT0 interrupts (corresponding to USERPHYSEL0) are generated
		0	C0MISCPULSE generation is disabled for MDIO LINKINT0 interrupts.
		1	C0MISCPULSE generation is enabled for MDIO LINKINT0 interrupts.
0	USERINT0EN		Enable C0MISCPULSE interrupt generation when MDIO USERINT0 interrupts (corresponding to USERACCESS0) are generated
		0	C0MISCPULSE generation is disabled for MDIO USERINT0.
		1	C0MISCPULSE generation is enabled for MDIO USERINT0.

31.3.8 EMAC Control Module Receive Threshold Interrupt Status Registers (C0RXTHRESHSTAT)

The EMAC control module receive threshold interrupt status register (C0RXTHRESHSTAT) is shown in Figure 31-22 and described in Table 31-18

Figure 31-22. EMAC Control Module Receive Threshold Interrupt Status Register (C0RXTHRESHSTAT)

31	16
Reserved	
R-0	

15	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
RXCH7THRESH STAT	RXCH6THRESH STAT	RXCH5THRESH STAT	RXCH4THRESH STAT	RXCH3THRESH STAT	RXCH2THRESH STAT	RXCH1THRESH STAT	RXCH0THRESH STAT
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 31-18. EMAC Control Module Receive Threshold Interrupt Status Register (C0RXTHRESHSTAT)

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	RXCH7THRESHSTAT		Interrupt status for RX Channel 7 masked by the C0RXTHRESHEN register
		0	RX Channel 7 does not satisfy conditions to generate a C0RXTHRESHPULSE interrupt.
		1	RX Channel 7 satisfies conditions to generate a C0RXTHRESHPULSE interrupt.
6	RXCH6THRESHSTAT		Interrupt status for RX Channel 6 masked by the C0RXTHRESHEN register
		0	RX Channel 6 does not satisfy conditions to generate a C0RXTHRESHPULSE interrupt.
		1	RX Channel 6 satisfies conditions to generate a C0RXTHRESHPULSE interrupt.
5	RXCH5THRESHSTAT		Interrupt status for RX Channel 5 masked by the C0RXTHRESHEN register
		0	RX Channel 5 does not satisfy conditions to generate a C0RXTHRESHPULSE interrupt.
		1	RX Channel 5 satisfies conditions to generate a C0RXTHRESHPULSE interrupt.
4	RXCH4THRESHSTAT		Interrupt status for RX Channel 4 masked by the C0RXTHRESHEN register
		0	RX Channel 4 does not satisfy conditions to generate a C0RXTHRESHPULSE interrupt.
		1	RX Channel 4 satisfies conditions to generate a C0RXTHRESHPULSE interrupt.
3	RXCH3THRESHSTAT		Interrupt status for RX Channel 3 masked by the C0RXTHRESHEN register
		0	RX Channel 3 does not satisfy conditions to generate a C0RXTHRESHPULSE interrupt.
		1	RX Channel 3 satisfies conditions to generate a C0RXTHRESHPULSE interrupt.
2	RXCH2THRESHSTAT		Interrupt status for RX Channel 2 masked by the C0RXTHRESHEN register
		0	RX Channel 2 does not satisfy conditions to generate a C0RXTHRESHPULSE interrupt.
		1	RX Channel 2 satisfies conditions to generate a C0RXTHRESHPULSE interrupt.
1	RXCH1THRESHSTAT		Interrupt status for RX Channel 1 masked by the C0RXTHRESHEN register
		0	RX Channel 1 does not satisfy conditions to generate a C0RXTHRESHPULSE interrupt.
		1	RX Channel 1 satisfies conditions to generate a C0RXTHRESHPULSE interrupt.
0	RXCH0THRESHSTAT		Interrupt status for RX Channel 0 masked by the C0RXTHRESHEN register
		0	RX Channel 0 does not satisfy conditions to generate a C0RXTHRESHPULSE interrupt.
		1	RX Channel 0 satisfies conditions to generate a C0RXTHRESHPULSE interrupt.

31.3.9 EMAC Control Module Receive Interrupt Status Registers (C0RXSTAT)

The EMAC control module receive interrupt status register (C0RXSTAT) is shown in Figure 31-23 and described in Table 31-19.

Figure 31-23. EMAC Control Module Receive Interrupt Status Register (C0RXSTAT)

31	16
Reserved	
R-0	

15	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
RXCH7STAT	RXCH6STAT	RXCH5STAT	RXCH4STAT	RXCH3STAT	RXCH2STAT	RXCH1STAT	RXCH0STAT
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 31-19. EMAC Control Module Receive Interrupt Status Register (C0RXSTAT)

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	RXCH7STAT		Interrupt status for RX Channel 7 masked by the C0RXEN register
		0	RX Channel 7 does not satisfy conditions to generate a C0RXPULSE interrupt.
		1	RX Channel 7 satisfies conditions to generate a C0RXPULSE interrupt.
6	RXCH6STAT		Interrupt status for RX Channel 6 masked by the C0RXEN register
		0	RX Channel 6 does not satisfy conditions to generate a C0RXPULSE interrupt.
		1	RX Channel 6 satisfies conditions to generate a C0RXPULSE interrupt.
5	RXCH5STAT		Interrupt status for RX Channel 5 masked by the C0RXEN register
		0	RX Channel 5 does not satisfy conditions to generate a C0RXPULSE interrupt.
		1	RX Channel 5 satisfies conditions to generate a C0RXPULSE interrupt.
4	RXCH4STAT		Interrupt status for RX Channel 4 masked by the C0RXEN register
		0	RX Channel 4 does not satisfy conditions to generate a C0RXPULSE interrupt.
		1	RX Channel 4 satisfies conditions to generate a C0RXPULSE interrupt.
3	RXCH3STAT		Interrupt status for RX Channel 3 masked by the C0RXEN register
		0	RX Channel 3 does not satisfy conditions to generate a C0RXPULSE interrupt.
		1	RX Channel 3 satisfies conditions to generate a C0RXPULSE interrupt.
2	RXCH2STAT		Interrupt status for RX Channel 2 masked by the C0RXEN register
		0	RX Channel 2 does not satisfy conditions to generate a C0RXPULSE interrupt.
		1	RX Channel 2 satisfies conditions to generate a C0RXPULSE interrupt.
1	RXCH1STAT		Interrupt status for RX Channel 1 masked by the C0RXEN register
		0	RX Channel 1 does not satisfy conditions to generate a C0RXPULSE interrupt.
		1	RX Channel 1 satisfies conditions to generate a C0RXPULSE interrupt.
0	RXCH0STAT		Interrupt status for RX Channel 0 masked by the C0RXEN register
		0	RX Channel 0 does not satisfy conditions to generate a C0RXPULSE interrupt.
		1	RX Channel 0 satisfies conditions to generate a C0RXPULSE interrupt.

31.3.10 EMAC Control Module Transmit Interrupt Status Registers (C0TXSTAT)

The EMAC control module transmit interrupt status register (C0TXSTAT) is shown in Figure 31-24 and described in Table 31-20.

Figure 31-24. EMAC Control Module Transmit Interrupt Status Register (C0TXSTAT)

31	16
Reserved	
R-0	

15	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
TXCH7STAT	TXCH6STAT	TXCH5STAT	TXCH4STAT	TXCH3STAT	TXCH2STAT	TXCH1STAT	TXCH0STAT
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 31-20. EMAC Control Module Transmit Interrupt Status Register (C0TXSTAT)

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	TXCH7STAT		Interrupt status for TX Channel 7 masked by the C0TXEN register
		0	TX Channel 7 does not satisfy conditions to generate a C0TXPULSE interrupt.
		1	TX Channel 7 satisfies conditions to generate a C0TXPULSE interrupt.
6	TXCH6STAT		Interrupt status for TX Channel 6 masked by the C0TXEN register
		0	TX Channel 6 does not satisfy conditions to generate a C0TXPULSE interrupt.
		1	TX Channel 6 satisfies conditions to generate a C0TXPULSE interrupt.
5	TXCH5STAT		Interrupt status for TX Channel 5 masked by the C0TXEN register
		0	TX Channel 5 does not satisfy conditions to generate a C0TXPULSE interrupt.
		1	TX Channel 5 satisfies conditions to generate a C0TXPULSE interrupt.
4	TXCH4STAT		Interrupt status for TX Channel 4 masked by the C0TXEN register
		0	TX Channel 4 does not satisfy conditions to generate a C0TXPULSE interrupt.
		1	TX Channel 4 satisfies conditions to generate a C0TXPULSE interrupt.
3	TXCH3STAT		Interrupt status for TX Channel 3 masked by the C0TXEN register
		0	TX Channel 3 does not satisfy conditions to generate a C0TXPULSE interrupt.
		1	TX Channel 3 satisfies conditions to generate a C0TXPULSE interrupt.
2	TXCH2STAT		Interrupt status for TX Channel 2 masked by the C0TXEN register
		0	TX Channel 2 does not satisfy conditions to generate a C0TXPULSE interrupt.
		1	TX Channel 2 satisfies conditions to generate a C0TXPULSE interrupt.
1	TXCH1STAT		Interrupt status for TX Channel 1 masked by the C0TXEN register
		0	TX Channel 1 does not satisfy conditions to generate a C0TXPULSE interrupt.
		1	TX Channel 1 satisfies conditions to generate a C0TXPULSE interrupt.
0	TXCH0STAT		Interrupt status for TX Channel 0 masked by the C0TXEN register
		0	TX Channel 0 does not satisfy conditions to generate a C0TXPULSE interrupt.
		1	TX Channel 0 satisfies conditions to generate a C0TXPULSE interrupt.

31.3.11 EMAC Control Module Miscellaneous Interrupt Status Registers (C0MISCSTAT)

The EMAC control module miscellaneous interrupt status register (C0MISCSTAT) is shown in Figure 31-25 and described in Table 31-21

Figure 31-25. EMAC Control Module Miscellaneous Interrupt Status Register (C0MISCSTAT)

31–16
Reserved
R-0

15–4	3	2	1	0
Reserved	STATPEND STAT	HOSTPEND STAT	LINKINT0 STAT	USERINT0 STAT
R-0	R-0	R-0	R-0	R-0

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

Table 31-21. EMAC Control Module Miscellaneous Interrupt Status Register (C0MISCSTAT)

Bit	Field	Value	Description
31-4	Reserved	0	Reserved
3	STATPENDSTAT		Interrupt status for EMAC STATPEND masked by the C0MISCEN register
		0	EMAC STATPEND does not satisfy conditions to generate a C0MISCPULSE interrupt.
		1	EMAC STATPEND satisfies conditions to generate a C0MISCPULSE interrupt.
2	HOSTPENDSTAT		Interrupt status for EMAC HOSTPEND masked by the C0MISCEN register
		0	EMAC HOSTPEND does not satisfy conditions to generate a C0MISCPULSE interrupt.
		1	EMAC HOSTPEND satisfies conditions to generate a C0MISCPULSE interrupt.
1	LINKINT0STAT		Interrupt status for MDIO LINKINT0 masked by the C0MISCEN register
		0	MDIO LINKINT0 does not satisfy conditions to generate a C0MISCPULSE interrupt.
		1	MDIO LINKINT0 satisfies conditions to generate a C0MISCPULSE interrupt.
0	USERINT0STAT		Interrupt status for MDIO USERINT0 masked by the C0MISCEN register
		0	MDIO USERINT0 does not satisfy conditions to generate a C0MISCPULSE interrupt.
		1	MDIO USERINT0 satisfies conditions to generate a C0MISCPULSE interrupt.

31.3.12 EMAC Control Module Receive Interrupts Per Millisecond Registers (C0RXIMAX)

The EMAC control module receive interrupts per millisecond register (C0RXIMAX) is shown in Figure 31-26 and described in Table 31-22.

Figure 31-26. EMAC Control Module Receive Interrupts Per Millisecond Register (C0RXIMAX)

31 – 16
Reserved
R-0

15 – 6	5 – 0
Reserved	RXIMAX
R-0	R/W-0

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

Table 31-22. EMAC Control Module Receive Interrupts Per Millisecond Register (C0RXIMAX)

Bit	Field	Value	Description
31-6	Reserved	0	Reserved
5-0	RXIMAX	2h-3Fh	RXIMAX is the desired number of C0RXPULSE interrupts generated per millisecond when C0RXPACEEN is enabled in INTCONTROL.

The pacing mechanism can be described by the following pseudo-code:

```
while(1) {
    interrupt_count = 0;
    /* Count interrupts over a 1ms window */
    for(i = 0; i < INTCONTROL[INTPRESCALE]*250; i++) {
        interrupt_count += NEW_INTERRUPT_EVENTS();
        if(i < INTCONTROL[INTPRESCALE]*pace_counter)
            BLOCK_EMAC_INTERRUPTS();
        else
            ALLOW_EMAC_INTERRUPTS();
    }
    ALLOW_EMAC_INTERRUPTS();
    if(interrupt_count > 2*RXIMAX)
        pace_counter = 255;
    else if(interrupt_count > 1.5*RXIMAX)
        pace_counter = previous_pace_counter*2 + 1;
    else if(interrupt_count > 1.0*RXIMAX)
        pace_counter = previous_pace_counter + 1;
    else if(interrupt_count > 0.5*RXIMAX)
        pace_counter = previous_pace_counter - 1;
    else if(interrupt_count != 0)
        pace_counter = previous_pace_counter/2;
    else
        pace_counter = 0;
    previous_pace_counter = pace_counter;
}
```

31.3.13 EMAC Control Module Transmit Interrupts Per Millisecond Registers (C0TXIMAX)

The EMAC control module transmit interrupts per millisecond register (C0TXIMAX) is shown in Figure 31-27 and described in Table 31-23.

Figure 31-27. EMAC Control Module Transmit Interrupts Per Millisecond Register (C0TXIMAX)

31	16
Reserved	
R-0	

15	6	5	0
Reserved		TXIMAX	
R-0		R/W-0	

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

Table 31-23. EMAC Control Module Transmit Interrupts Per Millisecond Register (C0TXIMAX)

Bit	Field	Value	Description
31-6	Reserved	0	Reserved
5-0	TXIMAX	2h-3Fh	TXIMAX is the desired number of C0TXPULSE interrupts generated per millisecond when C0TXPACEEN is enabled in INTCONTROL.

The pacing mechanism can be described by the following pseudo-code:

```
while(1) {
    interrupt_count = 0;

    /* Count interrupts over a 1ms window */
    for(i = 0; i < INTCONTROL[INTPRESCALE]*250; i++) {
        interrupt_count += NEW_INTERRUPT_EVENTS();

        if(i < INTCONTROL[INTPRESCALE]*pace_counter)
            BLOCK_EMAC_INTERRUPTS();
        else
            ALLOW_EMAC_INTERRUPTS();
    }

    ALLOW_EMAC_INTERRUPTS();

    if(interrupt_count > 2*TXIMAX)
        pace_counter = 255;
    else if(interrupt_count > 1.5*TXIMAX)
        pace_counter = previous_pace_counter*2 + 1;
    else if(interrupt_count > 1.0*TXIMAX)
        pace_counter = previous_pace_counter + 1;
    else if(interrupt_count > 0.5*TXIMAX)
        pace_counter = previous_pace_counter - 1;
    else if(interrupt_count != 0)
        pace_counter = previous_pace_counter/2;
    else
        pace_counter = 0;

    previous_pace_counter = pace_counter;
}
```

31.4 MDIO Registers

Table 31-24 lists the memory-mapped registers for the MDIO module. The base address for these registers is FCF7 8900h.

Table 31-24. Management Data Input/Output (MDIO) Registers

Offset	Acronym	Register Description	Section
0h	REVID	MDIO Revision ID Register	Section 31.4.1
4h	CONTROL	MDIO Control Register	Section 31.4.2
8h	ALIVE	PHY Alive Status register	Section 31.4.3
Ch	LINK	PHY Link Status Register	Section 31.4.4
10h	LINKINTRAW	MDIO Link Status Change Interrupt (Unmasked) Register	Section 31.4.5
14h	LINKINTMASKED	MDIO Link Status Change Interrupt (Masked) Register	Section 31.4.6
20h	USERINTRAW	MDIO User Command Complete Interrupt (Unmasked) Register	Section 31.4.7
24h	USERINTMASKED	MDIO User Command Complete Interrupt (Masked) Register	Section 31.4.8
28h	USERINTMASKSET	MDIO User Command Complete Interrupt Mask Set Register	Section 31.4.9
2Ch	USERINTMASKCLEAR	MDIO User Command Complete Interrupt Mask Clear Register	Section 31.4.10
80h	USERACCESS0	MDIO User Access Register 0	Section 31.4.11
84h	USERPHYSEL0	MDIO User PHY Select Register 0	Section 31.4.12
88h	USERACCESS1	MDIO User Access Register 1	Section 31.4.13
8Ch	USERPHYSEL1	MDIO User PHY Select Register 1	Section 31.4.14

31.4.1 *MDIO Revision ID Register (REVID)*

The MDIO revision ID register (REVID) is shown in Figure 31-28 and described in Table 31-25.

Figure 31-28. MDIO Revision ID Register (REVID)

31	0
REV	
R-0007 0105h	

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

Table 31-25. MDIO Revision ID Register (REVID) Field Descriptions

Bit	Field	Value	Description
31-0	REV		Identifies the MDIO Module revision.
		0007 0105h	Current revision of the MDIO Module.

31.4.2 MDIO Control Register (CONTROL)

The MDIO control register (CONTROL) is shown in Figure 31-29 and described in Table 31-26.

Figure 31-29. MDIO Control Register (CONTROL)

31	30	29	28 24	23 21	20	19	18	17 16
IDLE	ENABLE	Rsvd	HIGHEST_USER_CHANNEL	Reserved	PREAMBLE	FAULT	FAULTENB	Reserved
R-1	R/W-0	R-0	R-1	R-0	R/W-0	R/W1C-0	R/W-0	R-0

15 0
CLKDIV
R/W-FFh

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing a 0 has no effect); -*n* = value after reset

Table 31-26. MDIO Control Register (CONTROL) Field Descriptions

Bit	Field	Value	Description
31	IDLE		State machine IDLE status bit.
		0	State machine is not in idle state.
		1	State machine is in idle state.
30	ENABLE		State machine enable control bit. If the MDIO state machine is active at the time it is disabled, it will complete the current operation before halting and setting the idle bit.
		0	Disables the MDIO state machine.
		1	Enable the MDIO state machine.
29	Reserved	0	Reserved
28-24	HIGHEST_USER_CHANNEL	0-1Fh	Highest user channel that is available in the module. It is currently set to 1. This implies that MDIOUserAccess1 is the highest available user access channel.
23-21	Reserved	0	Reserved
20	PREAMBLE		Preamble disable
		0	Standard MDIO preamble is used.
		1	Disables this device from sending MDIO frame preambles.
19	FAULT		Fault indicator. This bit is set to 1 if the MDIO pins fail to read back what the device is driving onto them. This indicates a physical layer fault and the module state machine is reset. Writing a 1 to this bit clears this bit, writing a 0 has no effect.
		0	No failure
		1	Physical layer fault; the MDIO state machine is reset.
18	FAULTENB		Fault detect enable. This bit has to be set to 1 to enable the physical layer fault detection.
		0	Disables the physical layer fault detection.
		1	Enables the physical layer fault detection.
17-16	Reserved	0	Reserved
15-0	CLKDIV	0-FFFFh	Clock Divider bits. This field specifies the division ratio between the peripheral clock and the frequency of MDIO_CLK. MDIO_CLK is disabled when CLKDIV is cleared to 0. MDIO_CLK frequency = peripheral clock frequency/(CLKDIV + 1).

31.4.3 PHY Acknowledge Status Register (ALIVE)

The PHY acknowledge status register (ALIVE) is shown in Figure 31-30 and described in Table 31-27.

Figure 31-30. PHY Acknowledge Status Register (ALIVE)

31	0
ALIVE	
R/W1C-0	

LEGEND: R/W = Read/Write; W1C = Write 1 to clear (writing a 0 has no effect); -*n* = value after reset

Table 31-27. PHY Acknowledge Status Register (ALIVE) Field Descriptions

Bit	Field	Value	Description
31-0	ALIVE		MDIO Alive bits. Each of the 32 bits of this register is set if the most recent access to the PHY with address corresponding to the register bit number was acknowledged by the PHY; the bit is reset if the PHY fails to acknowledge the access. Both the user and polling accesses to a PHY will cause the corresponding ALIVE bit to be updated. The ALIVE bits are only meant to be used to give an indication of the presence or not of a PHY with the corresponding address. Writing a 1 to any bit will clear it, writing a 0 has no effect.
		0	The PHY fails to acknowledge the access.
		1	The most recent access to the PHY with an address corresponding to the register bit number was acknowledged by the PHY.

31.4.4 PHY Link Status Register (LINK)

The PHY link status register (LINK) is shown in Figure 31-31 and described in Table 31-28.

Figure 31-31. PHY Link Status Register (LINK)

31	0
LINK	
R-0	

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

Table 31-28. PHY Link Status Register (LINK) Field Descriptions

Bit	Field	Value	Description
31-0	LINK		MDIO Link state bits. This register is updated after a read of the generic status register of a PHY. The bit is set if the PHY with the corresponding address has link and the PHY acknowledges the read transaction. The bit is reset if the PHY indicates it does not have link or fails to acknowledge the read transaction. Writes to the register have no effect.
		0	The PHY indicates it does not have a link or fails to acknowledge the read transaction.
		1	The PHY with the corresponding address has a link and the PHY acknowledges the read transaction.

31.4.5 MDIO Link Status Change Interrupt (Unmasked) Register (LINKINTRAW)

The MDIO link status change interrupt (unmasked) register (LINKINTRAW) is shown in Figure 31-32 and described in Table 31-29.

Figure 31-32. MDIO Link Status Change Interrupt (Unmasked) Register (LINKINTRAW)

31–16
Reserved
R-0

15–2	1	0
Reserved	USERPHY1	USERPHY0
R-0	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing a 0 has no effect); -*n* = value after reset

Table 31-29. MDIO Link Status Change Interrupt (Unmasked) Register (LINKINTRAW) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	USERPHY1		MDIO Link change event, raw value. When asserted, the bit indicates that there was an MDIO link change event (that is, change in the LINK register) corresponding to the PHY address in USERPHYSEL1. Writing a 1 will clear the event, writing a 0 has no effect.
		0	No MDIO link change event.
		1	An MDIO link change event (change in the LINK register) corresponding to the PHY address in MDIO user PHY select register USERPHYSEL1.
0	USERPHY0		MDIO Link change event, raw value. When asserted, the bit indicates that there was an MDIO link change event (that is, change in the LINK register) corresponding to the PHY address in USERPHYSEL0. Writing a 1 will clear the event, writing a 0 has no effect.
		0	No MDIO link change event.
		1	An MDIO link change event (change in the LINK register) corresponding to the PHY address in MDIO user PHY select register USERPHYSEL0.

31.4.6 MDIO Link Status Change Interrupt (Masked) Register (LINKINTMASKED)

The MDIO link status change interrupt (masked) register (LINKINTMASKED) is shown in Figure 31-33 and described in Table 31-30.

Figure 31-33. MDIO Link Status Change Interrupt (Masked) Register (LINKINTMASKED)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		USERPHY1	USERPHY0
R-0		R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing a 0 has no effect); -*n* = value after reset

Table 31-30. MDIO Link Status Change Interrupt (Masked) Register (LINKINTMASKED) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	USERPHY1		MDIO Link change interrupt, masked value. When asserted, the bit indicates that there was an MDIO link change event (that is, change in the LINK register) corresponding to the PHY address in USERPHYSEL1 and the corresponding LINKINTENB bit was set. Writing a 1 will clear the event, writing a 0 has no effect.
		0	No MDIO link change event.
		1	An MDIO link change event (change in the LINK register) corresponding to the PHY address in MDIO user PHY select register USERPHYSEL1 and the LINKINTENB bit in USERPHYSEL1 is set to 1.
0	USERPHY0		MDIO Link change interrupt, masked value. When asserted, the bit indicates that there was an MDIO link change event (that is, change in the LINK register) corresponding to the PHY address in USERPHYSEL0 and the corresponding LINKINTENB bit was set. Writing a 1 will clear the event, writing a 0 has no effect.
		0	No MDIO link change event.
		1	An MDIO link change event (change in the LINK register) corresponding to the PHY address in MDIO user PHY select register USERPHYSEL0 and the LINKINTENB bit in USERPHYSEL0 is set to 1.

31.4.7 MDIO User Command Complete Interrupt (Unmasked) Register (USERINTRAW)

The MDIO user command complete interrupt (unmasked) register (USERINTRAW) is shown in Figure 31-34 and described in Table 31-31.

Figure 31-34. MDIO User Command Complete Interrupt (Unmasked) Register (USERINTRAW)

31 … 16
Reserved
R-0

15 … 2	1	0
Reserved	USERACCESS1	USERACCESS0
R-0	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing a 0 has no effect); -*n* = value after reset

Table 31-31. MDIO User Command Complete Interrupt (Unmasked) Register (USERINTRAW) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	USERACCESS1		MDIO User command complete event bit. When asserted, the bit indicates that the previously scheduled PHY read or write command using the USERACCESS1 register has completed. Writing a 1 will clear the event, writing a 0 has no effect.
		0	No MDIO user command complete event.
		1	The previously scheduled PHY read or write command using MDIO user access register USERACCESS1 has completed.
0	USERACCESS0		MDIO User command complete event bit. When asserted, the bit indicates that the previously scheduled PHY read or write command using the USERACCESS0 register has completed. Writing a 1 will clear the event, writing a 0 has no effect.
		0	No MDIO user command complete event.
		1	The previously scheduled PHY read or write command using MDIO user access register USERACCESS0 has completed.

31.4.8 MDIO User Command Complete Interrupt (Masked) Register (USERINTMASKED)

The MDIO user command complete interrupt (masked) register (USERINTMASKED) is shown in Figure 31-35 and described in Table 31-32.

Figure 31-35. MDIO User Command Complete Interrupt (Masked) Register (USERINTMASKED)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		USERACCESS1	USERACCESS0
R-0		R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing a 0 has no effect); -*n* = value after reset

Table 31-32. MDIO User Command Complete Interrupt (Masked) Register (USERINTMASKED) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	USERACCESS1		Masked value of MDIO User command complete interrupt. When asserted, The bit indicates that the previously scheduled PHY read or write command using that particular USERACCESS1 register has completed. Writing a 1 will clear the interrupt, writing a 0 has no effect.
		0	No MDIO user command complete event.
		1	The previously scheduled PHY read or write command using MDIO user access register USERACCESS1 has completed and the corresponding bit in USERINTMASKSET is set to 1.
0	USERACCESS0		Masked value of MDIO User command complete interrupt. When asserted, The bit indicates that the previously scheduled PHY read or write command using that particular USERACCESS0 register has completed. Writing a 1 will clear the interrupt, writing a 0 has no effect.
		0	No MDIO user command complete event.
		1	The previously scheduled PHY read or write command using MDIO user access register USERACCESS0 has completed and the corresponding bit in USERINTMASKSET is set to 1.

31.4.9 MDIO User Command Complete Interrupt Mask Set Register (USERINTMASKSET)

The MDIO user command complete interrupt mask set register (USERINTMASKSET) is shown in Figure 31-36 and described in Table 31-33.

Figure 31-36. MDIO User Command Complete Interrupt Mask Set Register (USERINTMASKSET)

31 - 16
Reserved
R-0

15 - 2	1	0
Reserved	USERACCESS1	USERACCESS0
R-0	R/W-0	R/W-0

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

Table 31-33. MDIO User Command Complete Interrupt Mask Set Register (USERINTMASKSET) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	USERACCESS1		MDIO user interrupt mask set for USERINTMASKED[1]. Setting a bit to 1 will enable MDIO user command complete interrupts for the USERACCESS1 register. MDIO user interrupt for USERACCESS1 is disabled if the corresponding bit is 0. Writing a 0 to this bit has no effect.
		0	MDIO user command complete interrupts for the MDIO user access register USERACCESS1 is disabled.
		1	MDIO user command complete interrupts for the MDIO user access register USERACCESS1 is enabled.
0	USERACCESS0		MDIO user interrupt mask set for USERINTMASKED[0]. Setting a bit to 1 will enable MDIO user command complete interrupts for the USERACCESS0 register. MDIO user interrupt for USERACCESS0 is disabled if the corresponding bit is 0. Writing a 0 to this bit has no effect.
		0	MDIO user command complete interrupts for the MDIO user access register USERACCESS0 is disabled.
		1	MDIO user command complete interrupts for the MDIO user access register USERACCESS0 is enabled.

31.4.10 MDIO User Command Complete Interrupt Mask Clear Register (USERINTMASKCLEAR)

The MDIO user command complete interrupt mask clear register (USERINTMASKCLEAR) is shown in Figure 31-37 and described in Table 31-34.

Figure 31-37. MDIO User Command Complete Interrupt Mask Clear Register (USERINTMASKCLEAR)

<table>
<tr><td>31</td><td>16</td></tr>
<tr><td colspan="2">Reserved</td></tr>
<tr><td colspan="2">R-0</td></tr>
</table>

<table>
<tr><td>15</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">Reserved</td><td>USERACCESS1</td><td>USERACCESS0</td></tr>
<tr><td colspan="2">R-0</td><td>R/W-0</td><td>R/W-0</td></tr>
</table>

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

Table 31-34. MDIO User Command Complete Interrupt Mask Clear Register (USERINTMASKCLEAR) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	USERACCESS1		MDIO user command complete interrupt mask clear for USERINTMASKED[1]. Setting the bit to 1 will disable further user command complete interrupts for USERACCESS1. Writing a 0 to this bit has no effect.
		0	MDIO user command complete interrupts for the MDIO user access register USERACCESS1 is enabled.
		1	MDIO user command complete interrupts for the MDIO user access register USERACCESS1 is disabled.
0	USERACCESS0		MDIO user command complete interrupt mask clear for USERINTMASKED[0]. Setting the bit to 1 will disable further user command complete interrupts for USERACCESS0. Writing a 0 to this bit has no effect.
		0	MDIO user command complete interrupts for the MDIO user access register USERACCESS0 is enabled.
		1	MDIO user command complete interrupts for the MDIO user access register USERACCESS0 is disabled.

31.4.11 MDIO User Access Register 0 (USERACCESS0)

The MDIO user access register 0 (USERACCESS0) is shown in Figure 31-38 and described in Table 31-35.

Figure 31-38. MDIO User Access Register 0 (USERACCESS0)

31	30	29	28 26	25 21	20 16
GO	WRITE	ACK	Reserved	REGADR	PHYADR
R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0

15 0
DATA
R/W-0

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

Table 31-35. MDIO User Access Register 0 (USERACCESS0) Field Descriptions

Bit	Field	Value	Description
31	GO	0-1	Go bit. Writing a 1 to this bit causes the MDIO state machine to perform an MDIO access when it is convenient for it to do so; this is not an instantaneous process. Writing a 0 to this bit has no effect. This bit is writeable only if the MDIO state machine is enabled. This bit will self clear when the requested access has been completed. Any writes to USERACCESS0 are blocked when the GO bit is 1.
30	WRITE		Write enable bit. Setting this bit to 1 causes the MDIO transaction to be a register write; otherwise, it is a register read.
		0	The user command is a read operation.
		1	The user command is a write operation.
29	ACK	0-1	Acknowledge bit. This bit is set if the PHY acknowledged the read transaction.
28-26	Reserved	0	Reserved
25-21	REGADR	0-1Fh	Register address bits. This field specifies the PHY register to be accessed for this transaction.
20-16	PHYADR	0-1Fh	PHY address bits. This field specifies the PHY to be accessed for this transaction.
15-0	DATA	0-FFFFh	User data bits. These bits specify the data value read from or to be written to the specified PHY register.

31.4.12 MDIO User PHY Select Register 0 (USERPHYSEL0)

The MDIO user PHY select register 0 (USERPHYSEL0) is shown in Figure 31-39 and described in Table 31-36.

Figure 31-39. MDIO User PHY Select Register 0 (USERPHYSEL0)

31 - 16
Reserved
R-0

15 - 8	7	6	5	4 - 0
Reserved	LINKSEL	LINKINTENB	Rsvd	PHYADRMON
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 31-36. MDIO User PHY Select Register 0 (USERPHYSEL0) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	LINKSEL		Link status determination select bit. Default value is 0, which implies that the link status is determined by the MDIO state machine. This is the only option supported on this device.
		0	The link status is determined by the MDIO state machine.
		1	Not supported.
6	LINKINTENB		Link change interrupt enable. Set to 1 to enable link change interrupts for PHY address specified in PHYADRMON. Link change interrupts are disabled if this bit is cleared to 0.
		0	Link change interrupts are disabled.
		1	Link change interrupts for PHY address specified in PHYADRMON bits are enabled.
5	Reserved	0	Reserved
4-0	PHYADRMON	0-1Fh	PHY address whose link status is to be monitored.

31.4.13 MDIO User Access Register 1 (USERACCESS1)

The MDIO user access register 1 (USERACCESS1) is shown in Figure 31-40 and described in Table 31-37.

Figure 31-40. MDIO User Access Register 1 (USERACCESS1)

31	30	29	28 26	25 21	20 16
GO	WRITE	ACK	Reserved	REGADR	PHYADR
R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0

15 0
DATA
R/W-0

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

Table 31-37. MDIO User Access Register 1 (USERACCESS1) Field Descriptions

Bit	Field	Value	Description
31	GO	0-1	Go bit. Writing 1 to this bit causes the MDIO state machine to perform an MDIO access when it is convenient for it to do so; this is not an instantaneous process. Writing 0 to this bit has no effect. This bit is writeable only if the MDIO state machine is enabled. This bit will self clear when the requested access has been completed. Any writes to USERACCESS1 are blocked when the GO bit is 1.
30	WRITE		Write enable bit. Setting this bit to 1 causes the MDIO transaction to be a register write; otherwise, it is a register read.
		0	The user command is a read operation.
		1	The user command is a write operation.
29	ACK	0-1	Acknowledge bit. This bit is set if the PHY acknowledged the read transaction.
28-26	Reserved	0	Reserved
25-21	REGADR	0-1Fh	Register address bits. This field specifies the PHY register to be accessed for this transaction.
20-16	PHYADR	0-1Fh	PHY address bits. This field specifies the PHY to be accessed for this transaction.
15-0	DATA	0-FFFFh	User data bits. These bits specify the data value read from or to be written to the specified PHY register.

31.4.14 MDIO User PHY Select Register 1 (USERPHYSEL1)

The MDIO user PHY select register 1 (USERPHYSEL1) is shown in Figure 31-41 and described in Table 31-38.

Figure 31-41. MDIO User PHY Select Register 1 (USERPHYSEL1)

31 16
Reserved
R-0

15 8	7	6	5	4 0
Reserved	LINKSEL	LINKINTENB	Rsvd	PHYADRMON
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 31-38. MDIO User PHY Select Register 1 (USERPHYSEL1) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	LINKSEL		Link status determination select bit. Default value is 0, which implies that the link status is determined by the MDIO state machine. This is the only option supported on this device.
		0	The link status is determined by the MDIO state machine.
		1	Not supported.
6	LINKINTENB		Link change interrupt enable. Set to 1 to enable link change status interrupts for the PHY address specified in PHYADRMON. Link change interrupts are disabled if this bit is cleared to 0.
		0	Link change interrupts are disabled.
		1	Link change status interrupts for PHY address specified in PHYADRMON bits are enabled.
5	Reserved	0	PHY address whose link status is to be monitored.
4-0	PHYADRMON	0-1Fh	PHY address whose link status is to be monitored.

31.5 EMAC Module Registers

Table 31-39 lists the memory-mapped registers for the EMAC. The base address for these registers is FCF7 8000h.

Table 31-39. Ethernet Media Access Controller (EMAC) Registers

Offset	Acronym	Register Description	Section
0h	TXREVID	Transmit Revision ID Register	Section 31.5.1
4h	TXCONTROL	Transmit Control Register	Section 31.5.2
8h	TXTEARDOWN	Transmit Teardown Register	Section 31.5.3
10h	RXREVID	Receive Revision ID Register	Section 31.5.4
14h	RXCONTROL	Receive Control Register	Section 31.5.5
18h	RXTEARDOWN	Receive Teardown Register	Section 31.5.6
80h	TXINTSTATRAW	Transmit Interrupt Status (Unmasked) Register	Section 31.5.7
84h	TXINTSTATMASKED	Transmit Interrupt Status (Masked) Register	Section 31.5.8
88h	TXINTMASKSET	Transmit Interrupt Mask Set Register	Section 31.5.9
8Ch	TXINTMASKCLEAR	Transmit Interrupt Clear Register	Section 31.5.10
90h	MACINVECTOR	MAC Input Vector Register	Section 31.5.11
94h	MACEOIVECTOR	MAC End Of Interrupt Vector Register	Section 31.5.12
A0h	RXINTSTATRAW	Receive Interrupt Status (Unmasked) Register	Section 31.5.13
A4h	RXINTSTATMASKED	Receive Interrupt Status (Masked) Register	Section 31.5.14
A8h	RXINTMASKSET	Receive Interrupt Mask Set Register	Section 31.5.15
ACh	RXINTMASKCLEAR	Receive Interrupt Mask Clear Register	Section 31.5.16
B0h	MACINTSTATRAW	MAC Interrupt Status (Unmasked) Register	Section 31.5.17
B4h	MACINTSTATMASKED	MAC Interrupt Status (Masked) Register	Section 31.5.18
B8h	MACINTMASKSET	MAC Interrupt Mask Set Register	Section 31.5.19
BCh	MACINTMASKCLEAR	MAC Interrupt Mask Clear Register	Section 31.5.20
100h	RXMBPENABLE	Receive Multicast/Broadcast/Promiscuous Channel Enable Register	Section 31.5.21
104h	RXUNICASTSET	Receive Unicast Enable Set Register	Section 31.5.22
108h	RXUNICASTCLEAR	Receive Unicast Clear Register	Section 31.5.23
10Ch	RXMAXLEN	Receive Maximum Length Register	Section 31.5.24
110h	RXBUFFEROFFSET	Receive Buffer Offset Register	Section 31.5.25
114h	RXFILTERLOWTHRESH	Receive Filter Low Priority Frame Threshold Register	Section 31.5.26
120h	RX0FLOWTHRESH	Receive Channel 0 Flow Control Threshold Register	Section 31.5.27
124h	RX1FLOWTHRESH	Receive Channel 1 Flow Control Threshold Register	Section 31.5.27
128h	RX2FLOWTHRESH	Receive Channel 2 Flow Control Threshold Register	Section 31.5.27
12Ch	RX3FLOWTHRESH	Receive Channel 3 Flow Control Threshold Register	Section 31.5.27
130h	RX4FLOWTHRESH	Receive Channel 4 Flow Control Threshold Register	Section 31.5.27
134h	RX5FLOWTHRESH	Receive Channel 5 Flow Control Threshold Register	Section 31.5.27
138h	RX6FLOWTHRESH	Receive Channel 6 Flow Control Threshold Register	Section 31.5.27
13Ch	RX7FLOWTHRESH	Receive Channel 7 Flow Control Threshold Register	Section 31.5.27
140h	RX0FREEBUFFER	Receive Channel 0 Free Buffer Count Register	Section 31.5.28
144h	RX1FREEBUFFER	Receive Channel 1 Free Buffer Count Register	Section 31.5.28
148h	RX2FREEBUFFER	Receive Channel 2 Free Buffer Count Register	Section 31.5.28
14Ch	RX3FREEBUFFER	Receive Channel 3 Free Buffer Count Register	Section 31.5.28
150h	RX4FREEBUFFER	Receive Channel 4 Free Buffer Count Register	Section 31.5.28
154h	RX5FREEBUFFER	Receive Channel 5 Free Buffer Count Register	Section 31.5.28
158h	RX6FREEBUFFER	Receive Channel 6 Free Buffer Count Register	Section 31.5.28
15Ch	RX7FREEBUFFER	Receive Channel 7 Free Buffer Count Register	Section 31.5.28
160h	MACCONTROL	MAC Control Register	Section 31.5.29

Table 31-39. Ethernet Media Access Controller (EMAC) Registers (continued)

Offset	Acronym	Register Description	Section
164h	MACSTATUS	MAC Status Register	Section 31.5.30
168h	EMCONTROL	Emulation Control Register	Section 31.5.31
16Ch	FIFOCONTROL	FIFO Control Register	Section 31.5.32
170h	MACCONFIG	MAC Configuration Register	Section 31.5.33
174h	SOFTRESET	Soft Reset Register	Section 31.5.34
1D0h	MACSRCADDRLO	MAC Source Address Low Bytes Register	Section 31.5.35
1D4h	MACSRCADDRHI	MAC Source Address High Bytes Register	Section 31.5.36
1D8h	MACHASH1	MAC Hash Address Register 1	Section 31.5.37
1DCh	MACHASH2	MAC Hash Address Register 2	Section 31.5.38
1E0h	BOFFTEST	Back Off Test Register	Section 31.5.39
1E4h	TPACETEST	Transmit Pacing Algorithm Test Register	Section 31.5.40
1E8h	RXPAUSE	Receive Pause Timer Register	Section 31.5.41
1ECh	TXPAUSE	Transmit Pause Timer Register	Section 31.5.42
500h	MACADDRLO	MAC Address Low Bytes Register	Section 31.5.43
504h	MACADDRHI	MAC Address High Bytes Register	Section 31.5.44
508h	MACINDEX	MAC Index Register	Section 31.5.45
600h	TX0HDP	Transmit Channel 0 DMA Head Descriptor Pointer Register	Section 31.5.46
604h	TX1HDP	Transmit Channel 1 DMA Head Descriptor Pointer Register	Section 31.5.46
608h	TX2HDP	Transmit Channel 2 DMA Head Descriptor Pointer Register	Section 31.5.46
60Ch	TX3HDP	Transmit Channel 3 DMA Head Descriptor Pointer Register	Section 31.5.46
610h	TX4HDP	Transmit Channel 4 DMA Head Descriptor Pointer Register	Section 31.5.46
614h	TX5HDP	Transmit Channel 5 DMA Head Descriptor Pointer Register	Section 31.5.46
618h	TX6HDP	Transmit Channel 6 DMA Head Descriptor Pointer Register	Section 31.5.46
61Ch	TX7HDP	Transmit Channel 7 DMA Head Descriptor Pointer Register	Section 31.5.46
620h	RX0HDP	Receive Channel 0 DMA Head Descriptor Pointer Register	Section 31.5.47
624h	RX1HDP	Receive Channel 1 DMA Head Descriptor Pointer Register	Section 31.5.47
628h	RX2HDP	Receive Channel 2 DMA Head Descriptor Pointer Register	Section 31.5.47
62Ch	RX3HDP	Receive Channel 3 DMA Head Descriptor Pointer Register	Section 31.5.47
630h	RX4HDP	Receive Channel 4 DMA Head Descriptor Pointer Register	Section 31.5.47
634h	RX5HDP	Receive Channel 5 DMA Head Descriptor Pointer Register	Section 31.5.47
638h	RX6HDP	Receive Channel 6 DMA Head Descriptor Pointer Register	Section 31.5.47
63Ch	RX7HDP	Receive Channel 7 DMA Head Descriptor Pointer Register	Section 31.5.47
640h	TX0CP	Transmit Channel 0 Completion Pointer Register	Section 31.5.48
644h	TX1CP	Transmit Channel 1 Completion Pointer Register	Section 31.5.48
648h	TX2CP	Transmit Channel 2 Completion Pointer Register	Section 31.5.48
64Ch	TX3CP	Transmit Channel 3 Completion Pointer Register	Section 31.5.48
650h	TX4CP	Transmit Channel 4 Completion Pointer Register	Section 31.5.48
654h	TX5CP	Transmit Channel 5 Completion Pointer Register	Section 31.5.48
658h	TX6CP	Transmit Channel 6 Completion Pointer Register	Section 31.5.48
65Ch	TX7CP	Transmit Channel 7 Completion Pointer Register	Section 31.5.48
660h	RX0CP	Receive Channel 0 Completion Pointer Register	Section 31.5.49
664h	RX1CP	Receive Channel 1 Completion Pointer Register	Section 31.5.49
668h	RX2CP	Receive Channel 2 Completion Pointer Register	Section 31.5.49
66Ch	RX3CP	Receive Channel 3 Completion Pointer Register	Section 31.5.49
670h	RX4CP	Receive Channel 4 Completion Pointer Register	Section 31.5.49
674h	RX5CP	Receive Channel 5 Completion Pointer Register	Section 31.5.49
678h	RX6CP	Receive Channel 6 Completion Pointer Register	Section 31.5.49

Table 31-39. Ethernet Media Access Controller (EMAC) Registers (continued)

Offset	Acronym	Register Description	Section
67Ch	RX7CP	Receive Channel 7 Completion Pointer Register	Section 31.5.49
		Network Statistics Registers	
200h	RXGOODFRAMES	Good Receive Frames Register	Section 31.5.50.1
204h	RXBCASTFRAMES	Broadcast Receive Frames Register	Section 31.5.50.2
208h	RXMCASTFRAMES	Multicast Receive Frames Register	Section 31.5.50.3
20Ch	RXPAUSEFRAMES	Pause Receive Frames Register	Section 31.5.50.4
210h	RXCRCERRORS	Receive CRC Errors Register	Section 31.5.50.5
214h	RXALIGNCODEERRORS	Receive Alignment/Code Errors Register	Section 31.5.50.6
218h	RXOVERSIZED	Receive Oversized Frames Register	Section 31.5.50.7
21Ch	RXJABBER	Receive Jabber Frames Register	Section 31.5.50.8
220h	RXUNDERSIZED	Receive Undersized Frames Register	Section 31.5.50.9
224h	RXFRAGMENTS	Receive Frame Fragments Register	Section 31.5.50.10
228h	RXFILTERED	Filtered Receive Frames Register	Section 31.5.50.11
22Ch	RXQOSFILTERED	Receive QOS Filtered Frames Register	Section 31.5.50.12
230h	RXOCTETS	Receive Octet Frames Register	Section 31.5.50.13
234h	TXGOODFRAMES	Good Transmit Frames Register	Section 31.5.50.14
238h	TXBCASTFRAMES	Broadcast Transmit Frames Register	Section 31.5.50.15
23Ch	TXMCASTFRAMES	Multicast Transmit Frames Register	Section 31.5.50.16
240h	TXPAUSEFRAMES	Pause Transmit Frames Register	Section 31.5.50.17
244h	TXDEFERRED	Deferred Transmit Frames Register	Section 31.5.50.18
248h	TXCOLLISION	Transmit Collision Frames Register	Section 31.5.50.19
24Ch	TXSINGLECOLL	Transmit Single Collision Frames Register	Section 31.5.50.20
250h	TXMULTICOLL	Transmit Multiple Collision Frames Register	Section 31.5.50.21
254h	TXEXCESSIVECOLL	Transmit Excessive Collision Frames Register	Section 31.5.50.22
258h	TXLATECOLL	Transmit Late Collision Frames Register	Section 31.5.50.23
25Ch	TXUNDERRUN	Transmit Underrun Error Register	Section 31.5.50.24
260h	TXCARRIERSENSE	Transmit Carrier Sense Errors Register	Section 31.5.50.25
264h	TXOCTETS	Transmit Octet Frames Register	Section 31.5.50.26
268h	FRAME64	Transmit and Receive 64 Octet Frames Register	Section 31.5.50.27
26Ch	FRAME65T127	Transmit and Receive 65 to 127 Octet Frames Register	Section 31.5.50.28
270h	FRAME128T255	Transmit and Receive 128 to 255 Octet Frames Register	Section 31.5.50.29
274h	FRAME256T511	Transmit and Receive 256 to 511 Octet Frames Register	Section 31.5.50.30
278h	FRAME512T1023	Transmit and Receive 512 to 1023 Octet Frames Register	Section 31.5.50.31
27Ch	FRAME1024TUP	Transmit and Receive 1024 to RXMAXLEN Octet Frames Register	Section 31.5.50.32
280h	NETOCTETS	Network Octet Frames Register	Section 31.5.50.33
284h	RXSOFOVERRUNS	Receive FIFO or DMA Start of Frame Overruns Register	Section 31.5.50.34
288h	RXMOFOVERRUNS	Receive FIFO or DMA Middle of Frame Overruns Register	Section 31.5.50.35
28Ch	RXDMAOVERRUNS	Receive DMA Overruns Register	Section 31.5.50.36

31.5.1 Transmit Revision ID Register (TXREVID)

The transmit revision ID register (TXREVID) is shown in Figure 31-42 and described in Table 31-40.

Figure 31-42. Transmit Revision ID Register (TXREVID)

31 … 0
TXREV
R-4EC0 020Dh

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

Table 31-40. Transmit Revision ID Register (TXREVID) Field Descriptions

Bit	Field	Value	Description
31-0	TXREV		Transmit module revision
		4EC0 020Dh	Current transmit revision value.

31.5.2 Transmit Control Register (TXCONTROL)

The transmit control register (TXCONTROL) is shown in Figure 31-43 and described in Table 31-41.

Figure 31-43. Transmit Control Register (TXCONTROL)

31 … 16
Reserved
R-0

15 … 1	0
Reserved	TXEN
R-0	R/W-0

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

Table 31-41. Transmit Control Register (TXCONTROL) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	TXEN		Transmit enable
		0	Transmit is disabled.
		1	Transmit is enabled.

31.5.3 Transmit Teardown Register (TXTEARDOWN)

The transmit teardown register (TXTEARDOWN) is shown in Figure 31-44 and described in Table 31-42.

Figure 31-44. Transmit Teardown Register (TXTEARDOWN)

31	16
Reserved	
R-0	

15	3	2	0
Reserved		TXTDNCH	
R-0		R/W-0	

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

Table 31-42. Transmit Teardown Register (TXTEARDOWN) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	TXTDNCH		Transmit teardown channel. The transmit channel teardown is commanded by writing the encoded value of the transmit channel to be torn down. The teardown register is read as 0.
		0	Teardown transmit channel 0
		1h	Teardown transmit channel 1
		2h	Teardown transmit channel 2
		3h	Teardown transmit channel 3
		4h	Teardown transmit channel 4
		5h	Teardown transmit channel 5
		6h	Teardown transmit channel 6
		7h	Teardown transmit channel 7

31.5.4 Receive Revision ID Register (RXREVID)

The receive revision ID register (RXREVID) is shown in Figure 31-45 and described in Table 31-43.

Figure 31-45. Receive Revision ID Register (RXREVID)

31	0
RXREV	
R-4EC0 020Dh	

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

Table 31-43. Receive Revision ID Register (RXREVID) Field Descriptions

Bit	Field	Value	Description
31-0	RXREV		Receive module revision
		4EC0 020Dh	Current receive revision value.

31.5.5 Receive Control Register (RXCONTROL)

The receive control register (RXCONTROL) is shown in Figure 31-46 and described in Table 31-44.

Figure 31-46. Receive Control Register (RXCONTROL)

31–16
Reserved
R-0

15–1	0
Reserved	RXEN
R-0	R/W-0

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

Table 31-44. Receive Control Register (RXCONTROL) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	RXEN		Receive enable
		0	Receive is disabled.
		1	Receive is enabled.

31.5.6 Receive Teardown Register (RXTEARDOWN)

The receive teardown register (RXTEARDOWN) is shown in Figure 31-47 and described in Table 31-45.

Figure 31-47. Receive Teardown Register (RXTEARDOWN)

31–16
Reserved
R-0

15–3	2–0
Reserved	RXTDNCH
R-0	R/W-0

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

Table 31-45. Receive Teardown Register (RXTEARDOWN) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	RXTDNCH		Receive teardown channel. The receive channel teardown is commanded by writing the encoded value of the receive channel to be torn down. The teardown register is read as 0.
		0	Teardown receive channel 0
		1h	Teardown receive channel 1
		2h	Teardown receive channel 2
		3h	Teardown receive channel 3
		4h	Teardown receive channel 4
		5h	Teardown receive channel 5
		6h	Teardown receive channel 6
		7h	Teardown receive channel 7

31.5.7 Transmit Interrupt Status (Unmasked) Register (TXINTSTATRAW)

The transmit interrupt status (unmasked) register (TXINTSTATRAW) is shown in Figure 31-48 and described in Table 31-46.

Figure 31-48. Transmit Interrupt Status (Unmasked) Register (TXINTSTATRAW)

31-16
Reserved
R-0

15-8
Reserved
R-0

7	6	5	4	3	2	1	0
TX7PEND	TX6PEND	TX5PEND	TX4PEND	TX3PEND	TX2PEND	TX1PEND	TX0PEND
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 31-46. Transmit Interrupt Status (Unmasked) Register (TXINTSTATRAW) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	TX7PEND	0-1	TX7PEND raw interrupt read (before mask)
6	TX6PEND	0-1	TX6PEND raw interrupt read (before mask)
5	TX5PEND	0-1	TX5PEND raw interrupt read (before mask)
4	TX4PEND	0-1	TX4PEND raw interrupt read (before mask)
3	TX3PEND	0-1	TX3PEND raw interrupt read (before mask)
2	TX2PEND	0-1	TX2PEND raw interrupt read (before mask)
1	TX1PEND	0-1	TX1PEND raw interrupt read (before mask)
0	TX0PEND	0-1	TX0PEND raw interrupt read (before mask)

31.5.8 Transmit Interrupt Status (Masked) Register (TXINTSTATMASKED)

The transmit interrupt status (masked) register (TXINTSTATMASKED) is shown in Figure 31-49 and described in Table 31-47.

Figure 31-49. Transmit Interrupt Status (Masked) Register (TXINTSTATMASKED)

31–16
Reserved
R-0

15–8
Reserved
R-0

7	6	5	4	3	2	1	0
TX7PEND	TX6PEND	TX5PEND	TX4PEND	TX3PEND	TX2PEND	TX1PEND	TX0PEND
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 31-47. Transmit Interrupt Status (Masked) Register (TXINTSTATMASKED) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	TX7PEND	0-1	TX7PEND masked interrupt read
6	TX6PEND	0-1	TX6PEND masked interrupt read
5	TX5PEND	0-1	TX5PEND masked interrupt read
4	TX4PEND	0-1	TX4PEND masked interrupt read
3	TX3PEND	0-1	TX3PEND masked interrupt read
2	TX2PEND	0-1	TX2PEND masked interrupt read
1	TX1PEND	0-1	TX1PEND masked interrupt read
0	TX0PEND	0-1	TX0PEND masked interrupt read

31.5.9 Transmit Interrupt Mask Set Register (TXINTMASKSET)

The transmit interrupt mask set register (TXINTMASKSET) is shown in Figure 31-50 and described in Table 31-48.

Figure 31-50. Transmit Interrupt Mask Set Register (TXINTMASKSET)

31	16
Reserved	
R-0	

15	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
TX7MASK	TX6MASK	TX5MASK	TX4MASK	TX3MASK	TX2MASK	TX1MASK	TX0MASK
R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0

LEGEND: R/W = Read/Write; R = Read only; W1S = Write 1 to set (writing a 0 has no effect); -*n* = value after reset

Table 31-48. Transmit Interrupt Mask Set Register (TXINTMASKSET) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	TX7MASK	0-1	Transmit channel 7 interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
6	TX6MASK	0-1	Transmit channel 6 interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
5	TX5MASK	0-1	Transmit channel 5 interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
4	TX4MASK	0-1	Transmit channel 4 interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
3	TX3MASK	0-1	Transmit channel 3 interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
2	TX2MASK	0-1	Transmit channel 2 interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
1	TX1MASK	0-1	Transmit channel 1 interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
0	TX0MASK	0-1	Transmit channel 0 interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.

31.5.10 Transmit Interrupt Mask Clear Register (TXINTMASKCLEAR)

The transmit interrupt mask clear register (TXINTMASKCLEAR) is shown in Figure 31-51 and described in Table 31-49.

Figure 31-51. Transmit Interrupt Mask Clear Register (TXINTMASKCLEAR)

31	16
Reserved	
R-0	

15	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
TX7MASK	TX6MASK	TX5MASK	TX4MASK	TX3MASK	TX2MASK	TX1MASK	TX0MASK
R/W1C-0	R/W1C-0	R/W1C-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 (writing a 0 has no effect); -*n* = value after reset

Table 31-49. Transmit Interrupt Mask Clear Register (TXINTMASKCLEAR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	TX7MASK	0-1	Transmit channel 7 interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
6	TX6MASK	0-1	Transmit channel 6 interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
5	TX5MASK	0-1	Transmit channel 5 interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
4	TX4MASK	0-1	Transmit channel 4 interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
3	TX3MASK	0-1	Transmit channel 3 interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
2	TX2MASK	0-1	Transmit channel 2 interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
1	TX1MASK	0-1	Transmit channel 1 interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
0	TX0MASK	0-1	Transmit channel 0 interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.

31.5.11 MAC Input Vector Register (MACINVECTOR)

The MAC input vector register (MACINVECTOR) is shown in Figure 31-52 and described in Table 31-50.

Figure 31-52. MAC Input Vector Register (MACINVECTOR)

31 – 28	27	26	25	24	23 – 16
Reserved	STATPEND	HOSTPEND	LINKINT0	USERINT0	TXPEND
R-0	R-0	R-0	R-0	R-0	R-0

15 – 8	7 – 0
RXTHRESHPEND	RXPEND
R-0	R-0

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

Table 31-50. MAC Input Vector Register (MACINVECTOR) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reserved
27	STATPEND	0-1	EMAC module statistics interrupt (STATPEND) pending status bit.
26	HOSTPEND	0-1	EMAC module host error interrupt (HOSTPEND) pending status bit.
25	LINKINT0	0-1	MDIO module USERPHYSEL0 (LINKINT0) status bit.
24	USERINT0	0-1	MDIO module USERACCESS0 (USERINT0) status bit.
23-16	TXPEND	0-FFh	Transmit channels 0-7 interrupt (TX*n*PEND) pending status. Bit 16 is TX0PEND.
15-8	RXTHRESHPEND	0-FFh	Receive channels 0-7 interrupt (RX*n*THRESHPEND) pending status. Bit 8 is RX0THRESHPEND.
7-0	RXPEND	0-FFh	Receive channels 0-7 interrupt (RX*n*PEND) pending status bit. Bit 0 is RX0PEND.

31.5.12 MAC End Of Interrupt Vector Register (MACEOIVECTOR)

The MAC end of interrupt vector register (MACEOIVECTOR) is shown in Figure 31-53 and described in Table 31-51.

Figure 31-53. MAC End Of Interrupt Vector Register (MACEOIVECTOR)

31	16
Reserved	
R-0	

15	5	4	0
Reserved		INTVECT	
R-0		R/W-0	

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

Table 31-51. MAC End Of Interrupt Vector Register (MACEOIVECTOR) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4-0	INTVECT		Acknowledge EMAC Control Module Interrupts
		0h	Acknowledge C0RXTHRESH Interrupt
		1h	Acknowledge C0RX Interrupt
		2h	Acknowledge C0TX Interrupt
		3h	Acknowledge C0MISC Interrupt (STATPEND, HOSTPEND, MDIO LINKINT0, MDIO USERINT0)
		4h	Acknowledge C1RXTHRESH Interrupt
		5h	Acknowledge C1RX Interrupt
		6h	Acknowledge C1TX Interrupt
		7h	Acknowledge C1MISC Interrupt (STATPEND, HOSTPEND, MDIO LINKINT0, MDIO USERINT0)
		8h	Acknowledge C2RXTHRESH Interrupt
		9h	Acknowledge C2RX Interrupt
		Ah	Acknowledge C2TX Interrupt
		Bh	Acknowledge C2MISC Interrupt (STATPEND, HOSTPEND, MDIO LINKINT0, MDIO USERINT0)
		Ch-1Fh	Reserved

31.5.13 Receive Interrupt Status (Unmasked) Register (RXINTSTATRAW)

The receive interrupt status (unmasked) register (RXINTSTATRAW) is shown in Figure 31-54 and described in Table 31-52.

Figure 31-54. Receive Interrupt Status (Unmasked) Register (RXINTSTATRAW)

31–16
Reserved
R-0

15	14	13	12	11	10	9	8
RX7THRESHPEND	RX6THRESHPEND	RX5THRESHPEND	RX4THRESHPEND	RX3THRESHPEND	RX2THRESHPEND	RX1THRESHPEND	RX0THRESHPEND
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

7	6	5	4	3	2	1	0
RX7PEND	RX6PEND	RX5PEND	RX4PEND	RX3PEND	RX2PEND	RX1PEND	RX0PEND
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 31-52. Receive Interrupt Status (Unmasked) Register (RXINTSTATRAW) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	RX7THRESHPEND	0-1	RX7THRESHPEND raw interrupt read (before mask)
14	RX6THRESHPEND	0-1	RX6THRESHPEND raw interrupt read (before mask)
13	RX5THRESHPEND	0-1	RX5THRESHPEND raw interrupt read (before mask)
12	RX4THRESHPEND	0-1	RX4THRESHPEND raw interrupt read (before mask)
11	RX3THRESHPEND	0-1	RX3THRESHPEND raw interrupt read (before mask)
10	RX2THRESHPEND	0-1	RX2THRESHPEND raw interrupt read (before mask)
9	RX1THRESHPEND	0-1	RX1THRESHPEND raw interrupt read (before mask)
8	RX0THRESHPEND	0-1	RX0THRESHPEND raw interrupt read (before mask)
7	RX7PEND	0-1	RX7PEND raw interrupt read (before mask)
6	RX6PEND	0-1	RX6PEND raw interrupt read (before mask)
5	RX5PEND	0-1	RX5PEND raw interrupt read (before mask)
4	RX4PEND	0-1	RX4PEND raw interrupt read (before mask)
3	RX3PEND	0-1	RX3PEND raw interrupt read (before mask)
2	RX2PEND	0-1	RX2PEND raw interrupt read (before mask)
1	RX1PEND	0-1	RX1PEND raw interrupt read (before mask)
0	RX0PEND	0-1	RX0PEND raw interrupt read (before mask)

31.5.14 Receive Interrupt Status (Masked) Register (RXINTSTATMASKED)

The receive interrupt status (masked) register (RXINTSTATMASKED) is shown in Figure 31-55 and described in Table 31-53.

Figure 31-55. Receive Interrupt Status (Masked) Register (RXINTSTATMASKED)

31 – 16
Reserved
R-0

15	14	13	12	11	10	9	8
RX7THRESH PEND	RX6THRESH PEND	RX5THRESH PEND	RX4THRESH PEND	RX3THRESH PEND	RX2THRESH PEND	RX1THRESH PEND	RX0THRESH PEND
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

7	6	5	4	3	2	1	0
RX7PEND	RX6PEND	RX5PEND	RX4PEND	RX3PEND	RX2PEND	RX1PEND	RX0PEND
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 31-53. Receive Interrupt Status (Masked) Register (RXINTSTATMASKED) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	RX7THRESHPEND	0-1	RX7THRESHPEND masked interrupt read
14	RX6THRESHPEND	0-1	RX6THRESHPEND masked interrupt read
13	RX5THRESHPEND	0-1	RX5THRESHPEND masked interrupt read
12	RX4THRESHPEND	0-1	RX4THRESHPEND masked interrupt read
11	RX3THRESHPEND	0-1	RX3THRESHPEND masked interrupt read
10	RX2THRESHPEND	0-1	RX2THRESHPEND masked interrupt read
9	RX1THRESHPEND	0-1	RX1THRESHPEND masked interrupt read
8	RX0THRESHPEND	0-1	RX0THRESHPEND masked interrupt read
7	RX7PEND	0-1	RX7PEND masked interrupt read
6	RX6PEND	0-1	RX6PEND masked interrupt read
5	RX5PEND	0-1	RX5PEND masked interrupt read
4	RX4PEND	0-1	RX4PEND masked interrupt read
3	RX3PEND	0-1	RX3PEND masked interrupt read
2	RX2PEND	0-1	RX2PEND masked interrupt read
1	RX1PEND	0-1	RX1PEND masked interrupt read
0	RX0PEND	0-1	RX0PEND masked interrupt read

31.5.15 Receive Interrupt Mask Set Register (RXINTMASKSET)

The receive interrupt mask set register (RXINTMASKSET) is shown in Figure 31-56 and described in Table 31-54.

Figure 31-56. Receive Interrupt Mask Set Register (RXINTMASKSET)

31-16
Reserved
R-0

15	14	13	12	11	10	9	8
RX7THRESH MASK	RX6THRESH MASK	RX5THRESH MASK	RX4THRESH MASK	RX3THRESH MASK	RX2THRESH MASK	RX1THRESH MASK	RX0THRESH MASK
R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0

7	6	5	4	3	2	1	0
RX7MASK	RX6MASK	RX5MASK	RX4MASK	RX3MASK	RX2MASK	RX1MASK	RX0MASK
R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0

LEGEND: R/W = Read/Write; R = Read only; W1S = Write 1 to set (writing a 0 has no effect); -*n* = value after reset

Table 31-54. Receive Interrupt Mask Set Register (RXINTMASKSET) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	RX7THRESHMASK	0-1	Receive channel 7 threshold mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
14	RX6THRESHMASK	0-1	Receive channel 6 threshold mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
13	RX5THRESHMASK	0-1	Receive channel 5 threshold mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
12	RX4THRESHMASK	0-1	Receive channel 4 threshold mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
11	RX3THRESHMASK	0-1	Receive channel 3 threshold mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
10	RX2THRESHMASK	0-1	Receive channel 2 threshold mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
9	RX1THRESHMASK	0-1	Receive channel 1 threshold mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
8	RX0THRESHMASK	0-1	Receive channel 0 threshold mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
7	RX7MASK	0-1	Receive channel 7 mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
6	RX6MASK	0-1	Receive channel 6 mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
5	RX5MASK	0-1	Receive channel 5 mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
4	RX4MASK	0-1	Receive channel 4 mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
3	RX3MASK	0-1	Receive channel 3 mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
2	RX2MASK	0-1	Receive channel 2 mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
1	RX1MASK	0-1	Receive channel 1 mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.
0	RX0MASK	0-1	Receive channel 0 mask set bit. Write 1 to enable interrupt; a write of 0 has no effect.

31.5.16 Receive Interrupt Mask Clear Register (RXINTMASKCLEAR)

The receive interrupt mask clear register (RXINTMASKCLEAR) is shown in Figure 31-57 and described in Table 31-55.

Figure 31-57. Receive Interrupt Mask Clear Register (RXINTMASKCLEAR)

31–16
Reserved
R-0

15	14	13	12	11	10	9	8
RX7THRESH MASK	RX6THRESH MASK	RX5THRESH MASK	RX4THRESH MASK	RX3THRESH MASK	RX2THRESH MASK	RX1THRESH MASK	RX0THRESH MASK
R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0

7	6	5	4	3	2	1	0
RX7MASK	RX6MASK	RX5MASK	RX4MASK	RX3MASK	RX2MASK	RX1MASK	RX0MASK
R/W1C-0	R/W1C-0	R/W1C-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 (writing a 0 has no effect); -*n* = value after reset

Table 31-55. Receive Interrupt Mask Clear Register (RXINTMASKCLEAR) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	RX7THRESHMASK	0-1	Receive channel 7 threshold mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
14	RX6THRESHMASK	0-1	Receive channel 6 threshold mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
13	RX5THRESHMASK	0-1	Receive channel 5 threshold mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
12	RX4THRESHMASK	0-1	Receive channel 4 threshold mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
11	RX3THRESHMASK	0-1	Receive channel 3 threshold mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
10	RX2THRESHMASK	0-1	Receive channel 2 threshold mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
9	RX1THRESHMASK	0-1	Receive channel 1 threshold mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
8	RX0THRESHMASK	0-1	Receive channel 0 threshold mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
7	RX7MASK	0-1	Receive channel 7 mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
6	RX6MASK	0-1	Receive channel 6 mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
5	RX5MASK	0-1	Receive channel 5 mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
4	RX4MASK	0-1	Receive channel 4 mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
3	RX3MASK	0-1	Receive channel 3 mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
2	RX2MASK	0-1	Receive channel 2 mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
1	RX1MASK	0-1	Receive channel 1 mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.
0	RX0MASK	0-1	Receive channel 0 mask clear bit. Write 1 to disable interrupt; a write of 0 has no effect.

31.5.17 MAC Interrupt Status (Unmasked) Register (MACINTSTATRAW)

The MAC interrupt status (unmasked) register (MACINTSTATRAW) is shown in Figure 31-58 and described in Table 31-56.

Figure 31-58. MAC Interrupt Status (Unmasked) Register (MACINTSTATRAW)

31	16
Reserved	
R-0	

15 2	1	0
Reserved	HOSTPEND	STATPEND
R-0	R-0	R-0

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

Table 31-56. MAC Interrupt Status (Unmasked) Register (MACINTSTATRAW) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	HOSTPEND	0-1	Host pending interrupt (HOSTPEND); raw interrupt read (before mask).
0	STATPEND	0-1	Statistics pending interrupt (STATPEND); raw interrupt read (before mask).

31.5.18 MAC Interrupt Status (Masked) Register (MACINTSTATMASKED)

The MAC interrupt status (masked) register (MACINTSTATMASKED) is shown in Figure 31-59 and described in Table 31-57.

Figure 31-59. MAC Interrupt Status (Masked) Register (MACINTSTATMASKED)

31	16
Reserved	
R-0	

15 2	1	0
Reserved	HOSTPEND	STATPEND
R-0	R-0	R-0

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

Table 31-57. MAC Interrupt Status (Masked) Register (MACINTSTATMASKED) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	HOSTPEND	0-1	Host pending interrupt (HOSTPEND); masked interrupt read.
0	STATPEND	0-1	Statistics pending interrupt (STATPEND); masked interrupt read.

31.5.19 MAC Interrupt Mask Set Register (MACINTMASKSET)

The MAC interrupt mask set register (MACINTMASKSET) is shown in Figure 31-60 and described in Table 31-58.

Figure 31-60. MAC Interrupt Mask Set Register (MACINTMASKSET)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		HOSTMASK	STATMASK
R-0		R/W1S-0	R/W1S-0

LEGEND: R/W = Read/Write; R = Read only; W1S = Write 1 to set (writing a 0 has no effect); -*n* = value after reset

Table 31-58. MAC Interrupt Mask Set Register (MACINTMASKSET) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	HOSTMASK	0-1	Host error interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.
0	STATMASK	0-1	Statistics interrupt mask set bit. Write 1 to enable interrupt, a write of 0 has no effect.

31.5.20 MAC Interrupt Mask Clear Register (MACINTMASKCLEAR)

The MAC interrupt mask clear register (MACINTMASKCLEAR) is shown in Figure 31-61 and described in Table 31-59.

Figure 31-61. MAC Interrupt Mask Clear Register (MACINTMASKCLEAR)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		HOSTMASK	STATMASK
R-0		R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear (writing a 0 has no effect); -*n* = value after reset

Table 31-59. MAC Interrupt Mask Clear Register (MACINTMASKCLEAR) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	HOSTMASK	0-1	Host error interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.
0	STATMASK	0-1	Statistics interrupt mask clear bit. Write 1 to disable interrupt, a write of 0 has no effect.

31.5.21 Receive Multicast/Broadcast/Promiscuous Channel Enable Register (RXMBPENABLE)

The receive multicast/broadcast/promiscuous channel enable register (RXMBPENABLE) is shown in Figure 31-62 and described in Table 31-60.

Figure 31-62. Receive Multicast/Broadcast/Promiscuous Channel Enable Register (RXMBPENABLE)

31	30	29	28	27		25	24
Reserved	RXPASSCRC	RXQOSEN	RXNOCHAIN	Reserved			RXCMFEN
R-0	R/W-0	R/W-0	R/W-0	R-0			R/W-0

23	22	21	20	19	18		16
RXCSFEN	RXCEFEN	RXCAFEN	Reserved		RXPROMCH		
R/W-0	R/W-0	R/W-0	R-0		R/W-0		

15	14	13	12	11	10		8
Reserved		RXBROADEN	Reserved		RXBROADCH		
R-0		R/W-0	R-0		R/W-0		

7	6	5	4	3	2		0
Reserved		RXMULTEN	Reserved		RXMULTCH		
R-0		R/W-0	R-0		R/W-0		

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

Table 31-60. Receive Multicast/Broadcast/Promiscuous Channel Enable Register (RXMBPENABLE) Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reserved
30	RXPASSCRC		Pass receive CRC enable bit
		0	Received CRC is discarded for all channels and is not included in the buffer descriptor packet length field.
		1	Received CRC is transferred to memory for all channels and is included in the buffer descriptor packet length.
29	RXQOSEN		Receive quality of service enable bit
		0	Receive QOS is disabled.
		1	Receive QOS is enabled.
28	RXNOCHAIN		Receive no buffer chaining bit
		0	Received frames can span multiple buffers.
		1	The Receive DMA controller transfers each frame into a single buffer, regardless of the frame or buffer size. All remaining frame data after the first buffer is discarded. The buffer descriptor buffer length field will contain the entire frame byte count (up to 65535 bytes).
27-25	Reserved	0	Reserved
24	RXCMFEN		Receive copy MAC control frames enable bit. Enables MAC control frames to be transferred to memory. MAC control frames are normally acted upon (if enabled), but not copied to memory. MAC control frames that are pause frames will be acted upon if enabled in MACCONTROL, regardless of the value of RXCMFEN. Frames transferred to memory due to RXCMFEN will have the CONTROL bit set in their EOP buffer descriptor.
		0	MAC control frames are filtered (but acted upon if enabled).
		1	MAC control frames are transferred to memory.
23	RXCSFEN		Receive copy short frames enable bit. Enables frames or fragments shorter than 64 bytes to be copied to memory. Frames transferred to memory due to RXCSFEN will have the FRAGMENT or UNDERSIZE bit set in their EOP buffer descriptor. Fragments are short frames that contain CRC / align / code errors and undersized are short frames without errors.
		0	Short frames are filtered.
		1	Short frames are transferred to memory.

Table 31-60. Receive Multicast/Broadcast/Promiscuous Channel Enable Register (RXMBPENABLE) Field Descriptions (continued)

Bit	Field	Value	Description
22	RXCEFEN		Receive copy error frames enable bit. Enables frames containing errors to be transferred to memory. The appropriate error bit will be set in the frame EOP buffer descriptor.
		0	Frames containing errors are filtered.
		1	Frames containing errors are transferred to memory.
21	RXCAFEN		Receive copy all frames enable bit. Enables frames that do not address match (includes multicast frames that do not hash match) to be transferred to the promiscuous channel selected by RXPROMCH bits. Such frames will be marked with the NOMATCH bit in their EOP buffer descriptor.
		0	Frames that do not address match are filtered.
		1	Frames that do not address match are transferred to the promiscuous channel selected by RXPROMCH bits.
20-19	Reserved	0	Reserved
18-16	RXPROMCH		Receive promiscuous channel select
		0	Select channel 0 to receive promiscuous frames.
		1h	Select channel 1 to receive promiscuous frames.
		2h	Select channel 2 to receive promiscuous frames.
		3h	Select channel 3 to receive promiscuous frames.
		4h	Select channel 4 to receive promiscuous frames.
		5h	Select channel 5 to receive promiscuous frames.
		6h	Select channel 6 to receive promiscuous frames.
		7h	Select channel 7 to receive promiscuous frames.
15-14	Reserved	0	Reserved
13	RXBROADEN		Receive broadcast enable. Enable received broadcast frames to be copied to the channel selected by RXBROADCH bits.
		0	Broadcast frames are filtered.
		1	Broadcast frames are copied to the channel selected by RXBROADCH bits.
12-11	Reserved	0	Reserved
10-8	RXBROADCH		Receive broadcast channel select
		0	Select channel 0 to receive broadcast frames.
		1h	Select channel 1 to receive broadcast frames.
		2h	Select channel 2 to receive broadcast frames.
		3h	Select channel 3 to receive broadcast frames.
		4h	Select channel 4 to receive broadcast frames.
		5h	Select channel 5 to receive broadcast frames.
		6h	Select channel 6 to receive broadcast frames.
		7h	Select channel 7 to receive broadcast frames.
7-6	Reserved	0	Reserved
5	RXMULTEN		RX multicast enable. Enable received hash matching multicast frames to be copied to the channel selected by RXMULTCH bits.
		0	Multicast frames are filtered.
		1	Multicast frames are copied to the channel selected by RXMULTCH bits.
4-3	Reserved	0	Reserved

Table 31-60. Receive Multicast/Broadcast/Promiscuous Channel Enable Register (RXMBPENABLE) Field Descriptions (continued)

Bit	Field	Value	Description
2-0	RXMULTCH		Receive multicast channel select
		0	Select channel 0 to receive multicast frames.
		1h	Select channel 1 to receive multicast frames.
		2h	Select channel 2 to receive multicast frames.
		3h	Select channel 3 to receive multicast frames.
		4h	Select channel 4 to receive multicast frames.
		5h	Select channel 5 to receive multicast frames.
		6h	Select channel 6 to receive multicast frames.
		7h	Select channel 7 to receive multicast frames.

31.5.22 Receive Unicast Enable Set Register (RXUNICASTSET)

The receive unicast enable set register (RXUNICASTSET) is shown in Figure 31-63 and described in Table 31-61.

Figure 31-63. Receive Unicast Enable Set Register (RXUNICASTSET)

31	16
Reserved	
R-0	

15	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
RXCH7EN	RXCH6EN	RXCH5EN	RXCH4EN	RXCH3EN	RXCH2EN	RXCH1EN	RXCH0EN
R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0	R/W1S-0

LEGEND: R/W = Read/Write; R = Read only; W1S = Write 1 to set (writing a 0 has no effect); -*n* = value after reset

Table 31-61. Receive Unicast Enable Set Register (RXUNICASTSET) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	RXCH7EN	0-1	Receive channel 7 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
6	RXCH6EN	0-1	Receive channel 6 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
5	RXCH5EN	0-1	Receive channel 5 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
4	RXCH4EN	0-1	Receive channel 4 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
3	RXCH3EN	0-1	Receive channel 3 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
2	RXCH2EN	0-1	Receive channel 2 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
1	RXCH1EN	0-1	Receive channel 1 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.
0	RXCH0EN	0-1	Receive channel 0 unicast enable set bit. Write 1 to set the enable, a write of 0 has no effect. May be read.

31.5.23 Receive Unicast Clear Register (RXUNICASTCLEAR)

The receive unicast clear register (RXUNICASTCLEAR) is shown in Figure 31-64 and described in Table 31-62.

Figure 31-64. Receive Unicast Clear Register (RXUNICASTCLEAR)

31	16
Reserved	
R-0	

15	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
RXCH7EN	RXCH6EN	RXCH5EN	RXCH4EN	RXCH3EN	RXCH2EN	RXCH1EN	RXCH0EN
R/W1C-0	R/W1C-0	R/W1C-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 (writing a 0 has no effect); -*n* = value after reset

Table 31-62. Receive Unicast Clear Register (RXUNICASTCLEAR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7	RXCH7EN	0-1	Receive channel 7 unicast enable clear bit. Write 1 to clear the enable, a write of 0 has no effect.
6	RXCH6EN	0-1	Receive channel 6 unicast enable clear bit. Write 1 to clear the enable, a write of 0 has no effect.
5	RXCH5EN	0-1	Receive channel 5 unicast enable clear bit. Write 1 to clear the enable, a write of 0 has no effect.
4	RXCH4EN	0-1	Receive channel 4 unicast enable clear bit. Write 1 to clear the enable, a write of 0 has no effect.
3	RXCH3EN	0-1	Receive channel 3 unicast enable clear bit. Write 1 to clear the enable, a write of 0 has no effect.
2	RXCH2EN	0-1	Receive channel 2 unicast enable clear bit. Write 1 to clear the enable, a write of 0 has no effect.
1	RXCH1EN	0-1	Receive channel 1 unicast enable clear bit. Write 1 to clear the enable, a write of 0 has no effect.
0	RXCH0EN	0-1	Receive channel 0 unicast enable clear bit. Write 1 to clear the enable, a write of 0 has no effect.

31.5.24 Receive Maximum Length Register (RXMAXLEN)

The receive maximum length register (RXMAXLEN) is shown in Figure 31-65 and described in Table 31-63.

Figure 31-65. Receive Maximum Length Register (RXMAXLEN)

31	16
Reserved	
R-0	

15	0
RXMAXLEN	
R/W-5EEh	

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

Table 31-63. Receive Maximum Length Register (RXMAXLEN) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	RXMAXLEN	0-FFFFh	Receive maximum frame length. These bits determine the maximum length of a received frame. The reset value is 5EEh (1518). Frames with byte counts greater than RXMAXLEN are long frames. Long frames with no errors are oversized frames. Long frames with CRC, code, or alignment error are jabber frames.

31.5.25 Receive Buffer Offset Register (RXBUFFEROFFSET)

The receive buffer offset register (RXBUFFEROFFSET) is shown in Figure 31-66 and described in Table 31-64.

Figure 31-66. Receive Buffer Offset Register (RXBUFFEROFFSET)

31 — 16
Reserved
R-0

15 — 0
RXBUFFEROFFSET
R/W-0

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

Table 31-64. Receive Buffer Offset Register (RXBUFFEROFFSET) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	RXBUFFEROFFSET	0-FFFFh	Receive buffer offset value. These bits are written by the EMAC into each frame SOP buffer descriptor Buffer Offset field. The frame data begins after the RXBUFFEROFFSET value of bytes. A value of 0 indicates that there are no unused bytes at the beginning of the data, and that valid data begins on the first byte of the buffer. A value of Fh (15) indicates that the first 15 bytes of the buffer are to be ignored by the EMAC and that valid buffer data starts on byte 16 of the buffer. This value is used for all channels.

31.5.26 Receive Filter Low Priority Frame Threshold Register (RXFILTERLOWTHRESH)

The receive filter low priority frame threshold register (RXFILTERLOWTHRESH) is shown in Figure 31-67 and described in Table 31-65.

Figure 31-67. Receive Filter Low Priority Frame Threshold Register (RXFILTERLOWTHRESH)

31 — 16
Reserved
R-0

15 — 8	7 — 0
Reserved	RXFILTERTHRESH
R-0	R/W-0

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

Table 31-65. Receive Filter Low Priority Frame Threshold Register (RXFILTERLOWTHRESH) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	RXFILTERTHRESH	0-FFh	Receive filter low threshold. These bits contain the free buffer count threshold value for filtering low priority incoming frames. This field should remain 0, if no filtering is desired.

31.5.27 Receive Channel Flow Control Threshold Registers (RX0FLOWTHRESH-RX7FLOWTHRESH)

The receive channel 0-7 flow control threshold register (RX*n*FLOWTHRESH) is shown in Figure 31-68 and described in Table 31-66.

Figure 31-68. Receive Channel *n* Flow Control Threshold Register (RX*n*FLOWTHRESH)

31	16
Reserved	
R-0	

15 8	7 0
Reserved	RX*n*FLOWTHRESH
R-0	R/W-0

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

Table 31-66. Receive Channel *n* Flow Control Threshold Register (RX*n*FLOWTHRESH) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	RX*n*FLOWTHRESH	0-FFh	Receive flow threshold. These bits contain the threshold value for issuing flow control on incoming frames for channel *n* (when enabled).

31.5.28 Receive Channel Free Buffer Count Registers (RX0FREEBUFFER-RX7FREEBUFFER)

The receive channel 0-7 free buffer count register (RX*n*FREEBUFFER) is shown in Figure 31-69 and described in Table 31-67.

Figure 31-69. Receive Channel *n* Free Buffer Count Register (RX*n*FREEBUFFER)

31	16
Reserved	
R-0	

15	0
RX*n*FREEBUF	
WI-0	

LEGEND: R = Read only; WI = Write to increment; -*n* = value after reset

Table 31-67. Receive Channel *n* Free Buffer Count Register (RX*n*FREEBUFFER) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	RX*n*FREEBUF	0-FFh	Receive free buffer count. These bits contain the count of free buffers available. The RXFILTERTHRESH value is compared with this field to determine if low priority frames should be filtered. The RX*n*FLOWTHRESH value is compared with this field to determine if receive flow control should be issued against incoming packets (if enabled). This is a write-to-increment field. This field rolls over to 0 on overflow. If hardware flow control or QOS is used, the host must initialize this field to the number of available buffers (one register per channel). The EMAC decrements the associated channel register for each received frame by the number of buffers in the received frame. The host must write this field with the number of buffers that have been freed due to host processing.

31.5.29 MAC Control Register (MACCONTROL)

The MAC control register (MACCONTROL) is shown in Figure 31-70 and described in Table 31-68.

Figure 31-70. MAC Control Register (MACCONTROL)

31	16
Reserved	
R-0	

15	14	13	12	11	10	9	8
RMIISPEED	RXOFFLENBLOCK	RXOWNERSHIP	Rsvd	CMDIDLE	TXSHORTGAPEN	TXPTYPE	Reserved
R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R-0

7	6	5	4	3	2	1	0
Reserved	TXPACE	GMIIEN	TXFLOWEN	RXBUFFERFLOWEN	Reserved	LOOPBACK	FULLDUPLEX
R-0	R/W-0	R/W-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 31-68. MAC Control Register (MACCONTROL) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15	RMIISPEED		RMII interface transmit and receive speed select.
		0	Operate RMII interface in 10 Mbps speed mode.
		1	Operate RMII interface in 100 Mbps speed mode.
14	RXOFFLENBLOCK		Receive offset / length word write block.
		0	Do not block the DMA writes to the receive buffer descriptor offset / buffer length word.
		1	Block all EMAC DMA controller writes to the receive buffer descriptor offset / buffer length words during packet processing. When this bit is set, the EMAC will never write the third word to any receive buffer descriptor.
13	RXOWNERSHIP		Receive ownership write bit value.
		0	The EMAC writes the Receive ownership bit to 0 at the end of packet processing.
		1	The EMAC writes the Receive ownership bit to 1 at the end of packet processing. If you do not use the ownership mechanism, you can set this mode to preclude the necessity of software having to set this bit each time the buffer descriptor is used.
12	Reserved	0	Reserved
11	CMDIDLE		Command Idle bit
		0	Idle is not commanded.
		1	Idle is commanded (read IDLE in the MACSTATUS register).
10	TXSHORTGAPEN		Transmit Short Gap Enable
		0	Transmit with a short IPG is disabled. Normal 96-bit time IPG is inserted between packets.
		1	Transmit with a short IPG is enabled. Shorter 88-bit time IPG is inserted between packets.
9	TXPTYPE		Transmit queue priority type
		0	The queue uses a round-robin scheme to select the next channel for transmission.
		1	The queue uses a fixed-priority (channel 7 highest priority) scheme to select the next channel for transmission.
8-7	Reserved	0	Reserved
6	TXPACE		Transmit pacing enable bit
		0	Transmit pacing is disabled.
		1	Transmit pacing is enabled.

Table 31-68. MAC Control Register (MACCONTROL) Field Descriptions (continued)

Bit	Field	Value	Description
5	GMIIEN		GMII enable bit. This bit must be set before the MAC transmits or receives data in any of the supported interface modes. (for instance, MII, RMII). This bit does not select the interface mode but rather holds or releases the MAC TX and RX state machines from reset.
		0	The MAC RX and TX state machines are held in reset.
		1	The MAC RX and TX state machines are released from reset and transmit and receive are enabled.
4	TXFLOWEN		Transmit flow control enable bit. This bit determines if incoming pause frames are acted upon in full-duplex mode. Incoming pause frames are not acted upon in half-duplex mode, regardless of this bit setting. The RXMBPENABLE bits determine whether or not received pause frames are transferred to memory.
		0	Transmit flow control is disabled. Full-duplex mode: incoming pause frames are not acted upon.
		1	Transmit flow control is enabled. Full-duplex mode: incoming pause frames are acted upon.
3	RXBUFFERFLOWEN		Receive buffer flow control enable bit
		0	Receive flow control is disabled. Half-duplex mode: no flow control generated collisions are sent. Full-duplex mode: no outgoing pause frames are sent.
		1	Receive flow control is enabled. Half-duplex mode: collisions are initiated when receive buffer flow control is triggered. Full-duplex mode: outgoing pause frames are sent when receive flow control is triggered.
2	Reserved	0	Reserved
1	LOOPBACK		Loopback mode. The loopback mode forces internal full-duplex mode regardless of the FULLDUPLEX bit. The loopback bit should be changed only when GMIIEN bit is deasserted.
		0	Loopback mode is disabled.
		1	Loopback mode is enabled.
0	FULLDUPLEX		Full duplex mode.
		0	Half-duplex mode is enabled.
		1	Full-duplex mode is enabled.

31.5.30 MAC Status Register (MACSTATUS)

The MAC status register (MACSTATUS) is shown in Figure 31-71 and described in Table 31-69.

Figure 31-71. MAC Status Register (MACSTATUS)

31	30 24	23 20	19	18 16
IDLE	Reserved	TXERRCODE	Rsvd	TXERRCH
R-0	R-0	R-0	R-0	R-0

15 12	11	10 8
RXERRCODE	Reserved	RXERRCH
R-0	R-0	R-0

7 3	2	1	0
Reserved	RXQOSACT	RXFLOWACT	TXFLOWACT
R-0	R-0	R-0	R-0

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

Table 31-69. MAC Status Register (MACSTATUS) Field Descriptions

Bit	Field	Value	Description
31	IDLE		EMAC idle bit. This bit is cleared to 0 at reset; one clock after reset, it goes to 1.
		0	The EMAC is not idle.
		1	The EMAC is in the idle state.
30-24	Reserved	0	Reserved
23-20	TXERRCODE		Transmit host error code. These bits indicate that EMAC detected transmit DMA related host errors. The host should read this field after a host error interrupt (HOSTPEND) to determine the error. Host error interrupts require hardware reset in order to recover. A 0 packet length is an error, but it is not detected.
		0	No error.
		1h	SOP error; the buffer is the first buffer in a packet, but the SOP bit is not set in software.
		2h	Ownership bit is not set in SOP buffer.
		3h	Zero next buffer descriptor pointer without EOP.
		4h	Zero buffer pointer.
		5h	Zero buffer length.
		6h	Packet length error (sum of buffers is less than packet length).
		7h-Fh	Reserved
19	Reserved	0	Reserved
18-16	TXERRCH		Transmit host error channel. These bits indicate which transmit channel the host error occurred on. This field is cleared to 0 on a host read.
		0	The host error occurred on transmit channel 0.
		1h	The host error occurred on transmit channel 1.
		2h	The host error occurred on transmit channel 2.
		3h	The host error occurred on transmit channel 3.
		4h	The host error occurred on transmit channel 4.
		5h	The host error occurred on transmit channel 5.
		6h	The host error occurred on transmit channel 6.
		7h	The host error occurred on transmit channel 7.

Table 31-69. MAC Status Register (MACSTATUS) Field Descriptions (continued)

Bit	Field	Value	Description
15-12	RXERRCODE		Receive host error code. These bits indicate that EMAC detected receive DMA related host errors. The host should read this field after a host error interrupt (HOSTPEND) to determine the error. Host error interrupts require hardware reset in order to recover.
		0	No error
		1h	Reserved
		2h	Ownership bit is not set in SOP buffer.
		3h	Reserved
		4h	Zero buffer pointer
		5h-Fh	Reserved
11	Reserved	0	Reserved
10-8	RXERRCH		Receive host error channel. These bits indicate which receive channel the host error occurred on. This field is cleared to 0 on a host read.
		0	The host error occurred on receive channel 0.
		1h	The host error occurred on receive channel 1.
		2h	The host error occurred on receive channel 2.
		3h	The host error occurred on receive channel 3.
		4h	The host error occurred on receive channel 4.
		5h	The host error occurred on receive channel 5.
		6h	The host error occurred on receive channel 6.
		7h	The host error occurred on receive channel 7.
7-3	Reserved	0	Reserved
2	RXQOSACT		Receive Quality of Service (QOS) active bit. When asserted, indicates that receive quality of service is enabled and that at least one channel freebuffer count (RX*n*FREEBUFFER) is less than or equal to the RXFILTERLOWTHRESH value.
		0	Receive quality of service is disabled.
		1	Receive quality of service is enabled.
1	RXFLOWACT		Receive flow control active bit. When asserted, at least one channel freebuffer count (RX*n*FREEBUFFER) is less than or equal to the channel's corresponding RX*n*FILTERTHRESH value.
		0	Receive flow control is inactive.
		1	Receive flow control is active.
0	TXFLOWACT		Transmit flow control active bit. When asserted, this bit indicates that the pause time period is being observed for a received pause frame. No new transmissions will begin while this bit is asserted, except for the transmission of pause frames. Any transmission in progress when this bit is asserted will complete.
		0	Transmit flow control is inactive.
		1	Transmit flow control is active.

31.5.31 Emulation Control Register (EMCONTROL)

The emulation control register (EMCONTROL) is shown in Figure 31-72 and described in Table 31-70.

Figure 31-72. Emulation Control Register (EMCONTROL)

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 31-70. Emulation Control Register (EMCONTROL) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1	SOFT		Emulation soft bit. This bit is used in conjunction with FREE bit to determine the emulation suspend mode. This bit has no effect if FREE = 1.
		0	Soft mode is disabled. EMAC stops immediately during emulation halt.
		1	Soft mode is enabled. During emulation halt, EMAC stops after completion of current operation.
0	FREE		Emulation free bit. This bit is used in conjunction with SOFT bit to determine the emulation suspend mode.
		0	Free-running mode is disabled. During emulation halt, SOFT bit determines operation of EMAC.
		1	Free-running mode is enabled. During emulation halt, EMAC continues to operate.

31.5.32 FIFO Control Register (FIFOCONTROL)

The FIFO control register (FIFOCONTROL) is shown in Figure 31-73 and described in Table 31-71.

Figure 31-73. FIFO Control Register (FIFOCONTROL)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		TXCELLTHRESH	
R-0		R/W-2h	

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

Table 31-71. FIFO Control Register (FIFOCONTROL) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved
1-0	TXCELLTHRESH		Transmit FIFO cell threshold. Indicates the number of 64-byte packet cells required to be in the transmit FIFO before the packet transfer is initiated. Packets with fewer cells will be initiated when the complete packet is contained in the FIFO. The default value is 2, but 3 is also valid. 0 and 1 are not valid values.
		0-1h	Not a valid value.
		2h	Two 64-byte packet cells required to be in the transmit FIFO.
		3h	Three 64-byte packet cells required to be in the transmit FIFO.

31.5.33 MAC Configuration Register (MACCONFIG)

The MAC configuration register (MACCONFIG) is shown in Figure 31-74 and described in Table 31-72.

Figure 31-74. MAC Configuration Register (MACCONFIG)

31 … 24	23 … 16
TXCELLDEPTH	RXCELLDEPTH
R-3h	R-3h

15 … 8	7 … 0
ADDRESSTYPE	MACCFIG
R-2h	R-2h

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

Table 31-72. MAC Configuration Register (MACCONFIG) Field Descriptions

Bit	Field	Value	Description
31-24	TXCELLDEPTH	3h	Transmit cell depth. These bits indicate the number of cells in the transmit FIFO.
23-16	RXCELLDEPTH	3h	Receive cell depth. These bits indicate the number of cells in the receive FIFO.
15-8	ADDRESSTYPE	2h	Address type
7-0	MACCFIG	2h	MAC configuration value

31.5.34 Soft Reset Register (SOFTRESET)

The soft reset register (SOFTRESET) is shown in Figure 31-75 and described in Table 31-73.

Figure 31-75. Soft Reset Register (SOFTRESET)

31 … 16
Reserved
R-0

15 … 1	0
Reserved	SOFTRESET
R-0	R/W-0

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

Table 31-73. Soft Reset Register (SOFTRESET) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	SOFTRESET		Software reset. Writing a 1 to this bit causes the EMAC logic to be reset. Software reset occurs when the receive and transmit DMA controllers are in an idle state to avoid locking up the Configuration bus. After writing a 1 to this bit, it may be polled to determine if the reset has occurred. If a 1 is read, the reset has not yet occurred. If a 0 is read, then a reset has occurred.
		0	A software reset has not occurred.
		1	A software reset has occurred.

31.5.35 MAC Source Address Low Bytes Register (MACSRCADDRLO)

The MAC source address low bytes register (MACSRCADDRLO) is shown in Figure 31-76 and described in Table 31-74.

Figure 31-76. MAC Source Address Low Bytes Register (MACSRCADDRLO)

31	16
Reserved	
R-0	

15	8	7	0
MACSRCADDR0		MACSRCADDR1	
R/W-0		R/W-0	

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

Table 31-74. MAC Source Address Low Bytes Register (MACSRCADDRLO) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-8	MACSRCADDR0	0-FFh	MAC source address lower 8-0 bits (byte 0)
7-0	MACSRCADDR1	0-FFh	MAC source address bits 15-8 (byte 1)

31.5.36 MAC Source Address High Bytes Register (MACSRCADDRHI)

The MAC source address high bytes register (MACSRCADDRHI) is shown in Figure 31-77 and described in Table 31-75.

Figure 31-77. MAC Source Address High Bytes Register (MACSRCADDRHI)

31	24	23	16
MACSRCADDR2		MACSRCADDR3	
R/W-0		R/W-0	

15	8	7	0
MACSRCADDR4		MACSRCADDR5	
R/W-0		R/W-0	

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

Table 31-75. MAC Source Address High Bytes Register (MACSRCADDRHI) Field Descriptions

Bit	Field	Value	Description
31-24	MACSRCADDR2	0-FFh	MAC source address bits 23-16 (byte 2)
23-16	MACSRCADDR3	0-FFh	MAC source address bits 31-24 (byte 3)
15-8	MACSRCADDR4	0-FFh	MAC source address bits 39-32 (byte 4)
7-0	MACSRCADDR5	0-FFh	MAC source address bits 47-40 (byte 5)

31.5.37 MAC Hash Address Register 1 (MACHASH1)

The MAC hash registers allow group addressed frames to be accepted on the basis of a hash function of the address. The hash function creates a 6-bit data value (Hash_fun) from the 48-bit destination address (DA) as follows:

```
Hash_fun(0)=DA(0) XOR DA(6) XOR DA(12) XOR DA(18) XOR DA(24) XOR DA(30) XOR DA(36) XOR DA(42);

Hash_fun(1)=DA(1) XOR DA(7) XOR DA(13) XOR DA(19) XOR DA(25) XOR DA(31) XOR DA(37) XOR DA(43);

Hash_fun(2)=DA(2) XOR DA(8) XOR DA(14) XOR DA(20) XOR DA(26) XOR DA(32) XOR DA(38) XOR DA(44);

Hash_fun(3)=DA(3) XOR DA(9) XOR DA(15) XOR DA(21) XOR DA(27) XOR DA(33) XOR DA(39) XOR DA(45);

Hash_fun(4)=DA(4) XOR DA(10) XOR DA(16) XOR DA(22) XOR DA(28) XOR DA(34) XOR DA(40) XOR DA(46);

Hash_fun(5)=DA(5) XOR DA(11) XOR DA(17) XOR DA(23) XOR DA(29) XOR DA(35) XOR DA(41) XOR DA(47);
```

This function is used as an offset into a 64-bit hash table stored in MACHASH1 and MACHASH2 that indicates whether a particular address should be accepted or not.

The MAC hash address register 1 (MACHASH1) is shown in Figure 31-78 and described in Table 31-76.

Figure 31-78. MAC Hash Address Register 1 (MACHASH1)

31	0
MACHASH1	
R/W-0	

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

Table 31-76. MAC Hash Address Register 1 (MACHASH1) Field Descriptions

Bit	Field	Value	Description
31-0	MACHASH1	0-FFFF FFFFh	Least-significant 32 bits of the hash table corresponding to hash values 0 to 31. If a hash table bit is set, then a group address that hashes to that bit index is accepted.

31.5.38 MAC Hash Address Register 2 (MACHASH2)

The MAC hash address register 2 (MACHASH2) is shown in Figure 31-79 and described in Table 31-77.

Figure 31-79. MAC Hash Address Register 2 (MACHASH2)

31	0
MACHASH2	
R/W-0	

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

Table 31-77. MAC Hash Address Register 2 (MACHASH2) Field Descriptions

Bit	Field	Value	Description
31-0	MACHASH2	0-FFFF FFFFh	Most-significant 32 bits of the hash table corresponding to hash values 32 to 63. If a hash table bit is set, then a group address that hashes to that bit index is accepted.

31.5.39 Back Off Test Register (BOFFTEST)

The back off test register (BOFFTEST) is shown in Figure 31-80 and described in Table 31-78.

Figure 31-80. Back Off Random Number Generator Test Register (BOFFTEST)

31 – 26	25 – 16
Reserved	RNDNUM
R-0	R-0

15 – 12	11 – 10	9 – 0
COLLCOUNT	Reserved	TXBACKOFF
R-0	R-0	R-0

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

Table 31-78. Back Off Test Register (BOFFTEST) Field Descriptions

Bit	Field	Value	Description
31-26	Reserved	0	Reserved
25-16	RNDNUM	0-3FFh	Backoff random number generator. This field allows the Backoff Random Number Generator to be read. Reading this field returns the generator's current value. The value is reset to 0 and begins counting on the clock after the deassertion of reset.
15-12	COLLCOUNT	0-Fh	Collision count. These bits indicate the number of collisions the current frame has experienced.
11-10	Reserved	0	Reserved
9-0	TXBACKOFF	0-3FFh	Backoff count. This field allows the current value of the backoff counter to be observed for test purposes. This field is loaded automatically according to the backoff algorithm, and is decremented by one for each slot time after the collision.

31.5.40 Transmit Pacing Algorithm Test Register (TPACETEST)

The transmit pacing algorithm test register (TPACETEST) is shown in Figure 31-81 and described in Table 31-79.

Figure 31-81. Transmit Pacing Algorithm Test Register (TPACETEST)

31 – 16
Reserved
R-0

15 – 5	4 – 0
Reserved	PACEVAL
R-0	R-0

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

Table 31-79. Transmit Pacing Algorithm Test Register (TPACETEST) Field Descriptions

Bit	Field	Value	Description
31-5	Reserved	0	Reserved
4-0	PACEVAL	0-1Fh	Pacing register current value. A nonzero value in this field indicates that transmit pacing is active. A transmit frame collision or deferral causes PACEVAL to be loaded with 1Fh (31); good frame transmissions (with no collisions or deferrals) cause PACEVAL to be decremented down to 0. When PACEVAL is nonzero, the transmitter delays four Inter Packet Gaps between new frame transmissions after each successfully transmitted frame that had no deferrals or collisions. If a transmit frame is deferred or suffers a collision, the IPG time is not stretched to four times the normal value. Transmit pacing helps reduce capture effects, which improves overall network bandwidth.

31.5.41 Receive Pause Timer Register (RXPAUSE)

The receive pause timer register (RXPAUSE) is shown in Figure 31-82 and described in Table 31-80.

Figure 31-82. Receive Pause Timer Register (RXPAUSE)

31	16
Reserved	
R-0	

15	0
PAUSETIMER	
R-0	

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

Table 31-80. Receive Pause Timer Register (RXPAUSE) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	PAUSETIMER	0-FFh	Receive pause timer value. These bits allow the contents of the receive pause timer to be observed. The receive pause timer is loaded with FF00h when the EMAC sends an outgoing pause frame (with pause time of FFFFh). The receive pause timer is decremented at slot time intervals. If the receive pause timer decrements to 0, then another outgoing pause frame is sent and the load/decrement process is repeated.

31.5.42 Transmit Pause Timer Register (TXPAUSE)

The transmit pause timer register (TXPAUSE) is shown in Figure 31-83 and described in Table 31-81.

Figure 31-83. Transmit Pause Timer Register (TXPAUSE)

31	16
Reserved	
R-0	

15	0
PAUSETIMER	
R-0	

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

Table 31-81. Transmit Pause Timer Register (TXPAUSE) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	PAUSETIMER	0-FFh	Transmit pause timer value. These bits allow the contents of the transmit pause timer to be observed. The transmit pause timer is loaded by a received (incoming) pause frame, and then decremented at slot time intervals down to 0, at which time EMAC transmit frames are again enabled.

31.5.43 MAC Address Low Bytes Register (MACADDRLO)

The MAC address low bytes register used in receive address matching (MACADDRLO), is shown in Figure 31-84 and described in Table 31-82.

Figure 31-84. MAC Address Low Bytes Register (MACADDRLO)

31 21	20	19	18 16
Reserved	VALID	MATCHFILT	CHANNEL
R-0	R/W-x	R/W-x	R/W-x

15 8	7 0
MACADDR0	MACADDR1
R/W-x	R/W-x

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

Table 31-82. MAC Address Low Bytes Register (MACADDRLO) Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	0	Reserved
20	VALID		Address valid bit. This bit should be cleared to zero for unused address channels
		0	Address is not valid and will not be used for matching or filtering incoming packets.
		1	Address is valid and will be used for matching or filtering incoming packets.
19	MATCHFILT		Match or filter bit
		0	The address will be used (if the VALID bit is set) to filter incoming packet addresses.
		1	The address will be used (if the VALID bit is set) to match incoming packet addresses.
18-16	CHANNEL	0-7h	Channel select. Determines which receive channel a valid address match will be transferred to. The channel is a don't care if MATCHFILT is cleared to 0.
15-8	MACADDR0	0-FFh	MAC address lower 8-0 bits (byte 0)
7-0	MACADDR1	0-FFh	MAC address bits 15-8 (byte 1)

31.5.44 MAC Address High Bytes Register (MACADDRHI)

The MAC address high bytes register used in receive address matching (MACADDRHI) is shown in Figure 31-85 and described in Table 31-83.

Figure 31-85. MAC Address High Bytes Register (MACADDRHI)

31 24	23 16
MACADDR2	MACADDR3
R/W-x	R/W-x

15 8	7 0
MACADDR4	MACADDR5
R/W-x	R/W-x

LEGEND: R/W = Read/Write; -x = value is indeterminate after reset

Table 31-83. MAC Address High Bytes Register (MACADDRHI) Field Descriptions

Bit	Field	Value	Description
31-24	MACADDR2	0-FFh	MAC source address bits 23-16 (byte 2)
23-16	MACADDR3	0-FFh	MAC source address bits 31-24 (byte 3)
15-8	MACADDR4	0-FFh	MAC source address bits 39-32 (byte 4)
7-0	MACADDR5	0-FFh	MAC source address bits 47-40 (byte 5). Bit 40 is the group bit. It is forced to 0 and read as 0. Therefore, only unicast addresses are represented in the address table.

31.5.45 MAC Index Register (MACINDEX)

The MAC index register (MACINDEX) is shown in Figure 31-86 and described in Table 31-84.

Figure 31-86. MAC Index Register (MACINDEX)

31 16
Reserved
R-0

15 3	2 0
Reserved	MACINDEX
R-0	R/W-0

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

Table 31-84. MAC Index Register (MACINDEX) Field Descriptions

Bit	Field	Value	Description
31-3	Reserved	0	Reserved
2-0	MACINDEX	0-7h	MAC address index. All eight addresses share the upper 40 bits. Only the lower byte is unique for each address. An address is written by first writing the address number (channel) into the MACINDEX register. The upper 32 bits of the address are then written to the MACADDRHI register, which is followed by writing the lower 16 bits of the address to the MACADDRLO register. Since all eight addresses share the upper 40 bits of the address, the MACADDRHI register only needs to be written the first time.

31.5.46 Transmit Channel DMA Head Descriptor Pointer Registers (TX0HDP-TX7HDP)

The transmit channel 0-7 DMA head descriptor pointer register (TX*n*HDP) is shown in Figure 31-87 and described in Table 31-85.

Figure 31-87. Transmit Channel *n* DMA Head Descriptor Pointer Register (TX*n*HDP)

31 … 0
TX*n*HDP
R/W-x

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

Table 31-85. Transmit Channel *n* DMA Head Descriptor Pointer Register (TX*n*HDP) Field Descriptions

Bit	Field	Value	Description
31-0	TX*n*HDP	0-FFFF FFFFh	Transmit channel *n* DMA Head Descriptor pointer. Writing a transmit DMA buffer descriptor address to a head pointer location initiates transmit DMA operations in the queue for the selected channel. Writing to these locations when they are nonzero is an error (except at reset). Host software must initialize these locations to 0 on reset.

31.5.47 Receive Channel DMA Head Descriptor Pointer Registers (RX0HDP-RX7HDP)

The receive channel 0-7 DMA head descriptor pointer register (RX*n*HDP) is shown in Figure 31-88 and described in Table 31-86.

Figure 31-88. Receive Channel *n* DMA Head Descriptor Pointer Register (RX*n*HDP)

31 … 0
RX*n*HDP
R/W-x

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

Table 31-86. Receive Channel *n* DMA Head Descriptor Pointer Register (RX*n*HDP) Field Descriptions

Bit	Field	Value	Description
31-0	RX*n*HDP	0-FFFF FFFFh	Receive channel *n* DMA Head Descriptor pointer. Writing a receive DMA buffer descriptor address to this location allows receive DMA operations in the selected channel when a channel frame is received. Writing to these locations when they are nonzero is an error (except at reset). Host software must initialize these locations to 0 on reset.

31.5.48 Transmit Channel Completion Pointer Registers (TX0CP-TX7CP)

The transmit channel 0-7 completion pointer register (TX*n*CP) is shown in Figure 31-89 and described in Table 31-87.

Figure 31-89. Transmit Channel *n* Completion Pointer Register (TX*n*CP)

31	0
TX*n*CP	
R/W-x	

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

Table 31-87. Transmit Channel *n* Completion Pointer Register (TX*n*CP) Field Descriptions

Bit	Field	Value	Description
31-0	TX*n*CP	0-FFFF FFFFh	Transmit channel *n* completion pointer register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing. The EMAC uses the value written to determine if the interrupt should be deasserted.

31.5.49 Receive Channel Completion Pointer Registers (RX0CP-RX7CP)

The receive channel 0-7 completion pointer register (RX*n*CP) is shown in Figure 31-90 and described in Table 31-88.

Figure 31-90. Receive Channel *n* Completion Pointer Register (RX*n*CP)

31	0
RX*n*CP	
R/W-x	

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

Table 31-88. Receive Channel *n* Completion Pointer Register (RX*n*CP) Field Descriptions

Bit	Field	Value	Description
31-0	RX*n*CP	0-FFFF FFFFh	Receive channel *n* completion pointer register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing. The EMAC uses the value written to determine if the interrupt should be deasserted.

31.5.50 Network Statistics Registers

The EMAC has a set of statistics that record events associated with frame traffic. The statistics values are cleared to zero 38 clocks after the rising edge of reset. When the GMIIEN bit in the MACCONTROL register is set, all statistics registers (see Figure 31-91) are write-to-decrement. The value written is subtracted from the register value with the result stored in the register. If a value greater than the statistics value is written, then zero is written to the register (writing FFFF FFFFh clears a statistics location). When the GMIIEN bit is cleared, all statistics registers are read/write (normal write direct, so writing 0000 0000h clears a statistics location). All write accesses must be 32-bit accesses.

The statistics interrupt (STATPEND) is issued, if enabled, when any statistics value is greater than or equal to 8000 0000h. The statistics interrupt is removed by writing to decrement any statistics value greater than 8000 0000h. The statistics are mapped into internal memory space and are 32-bits wide. All statistics rollover from FFFF FFFFh to 0000 0000h.

Figure 31-91. Statistics Register

31 - 0
COUNT
R/WD-0

LEGEND: R/W = Read/Write; WD = Write to decrement; -*n* = value after reset

31.5.50.1 Good Receive Frames Register (RXGOODFRAMES)

The total number of good frames received on the EMAC. A good frame is defined as having all of the following:

- Any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of length 64 to RXMAXLEN bytes inclusive
- Had no CRC error, alignment error, or code error

See Section 31.2.6.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

31.5.50.2 Broadcast Receive Frames Register (RXBCASTFRAMES)

The total number of good broadcast frames received on the EMAC. A good broadcast frame is defined as having all of the following:

- Any data or MAC control frame that was destined for address FF-FF-FF-FF-FF-FFh only
- Was of length 64 to RXMAXLEN bytes inclusive
- Had no CRC error, alignment error, or code error

See Section 31.2.6.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

31.5.50.3 Multicast Receive Frames Register (RXMCASTFRAMES)

The total number of good multicast frames received on the EMAC. A good multicast frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any multicast address other than FF-FF-FF-FF-FF-FFh
- Was of length 64 to RXMAXLEN bytes inclusive
- Had no CRC error, alignment error, or code error

See Section 31.2.6.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

31.5.50.4 Pause Receive Frames Register (RXPAUSEFRAMES)

The total number of IEEE 802.3X pause frames received by the EMAC (whether acted upon or not). A pause frame is defined as having all of the following:

- Contained any unicast, broadcast, or multicast address
- Contained the length/type field value 88.08h and the opcode 0001h
- Was of length 64 to RXMAXLEN bytes inclusive
- Had no CRC error, alignment error, or code error
- Pause-frames had been enabled on the EMAC (TXFLOWEN bit is set in MACCONTROL).

The EMAC could have been in either half-duplex or full-duplex mode. See Section 31.2.6.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

31.5.50.5 Receive CRC Errors Register (RXCRCERRORS)

The total number of frames received on the EMAC that experienced a CRC error. A frame with CRC errors is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of length 64 to RXMAXLEN bytes inclusive
- Had no alignment or code error
- Had a CRC error. A CRC error is defined as having all of the following:
  - A frame containing an even number of nibbles
  - Fails the frame check sequence test

See Section 31.2.6.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

31.5.50.6 Receive Alignment/Code Errors Register (RXALIGNCODEERRORS)

The total number of frames received on the EMAC that experienced an alignment error or code error. Such a frame is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of length 64 to RXMAXLEN bytes inclusive
- Had either an alignment error or a code error
  - An alignment error is defined as having all of the following:
    - A frame containing an odd number of nibbles
    - Fails the frame check sequence test, if the final nibble is ignored
  - A code error is defined as a frame that has been discarded because the EMACs MII_RXER pin is driven with a one for at least one bit-time's duration at any point during the frame's reception.

Overruns have no effect on this statistic.

CRC alignment or code errors can be calculated by summing receive alignment errors, receive code errors, and receive CRC errors.

31.5.50.7 Receive Oversized Frames Register (RXOVERSIZED)

The total number of oversized frames received on the EMAC. An oversized frame is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was greater than RXMAXLEN in bytes
- Had no CRC error, alignment error, or code error

See Section 31.2.6.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

31.5.50.8 Receive Jabber Frames Register (RXJABBER)

The total number of jabber frames received on the EMAC. A jabber frame is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was greater than RXMAXLEN bytes long
- Had a CRC error, alignment error, or code error

See Section 31.2.6.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

31.5.50.9 Receive Undersized Frames Register (RXUNDERSIZED)

The total number of undersized frames received on the EMAC. An undersized frame is defined as having all of the following:

- Was any data frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was less than 64 bytes long
- Had no CRC error, alignment error, or code error

See Section 31.2.6.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

31.5.50.10 Receive Frame Fragments Register (RXFRAGMENTS)

The total number of frame fragments received on the EMAC. A frame fragment is defined as having all of the following:

- Any data frame (address matching does not matter)
- Was less than 64 bytes long
- Had a CRC error, alignment error, or code error
- Was not the result of a collision caused by half duplex, collision based flow control

See Section 31.2.6.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

31.5.50.11 Filtered Receive Frames Register (RXFILTERED)

The total number of frames received on the EMAC that the EMAC address matching process indicated should be discarded. Such a frame is defined as having all of the following:

- Was any data frame (not MAC control frame) destined for any unicast, broadcast, or multicast address
- Did not experience any CRC error, alignment error, code error
- The address matching process decided that the frame should be discarded (filtered) because it did not match the unicast, broadcast, or multicast address, and it did not match due to promiscuous mode.

To determine the number of receive frames discarded by the EMAC for any reason, sum the following statistics (promiscuous mode disabled):

- Receive fragments
- Receive undersized frames
- Receive CRC errors
- Receive alignment/code errors
- Receive jabbers
- Receive overruns
- Receive filtered frames

This may not be an exact count because the receive overruns statistic is independent of the other statistics, so if an overrun occurs at the same time as one of the other discard reasons, then the above sum double-counts that frame.

31.5.50.12 Receive QOS Filtered Frames Register (RXQOSFILTERED)

The total number of frames received on the EMAC that were filtered due to receive quality of service (QOS) filtering. Such a frame is defined as having all of the following:

- Any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- The frame destination channel flow control threshold register (RX*n*FLOWTHRESH) value was greater than or equal to the channel's corresponding free buffer register (RX*n*FREEBUFFER) value
- Was of length 64 to RXMAXLEN
- RXQOSEN bit is set in RXMBPENABLE
- Had no CRC error, alignment error, or code error

See Section 31.2.6.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

31.5.50.13 Receive Octet Frames Register (RXOCTETS)

The total number of bytes in all good frames received on the EMAC. A good frame is defined as having all of the following:

- Any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of length 64 to RXMAXLEN bytes inclusive
- Had no CRC error, alignment error, or code error

See Section 31.2.6.5 for definitions of alignment, code, and CRC errors. Overruns have no effect on this statistic.

31.5.50.14 Good Transmit Frames Register (TXGOODFRAMES)

The total number of good frames transmitted on the EMAC. A good frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Was any length
- Had no late or excessive collisions, no carrier loss, and no underrun

31.5.50.15 Broadcast Transmit Frames Register (TXBCASTFRAMES)

The total number of good broadcast frames transmitted on the EMAC. A good broadcast frame is defined as having all of the following:

- Any data or MAC control frame destined for address FF-FF-FF-FF-FF-FFh only
- Was of any length
- Had no late or excessive collisions, no carrier loss, and no underrun

31.5.50.16 Multicast Transmit Frames Register (TXMCASTFRAMES)

The total number of good multicast frames transmitted on the EMAC. A good multicast frame is defined as having all of the following:

- Any data or MAC control frame destined for any multicast address other than FF-FF-FF-FF-FF-FFh
- Was of any length
- Had no late or excessive collisions, no carrier loss, and no underrun

31.5.50.17 Pause Transmit Frames Register (TXPAUSEFRAMES)

The total number of IEEE 802.3X pause frames transmitted by the EMAC. Pause frames cannot underrun or contain a CRC error because they are created in the transmitting MAC, so these error conditions have no effect on this statistic. Pause frames sent by software are not included in this count. Since pause frames are only transmitted in full-duplex mode, carrier loss and collisions have no effect on this statistic.

Transmitted pause frames are always 64-byte multicast frames so appear in the multicast transmit frames register and 64 octect frames register statistics.

31.5.50.18 Deferred Transmit Frames Register (TXDEFERRED)

The total number of frames transmitted on the EMAC that first experienced deferment. Such a frame is defined as having all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced no collisions before being successfully transmitted
- Found the medium busy when transmission was first attempted, so had to wait.

CRC errors have no effect on this statistic.

31.5.50.19 Transmit Collision Frames Register (TXCOLLISION)

The total number of times that the EMAC experienced a collision. Collisions occur under two circumstances:

- When a transmit data or MAC control frame has all of the following:
  - Was destined for any unicast, broadcast, or multicast address
  - Was any size
  - Had no carrier loss and no underrun
  - Experienced a collision. A jam sequence is sent for every non-late collision, so this statistic increments on each occasion if a frame experiences multiple collisions (and increments on late collisions).
- When the EMAC is in half-duplex mode, flow control is active, and a frame reception begins.

CRC errors have no effect on this statistic.

31.5.50.20 Transmit Single Collision Frames Register (TXSINGLECOLL)

The total number of frames transmitted on the EMAC that experienced exactly one collision. Such a frame is defined as having all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced one collision before successful transmission. The collision was not late.

CRC errors have no effect on this statistic.

31.5.50.21 Transmit Multiple Collision Frames Register (TXMULTICOLL)

The total number of frames transmitted on the EMAC that experienced multiple collisions. Such a frame is defined as having all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced 2 to 15 collisions before being successfully transmitted. None of the collisions were late.

CRC errors have no effect on this statistic.

31.5.50.22 Transmit Excessive Collision Frames Register (TXEXCESSIVECOLL)

The total number of frames when transmission was abandoned due to excessive collisions. Such a frame is defined as having all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced 16 collisions before abandoning all attempts at transmitting the frame. None of the collisions were late.

CRC errors have no effect on this statistic.

31.5.50.23 Transmit Late Collision Frames Register (TXLATECOLL)

The total number of frames when transmission was abandoned due to a late collision. Such a frame is defined as having all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced a collision later than 512 bit-times into the transmission. There may have been up to 15 previous (non-late) collisions that had previously required the transmission to be reattempted. The late collisions statistic dominates over the single, multiple, and excessive collisions statistics. If a late collision occurs, the frame is not counted in any of these other three statistics.

CRC errors, carrier loss, and underrun have no effect on this statistic.

31.5.50.24 Transmit Underrun Error Register (TXUNDERRUN)

The number of frames sent by the EMAC that experienced FIFO underrun. Late collisions, CRC errors, carrier loss, and underrun have no effect on this statistic.

31.5.50.25 Transmit Carrier Sense Errors Register (TXCARRIERSENSE)

The total number of frames on the EMAC that experienced carrier loss. Such a frame is defined as having all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address
- Was any size
- The carrier sense condition was lost or never asserted when transmitting the frame (the frame is not retransmitted)

CRC errors and underrun have no effect on this statistic.

31.5.50.26 Transmit Octet Frames Register (TXOCTETS)

The total number of bytes in all good frames transmitted on the EMAC. A good frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Was any length
- Had no late or excessive collisions, no carrier loss, and no underrun

31.5.50.27 Transmit and Receive 64 Octet Frames Register (FRAME64)

The total number of 64-byte frames received and transmitted on the EMAC. Such a frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Did not experience late collisions, excessive collisions, underrun, or carrier sense error
- Was exactly 64-bytes long. (If the frame was being transmitted and experienced carrier loss that resulted in a frame of this size being transmitted, then the frame is recorded in this statistic).

CRC errors, alignment/code errors, and overruns do not affect the recording of frames in this statistic.

31.5.50.28 Transmit and Receive 65 to 127 Octet Frames Register (FRAME65T127)

The total number of 65-byte to 127-byte frames received and transmitted on the EMAC. Such a frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Did not experience late collisions, excessive collisions, underrun, or carrier sense error
- Was 65-bytes to 127-bytes long

CRC errors, alignment/code errors, underruns, and overruns do not affect the recording of frames in this statistic.

31.5.50.29 Transmit and Receive 128 to 255 Octet Frames Register (FRAME128T255)

The total number of 128-byte to 255-byte frames received and transmitted on the EMAC. Such a frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Did not experience late collisions, excessive collisions, underrun, or carrier sense error
- Was 128-bytes to 255-bytes long

CRC errors, alignment/code errors, underruns, and overruns do not affect the recording of frames in this statistic.

31.5.50.30 Transmit and Receive 256 to 511 Octet Frames Register (FRAME256T511)

The total number of 256-byte to 511-byte frames received and transmitted on the EMAC. Such a frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Did not experience late collisions, excessive collisions, underrun, or carrier sense error
- Was 256-bytes to 511-bytes long

CRC errors, alignment/code errors, underruns, and overruns do not affect the recording of frames in this statistic.

31.5.50.31 Transmit and Receive 512 to 1023 Octet Frames Register (FRAME512T1023)

The total number of 512-byte to 1023-byte frames received and transmitted on the EMAC. Such a frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Did not experience late collisions, excessive collisions, underrun, or carrier sense error
- Was 512-bytes to 1023-bytes long

CRC errors, alignment/code errors, and overruns do not affect the recording of frames in this statistic.

31.5.50.32 Transmit and Receive 1024 to RXMAXLEN Octet Frames Register (FRAME1024TUP)

The total number of 1024-byte to RXMAXLEN-byte frames received and transmitted on the EMAC. Such a frame is defined as having all of the following:

- Any data or MAC control frame that was destined for any unicast, broadcast, or multicast address
- Did not experience late collisions, excessive collisions, underrun, or carrier sense error
- Was 1024-bytes to RXMAXLEN-bytes long

CRC/alignment/code errors, underruns, and overruns do not affect frame recording in this statistic.

31.5.50.33 Network Octet Frames Register (NETOCTETS)

The total number of bytes of frame data received and transmitted on the EMAC. Each frame counted has all of the following:

- Was any data or MAC control frame destined for any unicast, broadcast, or multicast address (address match does not matter)
- Was of any size (including less than 64-byte and greater than RXMAXLEN-byte frames)

Also counted in this statistic is:

- Every byte transmitted before a carrier-loss was experienced
- Every byte transmitted before each collision was experienced (multiple retries are counted each time)
- Every byte received if the EMAC is in half-duplex mode until a jam sequence was transmitted to initiate flow control. (The jam sequence is not counted to prevent double-counting).

Error conditions such as alignment errors, CRC errors, code errors, overruns, and underruns do not affect the recording of bytes in this statistic. The objective of this statistic is to give a reasonable indication of Ethernet utilization.

31.5.50.34 Receive FIFO or DMA Start of Frame Overruns Register (RXSOFOVERRUNS)

The total number of frames received on the EMAC that had either a FIFO or DMA start of frame (SOF) overrun. An SOF overrun frame is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of any size (including less than 64-byte and greater than RXMAXLEN-byte frames)
- The EMAC was unable to receive it because it did not have the resources to receive it (cell FIFO full or no DMA buffer available at the start of the frame).

CRC errors, alignment errors, and code errors have no effect on this statistic.

31.5.50.35 Receive FIFO or DMA Middle of Frame Overruns Register (RXMOFOVERRUNS)

The total number of frames received on the EMAC that had either a FIFO or DMA middle of frame (MOF) overrun. An MOF overrun frame is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of any size (including less than 64-byte and greater than RXMAXLEN-byte frames)
- The EMAC was unable to receive it because it did not have the resources to receive it (cell FIFO full or no DMA buffer available after the frame was successfully started - no SOF overrun).

CRC errors, alignment errors, and code errors have no effect on this statistic.

31.5.50.36 Receive DMA Overruns Register (RXDMAOVERRUNS)

The total number of frames received on the EMAC that had either a DMA start of frame (SOF) overrun or a DMA middle of frame (MOF) overrun. A receive DMA overrun frame is defined as having all of the following:

- Was any data or MAC control frame that matched a unicast, broadcast, or multicast address, or matched due to promiscuous mode
- Was of any size (including less than 64-byte and greater than RXMAXLEN-byte frames)
- The EMAC was unable to receive it because it did not have the DMA buffer resources to receive it (zero head descriptor pointer at the start or during the middle of the frame reception).

CRC errors, alignment errors, and code errors have no effect on this statistic.

Chapter 32
SPNU514C–March 2018

Universal Serial Bus (USB)

This chapter describes the universal serial bus (USB) controller on the microcontroller.

Topic **Page**

32.1 Overview 1817
32.2 USB Host Controller 1817
32.3 USB Device Controller 1844
32.4 USB Connectivity 1938

32.1 Overview

The microcontroller provides several varieties of USB functionality, including:

- USB host: device provides a two-port *USB Specification Revision 1.1* - compliant host controller, which is based on the *OHCI Specification for USB Release 1.0a*
- USB device: device provides a full-speed USB device
- Support for up to 2 active simultaneous ports (1 Function and 1 Host, or 2 Hosts)

32.2 USB Host Controller

The USB host controller (HC) is a two-port controller that communicates with USB devices at the USB low-speed (1.5M bit-per-second maximum) and full-speed (12M bit-per-second maximum) data rates. It is compatible with the *Universal Serial Bus Specification Revision 2.0* and the *Open HCI—Open Host Controller Interface Specification for USB,* Release 1.0a*,* available through the Compaq Computer Corporation web site, and hereafter called the *OHCI Specification for USB.* It is assumed that users of the USB host controller are already familiar with the *USB Specification* and *OHCI Specification for USB*.

The USB host controller implements the register set and makes use of the memory data structures defined in the *OHCI Specification for USB*. These registers and data structures are the mechanism by which a USB host controller driver software package can control the USB host controller. The *OHCI Specification for USB* also defines how the USB host controller implementation must interact with those registers and data structures in system memory. The device CPU accesses these registers via the module's peripheral bus.

To reduce processor software and interrupt overhead, the USB host controller generates USB traffic based on data structures and data buffers stored in system memory. The USB host controller accesses these data structures without direct intervention by the processor using the master access (bus transaction initiator) port. These data structures and data buffers can be located in internal or external RAM.

The USB host controller provides an interrupt to the Vectored Interrupt Manager (VIM) to signal certain hardware events to the host controller driver software.

32.2.1 USB Open Host Controller Interface Functionality

32.2.1.1 OHCI Controller Overview

The *Open HCI—Open Host Controller Interface Specification for USB*, Release 1.0a, defines a set of registers and data structures stored in system memory that define how a USB host controller interfaces to system software. This specification, in conjunction with the *Universal Serial Bus Specification Version 2.0*, defines most of the USB functionality that the USB host controller provides.

The *OHCI Specification for USB* focuses on two main aspects of the hardware implementation of a USB host controller: its register set and the memory data structures that define the activity to appear on the USB bus. Also discussed are issues such as interrupt generation, USB host controller state, USB frame management, and the methods that the hardware must use to process the lists of data structures in system memory.

This document does not duplicate the information presented in the *OHCI Specification for USB* or the *USB Specification*. USB host controller users can refer to the *USB Specification* and the *OHCI Specification for USB* for detailed discussions of USB requirements and OHCI controller operation.

32.2.2 USB Host Controller Differences From OHCI Specification for USB

The USB host controller implementation does not implement every aspect of the functionality defined in the *OHCI Specification for USB*. The differences focus on the OHCI ownership change interrupt. Other restrictions are imposed by the effects of the device pin multiplexing options.

32.2.2.1 Top-Level Pin Multiplexing and OHCI Registers

The RM4x microcontroller supports two USB Host ports and one USB Device port. The USB device port interface signals are multiplexed with the second USB Host port interface signals. All USB interface signals are also multiplexed with other functions on the RM4x microcontrollers. The configuration of these multiplexors is done by configuring the I/O Multiplexing Module control registers. Refer to the *I/O Multiplexing and Control Module (IOMM)* chapter for more information on the sequence for configuring these control registers.

The OHCI RHDESCRIPTORA register always reports two available USB host ports, regardless of the top-level pin multiplexing settings.

Table 32-1. USB Host / Device Interface Signal Multiplexing and Control

USB Host / Device Interface Signal Name	Control for Selecting USB Host Signal	Control for Selecting USB Device Signal
USB1.OVERCURRENT	PINMMR14 [7:0] = 08h	-
USB1.RCV	PINMMR13 [31:24] = 08h	-
USB1.VM	PINMMR12 [31:24] = 08h	-
USB1.VP	PINMMR12 [23:16] = 08h	-
USB1.PORTPOWER	PINMMR21 [15:8] = 02h	-
USB1.SPEED	PINMMR19 [15:8] = 04h	-
USB1.SUSPEND	PINMMR20 [23:16] = 08h	-
USB1.TXDAT	PINMMR18 [31:24] = 02h	-
USB1.TXEN	PINMMR17 [7:0] = 04h	-
USB1.TXSE0	PINMMR18 [15:8] = 02h	-
USB2.OVERCURRENT / USB_FUNC.VBUSI	PINMMR1 [15:8] = 08h	PINMMR1 [15:8] = 10h
USB2.RCV / USB_FUNC.RXDI	PINMMR0 [7:0] = 02h	PINMMR0 [7:0] = 04h
USB2.VM / USB_FUNC.RXDMI	PINMMR1 [7:0] = 02h	PINMMR1 [7:0] = 04h
USB2.VP / USB_FUNC.RXDPI	PINMMR0 [15:8] = 02h	PINMMR0 [15:8] = 04h
USB2.PORTPOWER / USB_FUNC.GZO	PINMMR6 [7:0] = 02h	PINMMR6 [7:0] = 04h
USB2.SPEED / USB_FUNC.PUENON	PINMMR4 [31:24] = 04h	PINMMR4 [31:24] = 08h
USB2.SUSPEND / USB_FUNC.SUSPENDO	PINMMR6 [23:16] = 04h	PINMMR6 [23:16] = 08h
USB2.TXDAT / USB_FUNC.TXDO	PINMMR2 [7:0] = 02h	PINMMR2 [7:0] = 04h
USB2.TXEN / USB_FUNC.PUENO	PINMMR4 [23:16] = 04h	PINMMR4 [23:16] = 08h
USB2.TXSE0 / USB_FUNC.SE0O	PINMMR3 [15:8] = 02h	PINMMR3 [15:8] = 04h

32.2.2.2 No Ownership Change Interrupt

The USB host controller does not implement the OHCI ownership change interrupt.

32.2.3 Implementation of OHCI Specification for USB

32.2.3.1 Isochronous Transmit Descriptor (TD) OFFSETX/PSWX Values

The USB host controller implements the *OHCI Specification for USB* optional feature of checking isochronous OFFSETX/PSWX values. If either OFFSETX or OFFSET(X+1) does not have a condition code of *Not Accessed*, or if the OFFSET(X+1) value is not greater than or equal to OFFSETX, then an unrecoverable error is reported. Unrecoverable errors issued for these reasons do not cause an update of the HOSTUEADDR, HOSTUESTATUS, or HOSTTIMEOUTCTRL registers.

32.2.3.2 USB Host Controller Endpoint Descriptor (ED) List Head Pointers

The OHCI *Specification for USB* provides a specific sequence of operations for the host controller driver to perform when setting up the host controller. Failure to follow that sequence can result in malfunction. As a specific example, the HCCONTROLHEADED and HCBULKHEADED pointer registers and the 32 HCCAINTERRUPTTABLE pointers must all point to valid physical addresses of valid endpoint descriptors.

The USB host controller does not check HCCONTROLHEADED registers, HCBULKHEADED registers, or the values in the 32 HCCAINTERRUPTTABLE pointers before using them to access EDs. If any of these pointers are NULL when the corresponding list enable bit is set, the USB host controller attempts to access using the physical address of 0, which causes an unrecoverable error to be signaled. HOSTUEADDR, HOSTUESTATUS, and HOSTTIMEOUTCTRL registers are updated in this case.

32.2.3.3 OHCI USB Suspend State

The USB host controller ignores upstream traffic from downstream devices for about 3 ms after the host controller state (HCCONTROL.HCFS) changes from USB resume state to USB operational state. If any TDs cause generation of downstream packets during that time, the downstream packets are sent, but any response provided by the downstream device is ignored. Any such TDs are aborted with completion codes marked as *Device Not Responding*. TDs on any of the lists (periodic, control, bulk, and isochronous) can cause such an occurrence.

The USB specification requires that system software must provide a 10-ms resume recovery time (T_{RSMRCY}) after a bus segment transitions from resume signaling to normal operational mode. During that time, only start of frame packets are to be sent on the bus segment. It is recommended that system software disable all list enable bits (HCCONTROL.PLE, HCCONTROL.IE, HCCONTROL.CLE, and HCCONTROL.BLE) and then wait for at least 1 ms before setting the host controller into USB suspend state (via HCCONTROL.HCFS). When restoring from suspend, system software must set the host controller into USB resume state, and wait for the host controller to transition into USB operational state. System software must then wait 10 ms before enabling the host controller list enable bits.

When the host controller has been placed into the USB suspend state under software control, but is brought out by a remote wake-up, system software must monitor the HCRHPORTSTATUS[x].PSS and HCRHPORTSTATUS[x].PSSC bits. The HCRHPORTSTATUS[x].PSS bit changes to 0 only after completion of resume signaling on the bus segment completes and completion of the 3-ms period where packets from downstream devices are ignored.

When using port-specific suspend, it is not necessary to disable the host controller lists so long as there are no active EDs and TDs directed toward devices that are downstream of the suspended port. For port-specific suspend operations, the host controller does not issue a root hub status change interrupt with the HCRHPORTSTATUS[n].PSSC bit = 1 and HCRHPORTSTATUS[n].PSS = 0 until after the approximately 3-ms delay after resume signaling completes.

When using port-specific suspend, system software must ensure that there are no active EDs for devices that are downstream of the suspended port before setting the port into suspend mode. While the port is in suspend or being resumed, system software must not enable any EDs for any devices downstream of the suspended port. Once the root hub status change interrupt occurs as a result of the suspended port PSS bit changing to 0, EDs can be enabled for devices downstream of the port that is now operational.

32.2.4 USB Host Controller Registers

Most of the host controller (HC) registers are the OHCI operational registers, which are defined by the *OHCI Specification for USB*. Four additional registers not specified by the *OHCI Specification for USB* provide additional information about the USB host controller state. USB host controller registers can be accessed in user and supervisor modes.

The USB host controller registers are listed in Table 32-2. Section 32.2.4.1 through Section 32.2.4.27 describe specific register bits. The base address for the USB Host controller registers is FCF7 8B00h.

Table 32-2. USB Host Controller Registers

Address	Register[1]	Description	Section
FCF7 8B00h	HCREVISION	OHCI revision number	Section 32.2.4.1
FCF7 8B04h	HCCONTROL	HC operating mode	Section 32.2.4.2
FCF7 8B08h	HCCOMMANDSTATUS	HC command and status	Section 32.2.4.3
FCF7 8B0Ch	HCINTERRUPTSTATUS	HC interrupt status	Section 32.2.4.4
FCF7 8B10h	HCINTERRUPTENABLE	HC interrupt enable	Section 32.2.4.5
FCF7 8B14h	HCINTERRUPTDISABLE	HC interrupt disable	Section 32.2.4.6
FCF7 8B18h	HCHCCA	Physical address of HCCA[2]	Section 32.2.4.7
FCF7 8B1Ch	HCPERIODCURRENTED	Physical address of current periodic endpoint descriptor[2]	Section 32.2.4.8
FCF7 8B20h	HCCONTROLHEADED	Physical address of head of control endpoint descriptor list[2]	Section 32.2.4.9
FCF7 8B24h	HCCONTROLCURRENTED	Physical address of current control endpoint descriptor[2]	Section 32.2.4.10
FCF7 8B28h	HCBULKHEADED	Physical address of head of bulk endpoint descriptor list[2]	Section 32.2.4.11
FCF7 8B2Ch	HCBULKCURRENTED	Physical of current bulk endpoint descriptor[2]	Section 32.2.4.12
FCF7 8B30h	HCDONEHEAD	Physical address of head of list of retired transfer descriptors[2]	Section 32.2.4.13
FCF7 8B34h	HCFMINTERVAL	HC frame interval	Section 32.2.4.14
FCF7 8B38h	HCFMREMAINING	HC frame remaining	Section 32.2.4.15
FCF7 8B3Ch	HCFMNUMBER	HC frame number	Section 32.2.4.16
FCF7 8B40h	HCPERIODICSTART	HC periodic start	Section 32.2.4.17
FCF7 8B44h	HCLSTHRESHOLD	HC low-speed threshold	Section 32.2.4.18
FCF7 8B48h	HCRHDESCRIPTORA	HC root hub A	Section 32.2.4.19
FCF7 8B4Ch	HCRHDESCRIPTORB	HC root hub B	Section 32.2.4.20
FCF7 8B50h	HCRHSTATUS	HC root hub status	Section 32.2.4.21
FCF7 8B54h	HCRHPORTSTATUS0	HC port 0 control and status	Section 32.2.4.22
FCF7 8B58h	HCRHPORTSTATUS1	HC port 1 control and status	Section 32.2.4.23
FCF7 8BE0h	HOSTUEADDR	Host UE address	Section 32.2.4.24
FCF7 8BE4h	HOSTUESTATUS	Host UE status	Section 32.2.4.25
FCF7 8BE8h	HOSTTIMEOUTCTRL	Host time-out control	Section 32.2.4.26
FCF7 8BECh	HOSTREVISION	Host revision	Section 32.2.4.27

[1] Access to these registers must be by 32-bit reads or 32-bit writes. Use of other access sizes may result in undefined operation.
[2] Restrictions apply to the physical addresses used in these registers. See Section 32.2.9.

32.2.4.1 OHCI Revision Number Register (HCREVISION)

The OHCI revision number (Figure 32-1) register reports the revision number of the *OHCI Specification for USB* upon which the USB host controller is based.

Figure 32-1. OHCI Revision Number Register (HCREVISION) [address = FCF78B00h]

31 ... 16
Reserved
R-0

15 ... 8	7 ... 0
Reserved	REV
R-0	R-10h

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

Table 32-3. OHCI Revision Number Register (HCREVISION) Bit Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	REV	10h	OHCI specification revision. The OHCI revision number upon which the USB host controller is based. Write has no effect.

32.2.4.2 HC Operating Mode Register (HCCONTROL)

The HC operating mode register (Figure 32-2) controls the operating mode of the USB host controller.

Figure 32-2. HC Operating Mode Register (HCCONTROL) [address = FCF78B04h]

31 ... 16
Reserved
R-0

15 ... 11	10	9	8
Reserved	RWE	RWC	IR
R-0	R/W-0	R/W-0	R/W-0

7 ... 6	5	4	3	2	1 ... 0
HCFS	BLE	CLE	IE	PLE	CBSR
R/W-00	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 32-4. HC Operating Mode Register (HCCONTROL) Bit Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	0	Reserved
10	RWE	0	Remote wake-up enable. This bit has no effect in the device. The USB host controller does not provide a processor wake-up mechanism.
9	RWC	0	Remote wake-up connected. This bit has no effect in the device. The USB host controller does not provide a processor wake-up mechanism.
8	IR	0	Interrupt routing. The USB host controller does not provide an SMI interrupt. This bit must be 0 to allow the USB host controller interrupt to propagate to the Vectored Interrupt Manager (VIM).

Table 32-4. HC Operating Mode Register (HCCONTROL) Bit Field Descriptions (continued)

Bit	Field	Value	Description
7-6	HCFS		Host controller functional state:
		0	USB reset
		1h	USB resume
		2h	USB operational
		3h	USB suspend
			A transition to USB operational causes SOF generation to begin in 1 ms. The USB host controller can automatically transition from USB suspend to USB resume if a downstream resume is received. The USB host controller enters USB suspend after a software reset. The USB host controller enters USB reset after a hardware reset. The USB reset state resets the root hub and causes downstream signaling of USB reset.
5	BLE		Bulk list enable:
		0	Bulk ED list not processed in the next 1-ms frame. Host controller driver can modify the list. If driver removes the ED pointed to by the HCBULKCURRENTED from the ED list, it must update HCBULKCURRENTED to point to an ED still on the list before it re-enables the bulk list.
		1	Enables processing of bulk ED list. HCBULKHEADED must be 0 or point to a valid ED before setting this bit. HCBULKCURRENTED must point to a valid ED or be 0 before setting this bit.
4	CLE		Control list enable:
		0	Control ED list is not processed in the next 1-ms frame. Host controller driver can modify the control ED list. If driver removes the ED pointed to by the HCCONTROLCURRENTED from the ED list, it must update HCCONTROLCURRENTED to point to an ED still on the list before it re-enables the control list.
		1	Enables processing of the control ED list. HCCONTROLHEADED must be 0 or point to a valid ED before setting this bit. HCCONTROLCURRENTED must be 0 or point to a valid ED before setting this bit.
3	IE		Isochronous enable
		0	Isochronous EDs are not processed. The USB host controller checks this bit every time it finds an isochronous ED in the periodic list.
		1	Enables processing of isochronous EDs. When this bit is written to 1, processing of isochronous EDs can be enabled in the next frame, if not in the current frame.
2	PLE		Periodic list enable
		0	The periodic ED lists are not processed. When written to 0, periodic list processing is disabled beginning with the next frame.
		1	Enables processing of the periodic ED lists. When written to 1, periodic list processing begins in the next frame.
1-0	CBSR		Control/bulk service ratio. Specifies the ratio between control and bulk EDs processed in a frame.
		0	One control ED per bulk ED.
		1h	Two control EDs per bulk ED.
		2h	Three control EDs per bulk ED.
		3h	Four control EDs per bulk ED.

32.2.4.3 HC Command and Status Register (HCCOMMANDSTATUS)

The HC command and status register shows the current state of the host controller and accepts commands from the host controller driver.

Figure 32-3. HC Command and Status Register (HCCOMMANDSTATUS) [address = FCF78B08h]

31	18	17	16
Reserved		SOC	
R-0		R-0	

15	4	3	2	1	0
Reserved		OCR	BLF	CLF	HCR
R-0		R/W-0	R/W-0	R/W-0	R/W-0

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

Table 32-5. HC Command and Status Register (HCCOMMANDSTATUS) Bit Field Descriptions

Bit	Field	Value	Description
31-18	Reserved	0	Reserved
17-16	SOC	0-3h	Scheduling overrun count Counts the number of times a scheduling overrun occurs. This count is incremented even if the host controller driver has not acknowledged any previous pending scheduling overrun interrupt.
15-4	Reserved	0	Reserved
3	OCR	0-1	Ownership change request This bit is set by the host controller driver to gain ownership of the host controller. The device does not support SMI interrupts, so no ownership change interrupt occurs.
2	BLF	0-1	Bulk list filled The host controller driver must set this bit if it modifies the bulk list to include new TDs. If HCBULKCURRENTED is 0, the USB host controller does not begin processing bulk list EDs unless this bit is set. When the USB host controller sees this bit set and begins processing the bulk list, it clears this bit.
1	CLF	0-1	Control list filled The host controller driver must set this bit if it modifies the control list to include new TDs. If HCCONTROLHEADED is 0, the USB host controller does not begin processing control list EDs unless this bit is set. When the USB host controller sees this bit set and begins processing the control list, it clears this bit.
0	HCR	0-1	Host controller reset. Write of 0 has no effect. Value of 1 initiates a software reset of the USB host controller. This transitions the USB host controller to the USB suspend state. This resets most USB host controller OHCI registers. OHCI register accesses must not be attempted until a read of this bit returns a 0. A write of 1 to this bit does not reset the root hub and does not signal USB reset to downstream USB functions.

32.2.4.4 HC Interrupt Status Register (HCINTERRUPTSTATUS)

The HC interrupt status register reports the status of the USB host controller internal interrupt sources.

Figure 32-4. HC Interrupt Status Register (HCINTERRUPTSTATUS) [address = FCF78B0Ch]

31	30	29	16
Reserved	OC	Reserved	
R-0	R-0	R-0	

15	7	6	5	4	3	2	1	0
Reserved		RHSC	FNO	UE	RD	SF	WDH	SO
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 at reset

Table 32-6. HC Interrupt Status Register (HCINTERRUPTSTATUS) Bit Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	Reserved
30	OC	0-1	Ownership change The USB host controller does not implement ownership change interrupts.
29-7	Reserved	0	Reserved
6	RHSC		Root hub status change When 1, indicates a root hub status change has occurred.
		0	Write of 0 has no effect.
		1	Write of 1 clears this bit.
5	FNO		Frame number overflow When 1, indicates a frame number overflow has occurred.
		0	Write of 0 has no effect.
		1	Write of 1 clears this bit.
4	UE		Unrecoverable error When 1, indicates that an unrecoverable error has occurred on the OCPI bus or that an isochronous TD PSW field condition code was not set to *Not Accessed* when the USB host controller attempted to perform a transfer using that PSW/offset pair.
		0	Write of 0 has no effect.
		1	Write of 1 clears this bit.
3	RD		Resume detected When 1, indicates that a downstream device has issued a resume request.
		0	Write of 0 has no effect.
		1	Write of 1 clears this bit.
2	SF		Start of frame When 1, indicates that a SOF has been issued.
		0	Write of 0 has no effect.
		1	Write of 1 clears this bit.
1	WDH		Write done head When 1, indicates that the USB host controller has updated the HCDONEHEAD register.
		0	Write of 0 has no effect.
		1	Write of 1 clears this bit. The host controller driver must read the value from HCDONEHEAD before writing 1 to this bit.
0	SO		Scheduling overrun When 1, indicates that a scheduling overrun has occurred.
		0	Write of 0 has no effect.
		1	Write of 1 clears this bit.

32.2.4.5 HC Interrupt Enable Register (HCINTERRUPTENABLE)

The HC interrupt enable register (Figure 32-5) enables various OHCI interrupt sources to generate interrupts to the device level 2 interrupt handler.

Figure 32-5. HC Interrupt Enable Register (HCINTERRUPTENABLE) [address = FCF78B10h]

31	30	29	16
MIE	OC	Reserved	
R/W-0	R-0	R-0	

15	7	6	5	4	3	2	1	0
Reserved		RHSC	FNO	UE	RD	SF	WDH	SO
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 at reset

Table 32-7. HC Interrupt Enable Register (HCINTERRUPTENABLE) Bit Field Descriptions

Bit	Field	Value	Description
31	MIE		Master interrupt enable
			When 1, allows other enabled OHCI interrupt sources to propagate to the Vectored Interrupt Manager (VIM).
			When 0, OHCI interrupt sources are ignored and no USB host controller interrupts are propagated to the Vectored Interrupt Manager (VIM).
		0	Write of 0 has no effect.
		1	Write of 1 sets this bit.
30	OC	0	Ownership change
			This bit has no effect on the device.
29-7	Reserved	0	Reserved
6	RHSC		Root hub status change
			When 1 and MIE is 1, allows root hub status change interrupts to propagate to the Vectored Interrupt Manager (VIM).
			When 0, or when MIE is 0, root hub status change interrupts do not propagate.
		0	Write of 0 has no effect.
		1	Write of 1 sets this bit.
5	FNO		Frame number overflow
			When 1 and MIE is 1, allows frame number overflow interrupts to propagate to the Vectored Interrupt Manager (VIM).
			When 0, or when MIE is 0, frame number overflow interrupts do not propagate.
		0	Write of 0 has no effect.
		1	Write of 1 sets this bit.
4	UE		Unrecoverable error
			When 1 and MIE is 1, allows unrecoverable error interrupts to propagate to the Vectored Interrupt Manager (VIM).
			When 0, or when MIE is 0, unrecoverable error interrupts do not propagate.
		0	Write of 0 has no effect.
		1	Write of 1 sets this bit.
3	RD		Resume detected
			When 1 and MIE is 1, allows resume detected interrupts to propagate to the Vectored Interrupt Manager (VIM).
			When 0, or when MIE is 0, resume detected interrupts do not propagate.
		0	Write of 0 has no effect.
		1	Write of 1 sets this bit.

Table 32-7. HC Interrupt Enable Register (HCINTERRUPTENABLE) Bit Field Descriptions (continued)

Bit	Field	Value	Description
2	SF		Start of frame
			When 1 and MIE is 1, allows start of frame interrupts to propagate to the Vectored Interrupt Manager (VIM).
			When 0, or when MIE is 0, start of frame interrupts do not propagate.
		0	Write of 0 has no effect.
		1	Write of 1 sets this bit.
1	WDH		Write done head
			When 1 and MIE is 1, allows write done head interrupts to propagate to the Vectored Interrupt Manager (VIM).
			When 0, or when MIE is 0, write done head interrupts do not propagate.
		0	Write of 0 has no effect.
		1	Write of 1 sets this bit.
0	SO		Scheduling overrun
			When 1 and MIE is 1, allows scheduling overrun interrupts to propagate to the Vectored Interrupt Manager (VIM).
			When 0, or when MIE is 0, scheduling overrun interrupts do not propagate.
		0	Write of 0 has no effect.
		1	Write of 1 sets this bit.

32.2.4.6 HC Interrupt Disable Register (HCINTERRUPTDISABLE)

The HC interrupt disable register is used to clear bits in the HCINTERRUPTENABLE register.

Figure 32-6. HC Interrupt Disable Register (HCINTERRUPTDISABLE) [address = FCF78B14h]

31	30	29	16
MIE	OC	Reserved	
R/W-0	R-0	R-0	

15	7	6	5	4	3	2	1	0
Reserved		RHSC	FNO	UE	RD	SF	WDH	SO
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 at reset

Table 32-8. HC Interrupt Disable Register (HCINTERRUPTDISABLE) Bit Field Descriptions

Bit	Field	Value	Description
31	MIE		Master interrupt enable
			Read always returns 0.
		0	Write of 0 has no effect.
		1	Write of 1 clears the HCINTERRUPTENABLE MIE bit.
30	OC	0	Ownership change
			This bit has no effect on the device.
29-7	Reserved	0	Reserved
6	RHSC		Root hub status change
			Read always returns 0.
		0	Write of 0 has no effect.
		1	Write of 1 clears the HCINTERRUPTENABLE RHSC bit.

Table 32-8. HC Interrupt Disable Register (HCINTERRUPTDISABLE) Bit Field Descriptions (continued)

Bit	Field	Value	Description
5	FNO		Frame number overflow
			Read always returns 0.
		0	Write of 0 has no effect.
		1	Write of 1 clears the HCINTERRUPTENABLE FNO bit.
4	UE		Unrecoverable error
			Read always returns 0.
		0	Write of 0 has no effect.
		1	Write of 1 clears the HCINTERRUPTENABLE UE bit.
3	RD		Resume detected
			Read always returns 0.
		0	Write of 0 has no effect.
		1	Write of 1 clears the HCINTERRUPTENABLE RD bit.
2	SF		Start of frame
			Read always returns 0.
		0	Write of 0 has no effect.
		1	Write of 1 clears the HCINTERRUPTENABLE SF bit.
1	WDH		Write done head
			Read always returns 0.
		0	Write of 0 has no effect.
		1	Write of 1 clears the HCINTERRUPTENABLE WDH bit.
0	SO		Scheduling overrun
			Read always returns 0.
		0	Write of 0 has no effect.
		1	Write of 1 clears the HCINTERRUPTENABLE SO bit.

32.2.4.7 HC HCCA Address Register (HCHCCA)

The HCCA address register defines the physical address of the beginning of the HCCA.

Figure 32-7. HC HCCA Address Register (HCHCCA) [address = FCF78B18h]

31	16
HCCA	
R/W-0	

15 8	7 0
HCCA	Reserved
R/W-0	R-0

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

Table 32-9. HC HCCA Address Register (HCHCCA) Bit Field Descriptions

Bit	Field	Value	Description
31-8	HCCA	0-FF FFFFh	Physical address of the beginning of the HCCA.
7-0	Reserved	0	Reserved

32.2.4.8 HC Current Periodic Register (HCPERIODCURRENTED)

The HC current periodic register defines the physical address of the next endpoint descriptor (ED) on the periodic ED List.

Figure 32-8. HC Current Periodic Register (HCPERIODCURRENTED) [address = FCF78B1Ch]

31 - 16
PCED
R-0

15 - 4	3 - 0
PCED	Reserved
R-0	R-0

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

Table 32-10. HC Current Periodic Register (HCPERIODCURRENTED) Bit Field Descriptions

Bit	Field	Value	Description
31-4	PCED	0-FFF FFFFh	Physical address of current ED on the periodic ED list. This field represents bits 31:4 of the physical address of the next ED on the periodic ED List. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See Section 32.2.9 for the restrictions on physical addresses.
3-0	Reserved	0	Reserved

32.2.4.9 HC Head Control Register (HCCONTROLHEADED)

The HC head control register defines the physical address of the head ED of the control ED list.

Figure 32-9. HC Head Control Register (HCCONTROLHEADED) [address = FCF78B20h]

31 - 16
CHED
R/W-0

15 - 4	3 - 0
CHED	Reserved
R/W-0	R-0

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

Table 32-11. HC Head Control Register (HCCONTROLHEADED) Bit Field Descriptions

Bit	Field	Value	Description
31-4	CHED	0-FFF FFFFh	Physical address of current ED on the periodic ED list. This field represents bits 31:4 of the physical address of the next ED on the periodic ED List. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See Section 32.2.9 for the restrictions on physical addresses.
3-0	Reserved	0	Reserved

32.2.4.10 HC Current Control Register (HCCONTROLCURRENTED)

The HC current control register defines the physical address of the next ED on the control ED list.

Figure 32-10. HC Current Control Register (HCCONTROLCURRENTED) [address = FCF78B24h]

31	16
CCED	
R/W-0	

15	4	3	0
CCED		Reserved	
R/W-0		R-0	

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

Table 32-12. HC Current Control Register (HCCONTROLCURRENTED) Bit Field Descriptions

Bit	Field	Value	Description
31-4	CCED	0-FFF FFFFh	Physical address of current ED on the periodic ED list. This field represents bits 31:4 of the physical address of the next ED on the periodic ED List. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See Section 32.2.9 for the restrictions on physical addresses. A value of 0x0000000 indicates that the USB host controller has reached the end of the control ED list without finding any transfers to process. This register is automatically updated by the USB host controller.
3-0	Reserved	0	Reserved

32.2.4.11 HC Head Bulk Register (HCBULKHEADED)

The head bulk register defines the physical address of the head ED on the bulk ED list.

Figure 32-11. HC Head Bulk Register (HCBULKHEADED) [address = FCF78B28h]

31	16
BHED	
R/W-0	

15	4	3	0
BHED		Reserved	
R/W-0		R-0	

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

Table 32-13. HC Head Bulk Register (HCBULKHEADED) Bit Field Descriptions

Bit	Field	Value	Description
31-4	BHED	0-FFF FFFFh	Physical address of current ED on the periodic ED list. This field represents bits 31:4 of the physical address of the next ED on the periodic ED List. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See Section 32.2.9 for the restrictions on physical addresses. This register is automatically updated by the USB host controller.
3-0	Reserved	0	Reserved

32.2.4.12 HC Current Bulk Register (HCBULKCURRENTED)

The current bulk register defines the physical address of the next ED on the bulk ED list.

Figure 32-12. HC Current Bulk Register (HCBULKCURRENTED) [address = FCF78B2Ch]

31	16
BCED	
R/W-0	

15	4	3	0
BCED		Reserved	
R/W-0		R-0	

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

Table 32-14. HC Current Bulk Register (HCBULKCURRENTED) Bit Field Descriptions

Bit	Field	Value	Description
31-4	BCED	0-FFF FFFFh	Physical address of current ED on the periodic ED list. This field represents bits 31:4 of the physical address of the next ED on the periodic ED List. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See Section 32.2.9 for the restrictions on physical addresses. A value of 0x0000000 indicates that the USB host controller has reached the end of the bulk ED list without finding any transfers to process. This register is automatically updated by the USB host controller.
3-0	Reserved	0	Reserved

32.2.4.13 HC Head Done Register (HCDONEHEAD)

The head done register defines the physical address of the current head of the done TD queue.

Figure 32-13. HC Head Done Register (HCDONEHEAD) [address = FCF78B30h]

31	16
DH	
R/W-0	

15	4	3	0
DH		Reserved	
R/W-0		R-0	

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

Table 32-15. HC Head Done Register (HCDONEHEAD) Bit Field Descriptions

Bit	Field	Value	Description
31-4	DH	0-FFF FFFFh	Physical address of current ED on the periodic ED list. This field represents bits 31:4 of the physical address of the next ED on the periodic ED List. EDs are assumed to begin at 16-byte-aligned address, so bits 3:0 of this pointer are assumed to be 0. See Section 32.2.9 for the restrictions on physical addresses. This register is automatically updated by the USB host controller.
3-0	Reserved	0	Reserved

32.2.4.14 HC Frame Interval Register (HCFMINTERVAL)

The frame interval register defines the number of 12-MHz clock pulses in each USB frame.

Figure 32-14. HC Frame Interval Register (HCFMINTERVAL) [address = FCF78B34h]

31	30 16
FIT	FSMPS
R/W-0	R/W-0

15 14	13 0
Reserved	FI
R-0	R/W-2EDFh

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

Table 32-16. HC Frame Interval Register (HCFMINTERVAL) Bit Field Descriptions

Bit	Field	Value	Description
31	FIT	0-1	Frame interval toggle The host controller driver must toggle this bit any time it changes the frame interval field.
30-16	FSMPS	0-7FFFh	Largest data packet Largest data packet size allowed for full-speed packets, in bit times.
15-14	Reserved	0	Reserved
13-0	FI	0-3FFFh	Frame interval Number of 12-MHz clocks in the USB frame. Nominally, this is set to 11,999 to give a 1-ms frame. The host controller driver can make minor changes to this field to attempt to manually synchronize with another clock source.

32.2.4.15 HC Frame Remaining Register (HCFMREMAINING)

The HC frame remaining register reports the number of full-speed bit times remaining in the current frame.

Figure 32-15. HC Frame Remaining Register (HCFMREMAINING) [address = FCF78B38h]

31	30 16
FRT	Reserved
R-0	R-0

15 14	13 0
Reserved	FR
R-0	R-0

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

Table 32-17. HC Frame Remaining Register (HCFMREMAINING) Bit Field Descriptions

Bit	Field	Value	Description
31	FRT	0-1	Frame interval toggle The host controller driver must toggle this bit any time it changes the frame interval field.
30-14	Reserved	0	Reserved
13-0	FR	0-3FFFh	Frame remaining The number of full-speed bit times remaining in the current frame. This field is automatically reloaded with the frame interval field value at the beginning of every frame.

32.2.4.16 HC Frame Number Register (HCFMNUMBER)

The HC frame number register reports the current USB frame number.

Figure 32-16. HC Frame Number Register (HCFMNUMBER) [address = FCF78B3Ch]

31	16
Reserved	
R-0	

15	0
FN	
R-0	

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

Table 32-18. HC Frame Number Register (HCFMNUMBER) Bit Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Reserved
13-0	FN	0-3FFFh	Frame number This field reports the current USB frame number. It is incremented when the frame remaining field is reloaded with the frame interval field value. Frame number automatically rolls over from FFFFh to 0. After frame number is incremented, its new value is written to the HCCA and the USB host controller sets the SOF interrupt status bit and begins processing the ED lists.

32.2.4.17 HC Periodic Start Register (HCPERIODICSTART)

The HC periodic start register defines the position within the USB frame where EDs on the periodic list have priority over EDs on the bulk and control lists.

Figure 32-17. HC Periodic Start Register (HCPERIODICSTART) [address = FCF78B40h]

31	16
Reserved	
R-0	

15 14	13 0
Reserved	PS
R-0	R/W-0

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

Table 32-19. HC Periodic Start Register (HCPERIODICSTART) Bit Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Reserved
13-0	PS	0-3FFFh	Periodic start The host controller driver must program this value to be about 10% less than the frame interval field value so that control and bulk EDs have priority for the first 10% of the frame; then periodic EDs have priority for the remaining 90% of the frame.

32.2.4.18 HC Low-Speed Threshold Register (HCLSTHRESHOLD)

The HC low-speed threshold register defines the latest time in a frame that the USB host controller can begin a low-speed packet.

Figure 32-18. HC Low-Speed Threshold Register (HCLSTHRESHOLD) [address = FCF78B44h]

31	16
Reserved	
R-0	

15	14	13	0
Reserved		LST	
R-0		R/W-628h	

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

Table 32-20. HC Low-Speed Threshold Register (HCLSTHRESHOLD) Bit Field Descriptions

Bit	Field	Value	Description
31-14	Reserved	0	Reserved
13-0	LST	0-3FFFh	Low-speed threshold This field defines the number of full-speed bit times in the frame after which the USB host controller cannot start an 8-byte low-speed packet. The USB host controller only begins a low-speed transaction if the frame remaining field is greater than the low-speed threshold. The host controller driver must set this field to a value that ensures that an 8-byte low-speed TD completes before the end of the frame. When set, the host controller driver must not change the value.

32.2.4.19 HC Root Hub A Register (HCRHDESCRIPTORA)

The HC root hub A register defines several aspects of the USB host controller root hub functionality.

Figure 32-19. HC Root Hub A Register (HCRHDESCRIPTORA) [address = FCF78B48h]

31	24	23	16
POTPG		Reserved	
R/W-2h		R-0	

15	13	12	11	10	9	8	7	0
Reserved		NOCP	OCPM	DT	NPS	PSM	NDP	
R/W-0		R/W-1	R/W-0	R-0	R/W-1	R/W-0	R-2h	

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

Table 32-21. HC Root Hub A Register (HCRHDESCRIPTORA) Field Descriptions

Bit	Field	Value	Description
31-24	POTPG	0-FFh	Power-on to power-good time This field defines the minimum amount of time (2 ms × POTPG) between the USB host controller turning on power to a downstream port and when the USB host can access the downstream device. The default value is 0x2, so that the time is 4 ms. This field has no effect on USB host controller operation. After turning on power to a port, the USB host controller driver must delay the amount of time implied by POTPG before attempting to reset an attached downstream device. The required amount of time is system implementation-specific and must be calculated based on the amount of time the VBUS supply takes to provide valid VBUS to a worst-case downstream USB function controller. The implementation-specific value must be computed and then written to this register before the USB host controller driver is initialized.
23-13	Reserved	0	Reserved

Table 32-21. HC Root Hub A Register (HCRHDESCRIPTORA) Field Descriptions (continued)

Bit	Field	Value	Description
12	NOCP		No overcurrent protection
		0	When 0, the USB Host ports support over-current protection inputs. The USB Host ports must not be in the Reset state for the over-current protection inputs to take effect.
		1	When 1, this bit indicates that the USB host controller does not implement overcurrent protection inputs. This bit defaults to 1 and can be cleared by the application software.
11	OCPM		Overcurrent protection mode
		0	Over-current status is reported collectively for all downstream ports. This is the default value.
		1	Over-current status is reported on a per-port basis.
			This field is valid only if the NoOverCurrentProtection bit is cleared.
10	DT		Device type
		0	This bit is always 0, which indicates that the USB host controller implemented is not a compound device.
9	NPS		No power switching
		0	Indicates that VBUS power switching is supported and is either per-port or all-port switched per the power switching mode field.
		1	Indicates that VBUS power switching is not supported and that power is available to all downstream ports when the USB host controller is powered. This is the default value.
			The RM4x microcontroller USB Host ports support over-current protection inputs and automatic power switching upon indication of an over-current condition. The USB Host ports must not be in the Reset state for the over-current protection inputs to take effect.
8	PSM		Power switching mode
		0	Indicates that all ports are powered at the same time. This is the default value.
		1	Indicates that each port is powered individually. This mode allows port power to be controlled by either the global switch or per-port switching.
			If the PortPowerControlMask (PPCM) bit in the HC Root Hub B Register (HCRHDESCRIPTORB) is set, the USB host port only responds to port power commands (Set/ClearPortPower). If the port mask is cleared, then the USB host port is controlled by the global power switch (Set/ClearGlobalPower).
7-0	NDP		Number of downstream ports
		2h	This register defaults to 2 to indicate two downstream ports.
			The USB signal multiplexing mode and top-level pin multiplexing features can place the device in a mode where 0, 1, or 2 of the USB host controller downstream ports are usable. This register reports two ports, regardless of top-level pin multiplexing mode.

32.2.4.20 HC Root Hub B Register (HCRHDESCRIPTORB)

The HC root hub B register defines several aspects of the USB host controller root hub functionality.

Figure 32-20. HC Root Hub B Register (HCRHDESCRIPTORB) [address = FCF78B4Ch]

31	16
PPCM[15-0]	
R/W-0	

15	0
DR[15-0]	
R/W-0	

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

Table 32-22. HC Root Hub B Register (HCRHDESCRIPTORB) Bit Field Descriptions

Bit	Field	Value	Description
31-16	PPCM	0-FFFFh	Port power control mask Each bit defines whether a corresponding downstream port has port power controlled by the global power control. If set, per-port power control is implemented for the corresponding port. If clear, global power control is implemented for the corresponding port. PPCM bit 0 is reserved. PPCM bit 1 is the port power control mask for downstream port 0. PPCM bit 2 is the port power control mask for downstream port 1. PPCM bits 3 through 15 are reserved.
15-0	DR	0-FFFFh	Device removable Each bit defines whether a corresponding downstream port has a removable or non-removable device. A cleared bit indicates the corresponding port may have a removable device attached. A set bit indicates that the corresponding port has a non-removable device attached. DR bit 0 is reserved.. DR bit 1 is the device removable bit for downstream port 0. DR bit 2 is the device removable bit for downstream port 1. DR bits 3 through 15 are reserved.

32.2.4.21 HC Root Hub Status Register (HCRHSTATUS)

The HC root hub status register reports the USB host controller root hub status.

Figure 32-21. HC Root Hub Status Register (HCRHSTATUS) [address = FCF78B50h]

31	30	18	17	16
CRWE	Reserved		OCIC	LPSC
R/W-0	R-0		R/W-0	R/W-0

15	14	2	1	0
DRWE	Reserved		OCI	LPS
R/W-0	R-0		R-0	R/W-0

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

Table 32-23. HC Root Hub Status Register (HCRHSTATUS) Bit Field Descriptions

Bit	Field	Value	Description
31	CRWE	0-1	Clear remote wake-up enable Write of 0 has no effect. Write of 1 clears the device remote wake-up enable bit.
30-18	Reserved	0	Reserved
17	OCIC		Overcurrent indication change This bit is automatically set when the overcurrent indicator bit changes.
		0	Write of 0 has no effect.
		1	Write of 1 clears this bit.
16	LPSC		Local power status change
		0	Write of 0 has no effect.
		1	Write of 1 sets the port power status bits for all ports if power switching mode is 0. A write of 1 sets port power status bits for ports with their corresponding port power control mask bits cleared if power switching mode is 1.
15	DRWE		Device remote wake-up enable When 1, this bit enables a connect status change event to be treated as a resume event, which causes a transition from USB suspend to USB resume state and sets the resume detected interrupt status bit. When 0, connect status change events do not cause a transition from USB suspend to USB resume state and the resume detected interrupt is not changed.
		0	Write of 0 has no effect.
		1	Write of 1 sets the device remote wake-up enable bit.
14-2	Reserved	0	Reserved
1	OCI		Overcurrent indicator. This bit reports global overcurrent indication if global overcurrent reporting is selected.
		0	When 0, no overcurrent condition has been sensed.
		1	When 1, this bit indicates that an overcurrent condition has been sensed.
0	LPS		Local power status
		0	Write of 0 has no effect.
		1	Write of 1 when in global power mode (power switching mode = 0), turns off power to all ports. If in per-port power mode (power switching mode = 1), a write of 1 turns off power to those ports whose corresponding port power control mask bit is 0.

32.2.4.22 HC Port 0 Status and Control Register (HCRHPORTSTATUS0)

The HC port 0 status and control register reports and controls the state of USB host port 0.

Figure 32-22. HC Port 0 Status and Control Register (HCRHPORTSTATUS0) [address = FCF78B54h]

<table>
<tr><td>31</td><td></td><td></td><td></td><td></td><td></td><td>24</td></tr>
<tr><td colspan="7">Reserved</td></tr>
<tr><td colspan="7">R-0</td></tr>
<tr><td>23</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="2">Reserved</td><td>PRSC</td><td>OCIC</td><td>PSSC</td><td>PESC</td><td>CSC</td></tr>
<tr><td colspan="2">R-0</td><td>R/W-0</td><td>R/W-0</td><td>R/W-0</td><td>R/W-0</td><td>R/W-0</td></tr>
<tr><td>15</td><td></td><td></td><td></td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="5">Reserved</td><td>LSDA/CPP</td><td>PPS/SPP</td></tr>
<tr><td colspan="5">R-0</td><td>R/W-0</td><td>R/W-1</td></tr>
<tr><td>7</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">Reserved</td><td>PRS/SPR</td><td>POCI/CSS</td><td>PSS/SPS</td><td>PES/SPE</td><td>CCS/CPE</td></tr>
<tr><td colspan="2">R-0</td><td>R/W-0</td><td>R/W-0</td><td>R/W-0</td><td>R/W-0</td><td>R/W-0</td></tr>
</table>

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

Table 32-24. HC Port 0 Status and Control Register (HCRHPORTSTATUS0) Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	0	Reserved
20	PRSC		Port 0 reset status change
			This bit indicates, when 1, that the port 0 port reset status bit has changed.
		0	Write of 0 has no effect.
		1	Write of 1 clears this bit.
19	OCIC		Port 0 overcurrent indicator change
			This bit indicates, when 1, that the port 0 port overcurrent indicator has changed.
		0	Write of 0 has no effect.
		1	Write of 1 clears this bit.
18	PSSC		Port 0 suspend status change
			This bit indicates, when 1, that the port 0 port suspend status has changed. Suspend status is considered to have changed only after the resume pulse, low-speed EOP, and 3-ms synchronization delays have been completed.
		0	Write of 0 has no effect.
		1	Write of 1 clears this bit.
17	PESC		Port 0 enable status change
			This bit indicates, when 1, that the port 0 port enable status changed.
		0	Write of 0 has no effect.
		1	Write of 1 clears this bit.
16	CSC		Port 0 connect status change
			This bit indicates, when 1, that the port 0 current connect status has changed due to a connect or disconnect event. If current connect status is 0 when a set port reset, set port enable, or set port suspend write occurs, then this bit is set.
		0	Write of 0 has no effect.
		1	Write of 1 clears this bit.
			If the HCRHDESCRIPTORB.DR[1] bit is set to indicate a non-removable USB device on port 0, this bit is set only after a root hub reset to inform the system that the device is attached.
15-10	Reserved	0	Reserved

Table 32-24. HC Port 0 Status and Control Register (HCRHPORTSTATUS0) Field Descriptions (continued)

Bit	Field	Value	Description
9	LSDA/CPP		Port 0 low-speed device attached/clear port power This bit indicates, when read as 1, that a low-speed device is attached to port 0. A 0 in this bit indicates a full-speed device. This bit is valid only when port 0 current connect status is 1.
		0	Write of 0 has no effect.
		1	The host controller driver can write a 1 to this bit to clear the port 0 port power status.
8	PPS/SPP		Port 0 port power status/set port power This bit indicates, when read as 1, that the port 0 power is enabled. When read as 0, port 0 power is not enabled.
		0	Write of 0 has no effect.
		1	Write of 1 to this bit sets the port 0 port power status bit.
7-5	Reserved	0	Reserved
4	PRS/SPR		Port 0 port reset status/set port reset When read as 1, indicates that port 0 is signaling the USB reset. When read as 0, USB reset is not being sent to port 0.
		0	Write of 0 has no effect.
		1	Write of 1 to this bit sets the port 0 port reset status bit and causes the USB host controller to begin signaling USB reset to port 0.
3	POCI/CSS		Port 0 port overcurrent indicator/clear suspend status When read as 1, indicates a port 0 port overcurrent condition has occurred. When 0, no port 0 port overcurrent condition has occurred.
		0	Write of 0 has no effect.
		1	Write of 1 to this bit when port 0 port suspend status is 1 causes resume signaling on port 0. A write of 1 when port 0 port suspend status is 0 has no effect.
2	PSS/SPS		Port 0 port suspend status/set port suspend When read as 0, indicates that port 0 is not in the USB suspend state. This bit is cleared automatically at the end of the USB resume sequence and also at the end of the USB reset sequence. When read as 1, indicates that port 0 is in the USB suspend state or is in the resume sequence.
		0	Write of 0 has no effect.
		1	If port 0 current connect status is 1, a write of 1 to this bit sets the port 0 port suspend status bit and places port 0 in USB suspend state. If current connect status is 0, a write of 1 instead sets connect status change to inform the USB host controller driver software of an attempt to suspend a disconnected device.
1	PES/SPE		Port 0 port enable status/set port enable When read as 0, this bit indicates that port 0 is not enabled. When read as 1, indicates that port 0 is enabled. This bit is automatically set at completion of port 0 USB reset if it was not already set before the USB reset completed, and is automatically set at the end of a USB suspend if the port was not enabled when the USB resume completed.
		0	Write of 0 has no effect.
		1	Write of 1 to this bit when port 0 current connect status is 1 sets the port 0 port enable status bit. A write of 1 when port 0 current connect status is 0 has no effect.
0	CCS/CPE		Port 0 current connection status/clear port enable When read as 1, indicates that port 0 currently has a USB device attached. When 0, indicates that no USB device is attached to port 0. This bit is set to 1 after root hub reset if the HCRHDESCRIPTORB.DR[1] bit is set to indicate a non-removable device on port 0.
		0	Write of 0 has no effect.
		1	Write of 1 to this bit clears the port 0 port enable bit.

32.2.4.23 HC Port 1 Status and Control Register (HCRHPORTSTATUS1)

The HC port 1 status register reports and controls the state of USB host port 1.

Figure 32-23. HC Port 1 Status and Control Register (HCRHPORTSTATUS1) [address = FCF78B58h]

31-24
Reserved
R-0

23-21	20	19	18	17	16
Reserved	PRSC	OCIC	PSSC	PESC	CSC
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15-10	9	8
Reserved	LSDA/CPP	PPS/SPP
R-0	R/W-0	R/W-1

7-5	4	3	2	1	0
Reserved	PRS/SPR	POCI/CSS	PSS/SPS	PES/SPE	CCS/CPE
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 32-25. HC Port 1 Status and Control Register (HCRHPORTSTATUS1) Field Descriptions

Bit	Field	Value	Description
31-21	Reserved	0	Reserved
20	PRSC		Port 1 reset status change
			This bit indicates, when 1, that the port 1 port reset status bit has changed.
		0	Write of 0 has no effect.
		1	Write of 1 clears this bit.
19	OCIC		Port 1 overcurrent indicator change
			This bit indicates, when 1, that the port 1 port overcurrent indicator has changed.
		0	Write of 0 has no effect.
		1	Write of 1 clears this bit.
18	PSSC		Port 1 suspend status changed
			This bit indicates, when 1, that the port 1 port suspend status has changed. Suspend status is considered to have changed only after the resume pulse, low-speed EOP, and 3-ms synchronization delays have been completed.
		0	Write of 0 has no effect.
		1	Write of 1 clears this bit.
17	PESC		Port 1 enable status change
			This bit indicates, when 1, that the port 1 port enable status changed.
		0	Write of 0 has no effect.
		1	Write of 1 clears this bit.
16	CSC		Port 1 connect status change
			This bit indicates, when 1, that the port 1 current connect status has changed due to a connect or disconnect event. If current connect status is 0 when a set port reset, set port enable, or set port suspend write occurs, then this bit is set.
		0	Write of 0 has no effect.
		1	Write of 1 clears this bit.
			If the HCRHDESCRIPTORB.DR[2] bit is set to indicate a nonremovable USB device on port 1, this bit is set only after a root hub reset to inform the system that the device is attached.
15-10	Reserved	0	Reserved

Table 32-25. HC Port 1 Status and Control Register (HCRHPORTSTATUS1) Field Descriptions (continued)

Bit	Field	Value	Description
9	LSDA/CPP		Port 1 low-speed device attached/clear port power
			This bit indicates, when read as 1, that a low-speed device is attached to port 1. A 0 in this bit indicates a full-speed device.
			This bit is valid only when port 1 current connect status is 1.
		0	Write of 0 has no effect.
		1	The host controller driver can write a 1 to this bit to clear the port 1 port power status.
8	PPS/SPP		Port 1 port power status/set port power
			This bit indicates, when read as 1, that the port 1 power is enabled. When read as 0, port 1 power is not enabled.
		0	Write of 0 has no effect.
		1	Write of 1 to this bit sets the port 1 port power status bit.
7-5	Reserved	0	Reserved
4	PRS/SPR		Port 1 port reset status/set port reset
			When read as 1, indicates that port 1 is sending a USB reset. When read as 0, USB reset is not being sent to port 1.
		0	Write of 0 has no effect.
		1	Write of 1 to this bit sets the port 1 port reset status bit and causes the USB host controller to begin signaling USB reset to port 1.
3	POCI/CSS		Port 1 port overcurrent indicator/clear suspend status
			When read as 1, indicates that a port 1 port overcurrent condition has occurred. When 0, no port 1 port overcurrent condition has occurred.
		0	Write of 0 has no effect.
		1	Write of 1 to this bit when port 1 port suspend status is 1 causes resume signaling on port 1. A write of 1 when port 1 port suspend status is 0 has no effect.
2	PSS/SPS		Port 1 port suspend status/set port suspend
			When read as 0, indicates that port 1 is not in the USB suspend state. This bit is cleared automatically at the end of the USB resume sequence and also at the end of the USB reset sequence.
			When read as 1, indicates that port 1 is in the USB suspend state, or is in the resume sequence.
		0	Write of 0 has no effect.
		1	If port 1 current connect status is 1, a write of 1 to this bit sets the port 1 port suspend status bit and places port 1 in USB suspend state. If current connect status is 0, a write of 1 instead sets connect status change to inform the USB host controller driver software of an attempt to suspend a disconnected device.
1	PES/SPE		Port 1 port enable status/set port enable
			When read as 0, this bit indicates that port 1 is not enabled.
			When read as 1, indicates that port 1 is enabled. This bit is automatically set at completion of port 1 USB reset if it was not already set before the USB reset completed and is automatically set at the end of a USB suspend if the port was not enabled when the USB resume completed.
		0	Write of 0 has no effect.
		1	Write of 1 to this bit when port 1 current connect status is 1 sets the port 1 port enable status bit. A write of 1 when port 1 current connect status is 0 has no effect.
0	CCS/CPE		Port 1 current connection status/clear port enable
			When read as 1, indicates that port 1 currently has a USB device attached. When 0, indicates that no USB device is attached to port 1.
			This bit is set to 1 after root hub reset if the HCRHDESCRIPTORB.DR[2] bit is set to indicate a non-removable device on port 1.
		0	Write of 0 has no effect.
		1	Write of 1 to this bit clears the port 1 port enable bit.

32.2.4.24 Host UE Address Register (HOSTUEADDR)

The host UE address register reports the physical address of the last OCPI bus access that caused an unrecoverable error (UE). This register has no meaning until an unrecoverable error has occurred. It also has no meaning if the USB host controller issues an unrecoverable error because the offset checking fault occurred while processing an isochronous TD. This register is not defined by the OHCI specification.

Figure 32-24. Host UE Address Register (HOSTUEADDR) [address = FCF78BE0h]

31–16
UE_ADDR
R-0

15–0
UE_ADDR
R-0

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

Table 32-26. Host UE Address Register (HOSTUEADDR) Field Descriptions

Bit	Field	Value	Description
31-0	UE_ADDR	0-FFFF FFFFh	Unrecoverable error address This register captures the physical address of any bus transaction that is started by the USB host controller that encounters an unrecoverable error condition. This information, along with the information in HOSTUESTATUS, can help a developer determine why the USB host issued an access to a physical address that resulted in an unrecoverable error.

32.2.4.25 Host UE Status Register (HOSTUESTATUS)

The host UE status register reports the bus cycle-type for the last unrecoverable error that occurred. This register has no meaning until an unrecoverable error has occurred. It also has no meaning if the USB host controller issues an unrecoverable error because the offset checking fault occurred while processing an isochronous TD. This register is not defined by the OHCI specification.

Figure 32-25. Host UE Status Register (HOSTUESTATUS) [address = FCF78BE4h]

31–16
Reserved
R-x

15–1	0
Reserved	UEACCESS
R-x	R-0

LEGEND: R = Read only; -*n* = value at reset; -x = value is indeterminate after reset

Table 32-27. Host UE Status Register (HOSTUESTATUS) Bit Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	x	Reserved
0	UEACCESS	0-1	Access type when unrecoverable error occurred When an unrecoverable error occurs because of time-out of a bus write, this bit is set. When an unrecoverable error occurs because of time-out of a bus read, this bit is cleared. This bit has no meaning before an unrecoverable error occurs. This information, along with the information in HOSTUEADDR, can help a developer determine why the USB host issued a bus access that resulted in an unrecoverable error.

32.2.4.26 Host Time-out Control Register (HOSTTIMEOUTCTRL)

The host time-out control register controls the USB host controller bus time-out mechanism. This register is not defined by the OHCI specification.

Figure 32-26. Host Time-out Control Register (HOSTTIMEOUTCTRL) [address = FCF78BE8h]

31-16
Reserved
R-0

15-1	0
Reserved	TO_DIS
R-0	R/W-0

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

Table 32-28. Host Time-out Control Register (HOSTTIMEOUTCTRL) Bit Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	TO_DIS		Bus time-out disable
		0	When 0 (the default state), the USB host controller waits no more than 4096 VCLK3 clocks for completion of a bus access to system memory. If the bus cycle does not complete in that time, the USB host controller signals an unrecoverable error.
		1	When 1, the USB host controller bus time-out counter is disabled and the host controller waits indefinitely for completion of a USB host controller access to system memory.

32.2.4.27 Host Revision Register (HOSTREVISION)

The host revision register returns the revision number for the USB host controller. This register is not defined by the OHCI specification.

Figure 32-27. Host Revision Register (HOSTREVISION) [address = FCF78BECh]

31-16
Reserved
R-x

15-8	7-4	3-0
Reserved	MAJORREV	MINORREV
R-x	R-x	R-x

LEGEND: R = Read only; -*n* = value at reset; -x = value is indeterminate after reset

Table 32-29. Host Revision Register (HOSTREVISION) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	x	Reserved
7-4	MAJORREV	0-Fh	Major revision number MAJORREV indicates the major revision number of the USB host controller. The original USB host controller version implements a major revision number of 0.
3-0	MINORREV	0-Fh	Minor revision number MINORREv indicates the minor revision number of the USB host controller. The original USB host controller version implements a minor revision number of 0.

32.2.5 USB Host Controller Reserved Registers and Reserved Bit Fields

To enhance code reusability with possible future versions of the USB host controller, reads and writes to reserved USB host controller register addresses are to be avoided. Unless otherwise specified, when writing registers that have reserved bits, read-modify-write operations must be used so that the reserved bits are written with their previous values.

32.2.6 USB Host Controller Registers, USB Reset, and USB Clocking

The Global Clock Module (GCM) provides VCLK3 to the USB host controller for the bus accesses. The GCM also provides the 48MHz clock (VCLKA3) and 12MHz clock (VCLKA3_DIVR) to the USB host controller. If the application disables the VCLK3 domain or the VCLKA3 domain from the GCM, reads from and writes to the USB host controller registers do not occur correctly. To properly access the USB host controller registers, the USB host controller must be clocked and must be out of reset.

The USB host controller hardware reset is controlled by the system reset signal from the SYS module. The USB host controller is held in reset whenever the system reset signal is low or the HostControllerReset (HCR) field of the HcCommandStatus Register is 1. The USB host controller can transition out of reset whenever the clock is enabled and system reset is high and HcCommandStatus.HCR is 0.

The USB host controller completes its reset within about 72 cycles of the 48-MHz clock after the host controller clock transitions out of reset. After system software turns on the clock to the USB host controller and removes it from reset, it is necessary to wait until the USB host controller internal reset completes. To ensure that the USB host controller has completely reset, system software must wait until reads of both the HCREVISION register and the HCHCCA register return their correct reset default values.

32.2.7 OHCI Interrupts

The USB host controller provides an interrupt output to the Vectored Interrupt Manager (VIM). This is a level-sensitive interrupt signal.

32.2.7.1 OHCI Scheduling Overrun Interrupt

The OHCI scheduling overrun interrupt is supported as described in the *OHCI Specification for USB*.

32.2.7.2 OHCI HCDONEHEAD Writeback Interrupt

The OHCI HCDONEHEAD writeback interrupt is supported as described in the *OHCI Specification for USB*.

32.2.7.3 OHCI Start Of Frame Interrupt

The OHCI start of frame interrupt is supported as described in the *OHCI Specification for USB*.

32.2.7.4 OHCI Resume Detect Interrupt

The OHCI resume detect interrupt is supported as described in the *OHCI Specification for USB*.

32.2.7.5 OHCI Unrecoverable Error Interrupt

The OHCI unrecoverable error interrupt is supported as described in the *OHCI Specification for US*B. This interrupt occurs if the USB host controller is unable to complete a bus read or write within 4096 VCLK3 clocks when the USB host bus time-out feature is enabled. When a bus time-out causes an unrecoverable error, HOSTUEADDR and HOSTUESTATUS are updated. When an isochronous TD is processed with an OFFSET/PSW field that is not set for *Not Accessed*, an unrecoverable error interrupt is generated, but HOSTUEADDR and HOSTUESTATUS are not updated.

32.2.7.6 OHCI Frame Number Overflow

The OHCI frame number overflow interrupt is supported as described in the *OHCI Specification for USB*.

32.2.7.7 OHCI Root Hub Status Change

The OHCI root hub status change interrupt is supported as described in the *OHCI Specification for USB*.

32.2.7.8 OHCI Ownership Change Interrupt

The optional OHCI ownership change interrupt is not supported.

32.2.8 USB Host Controller Access to System Memory

The USB host controller has access to system memory to read and write the OHCI data structures and data buffers associated with USB traffic.

32.2.9 Physical Addressing

All system memory accesses initiated by the USB host controller use physical addresses.

32.2.10 USB Host Controller Bus Addressing and OHCI Data Structure Pointers

The USB host controller OHCI registers that point to the HCCA and the ED lists must be programmed with values that correspond to the physical addresses of the particular data structure. System software must use physical addresses when manipulating the OHCI control registers that point to the HCCA, and to the ED and TD lists. System software must also use physical addresses for the ED and TD pointers that are stored within ED and TD entries.

The USB host controller driver software must also be able to examine the list of completed transfer descriptors that the host controller creates as it retires transfer descriptors. This list is pointed to by the HCDONEHEAD register, which contains a physical address that points to the most recent transfer descriptor that has been retired.

32.2.11 NULL Pointers

The *OHCI Specification for USB* uses NULL pointers to indicate the end of a list. The USB host controller compares the ED and TD pointers against the value 0x00000000 to determine if the pointer is a null pointer. Address conversion routines must understand this usage.

32.2.12 USB Host Controller Power Management

Power management of the USB host controller is limited to disabling the clock to the USB host by clearing UHOST_EN. When UHOST_EN is 0, the USB host controller clocks are disabled and the USB host controller is held in reset. The USB signal multiplexing controlled by register in the I/O Multiplexing Module (IOMM).

When the host controller 48-MHz clock is disabled or UHOST_EN is 0, all USB host controller OHCI registers and the HOSTUEADDR, HOSTUESTATUS, HOSTTIMEOUTCTRL, and HOSTREVISION registers are inaccessible.

32.3 USB Device Controller

The USB device controller supports the implementation of a full-speed device fully compliant with the USB 1.1 standard.

It provides an interface between the host processor and the USB wire and handles USB transactions with minimal CPU software intervention.

The USB device controller module supports one control endpoint (EP0), up to 15 IN endpoints, and up to 15 OUT endpoints. The exact endpoint configuration is software programmable. The specific items of a configuration are for each endpoint, the size in bytes, the direction (IN, OUT), the type (bulk/interrupt or ISO), and the associated number.

The USB device controller module also supports three DMA channels for IN endpoints and three DMA channels for OUT endpoints for either bulk/interrupt or ISO transactions.

32.3.1 USB Device Controller Registers

Table 32-30 lists the USB device controller registers. These registers are aligned at a 16-bit boundary, so that the memory address offsets for the registers are 0, 2h, 4h, 6h, and so on. The registers support 8-bit and 16-bit read/write accesses. A 32-bit read access will return the same contents on the upper 16 bits as on the lower 16 bits. Section 32.3.1.1 through Section 32.3.1.23 describe the register bits.

Table 32-30. USB Device Controller Registers

Address	Register	Description	Section
FCF7 8A00h	REV	Revision	Section 32.3.1.1
		Endpoint	
FCF7 8A02h	EP_NUM	Endpoint selection	Section 32.3.1.2
FCF7 8A04h	DATA	Data	Section 32.3.1.3
FCF7 8A06h	CTRL	Control	Section 32.3.1.4
FCF7 8A08h	STAT_FLG	Status	Section 32.3.1.5
FCF7 8A0Ah	RXFSTAT	Receive FIFO status	Section 32.3.1.6
FCF7 8A0Ch	SYSCON1	System configuration 1	Section 32.3.1.7
FCF7 8A0Eh	SYSCON2	System configuration 2	Section 32.3.1.8
FCF7 8A10h	DEVSTAT	Device status	Section 32.3.1.9
FCF7 8A12h	SOF	Start of frame	Section 32.3.1.10
FCF7 8A14h	IRQ_EN	Interrupt enable	Section 32.3.1.11
FCF7 8A16h	DMA_IRQ_EN	DMA interrupt enable	Section 32.3.1.12
FCF7 8A18h	IRQ_SRC	Interrupt source	Section 32.3.1.13
FCF7 8A1Ah	EPN_STAT	Non-ISO endpoint interrupt enable	Section 32.3.1.14
FCF7 8A1Ch	DMAN_STAT	Non-ISO DMA interrupt enable	Section 32.3.1.15
		DMA Configuration	
FCF7 8A20h	RXDMA_CFG	DMA receive channels configuration	Section 32.3.1.16
FCF7 8A22h	TXDMA_CFG	DMA transmit channels configuration	Section 32.3.1.17
FCF7 8A24h	DATA_DMA	DMA FIFO data	Section 32.3.1.18
FCF7 8A26h	TXDMA0	Transmit DMA control 0	Section 32.3.1.19
FCF7 8A28h	TXDMA1	Transmit DMA control 1	Section 32.3.1.19
FCF7 8A2Ah	TXDMA2	Transmit DMA control 2	Section 32.3.1.19
FCF7 8A30h	RXDMA0	Receive DMA control 0	Section 32.3.1.20
FCF7 8A32h	RXDMA1	Receive DMA control 1	Section 32.3.1.20
FCF7 8A34h	RXDMA2	Receive DMA control 2	Section 32.3.1.20
		Endpoint Configuration	
FCF7 8A40h	EP0	Endpoint configuration 0	Section 32.3.1.21
FCF7 8A42h- FCF7 8A5Eh	EP1_RX to EP15_RX	Receive endpoint configuration 1 to 15	Section 32.3.1.22
FCF7 8A62h- FCF7 8A7Eh	EP1_TX to EP15_TX	Transmit endpoint configuration 1 to 15	Section 32.3.1.23

32.3.1.1 Revision Register (REV)

This read-only register contains the revision number of the module. A write to this register is denied. Device reset and USB controller reset have no effect on this register.

Figure 32-28. Revision Register (REV) [address = FCF78A00h]

15	8
Reserved	
R-0	

7	0
REV_NB	
R-61h	

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

Table 32-31. Revision Register (REV) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved. Reads return zeros, writes have no effect.
7-0	REV_NB		This 8-bit field indicates revision number of current USB device controller module—value fixed by hardware:
		61h	Revision 6.1

32.3.1.2 Endpoint Selection Register (EP_NUM)

This read/write register selects and enables the endpoint that can be accessed by the USB device controller.

Figure 32-29. Endpoint Selection Register (EP_NUM) [address = FCF78A02h]

15	8
Reserved	
R-0	

7	6	5	4	3	0
Reserved	SETUP_SEL	EP_SEL	EP_DIR	EP_NUM	
R-0	RW-0	RW-0	RW-0	RW-0	

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

Table 32-32. Endpoint Selection Register (EP_NUM) Field Descriptions

Bit	Field	Value	Description
15-7	Reserved	0	Reserved. Reads return zeros, writes have no effect.
6	SETUP_SEL		Set by the CPU in response to a Setup general USB Interrupt, to access the EP0 read-only Setup FIFO when reading the DATA register. Setting this bit will clear the IRQ_SRC.Setup bit. When this bit is set, other EP_NUM register bits must be 0. Note: After having read the Setup FIFO, the CPU must clear this bit by writing 0 to it.
		0	No access
		1	Access permitted
			The value after system reset or USB reset is low.
5	EP_SEL		Set by the CPU to access the status (STAT_FLG, RXFSTAT) and data (DATA) registers for the endpoint selected. If the EP_NUM.EP_DIR bit is set to 0, the CPU can read data from endpoint RX FIFO by reading the DATA register. If the EP_NUM.EP_DIR bit is set to 1, the CPU can write data into endpoint TX FIFO by writing into the DATA register. After each access to an endpoint during interrupt handling, the CPU must absolutely clear this bit. Note: When the CPU sets this bit, it must set the SETUP_SEL bit to 0. After having accessed the endpoint's FIFO for either read or write access, the CPU must clear this bit by writing a 0 to it.
		0	No access
		1	Access permitted.
			Value after system reset or USB reset is low.
4	EP_DIR		The endpoint direction bit gives the direction associated with the endpoint number selected in EP_NUM.EP_NUM:
		0	OUT endpoint
		1	IN endpoint
			Value after system reset or USB reset is low.
3-0	EP_NUM		The endpoint number binary encoded in these four bits, associated with the direction given by EP_NUM.EP_DIR bit, is the current endpoint selected. All reads and writes to the endpoint status and the control and data locations are for this endpoint.
		0	EP0
		1h	EP1
		:	:
		Fh	EP15
			Value after system reset or USB reset is low.

32.3.1.3 Data Register (DATA)

This register is the entry point to write into a selected TX endpoint or to read data from a selected RX endpoint, or to read data from setup FIFO. If selected endpoint direction is OUT, this register is read-only and a write into it is denied. If selected endpoint direction is IN, this register is write-only and a read to this register is denied.

Figure 32-30. Data Register (DATA) [address = FCF78A04h]

<table>
<tr><td>15</td><td align="right">0</td></tr>
<tr><td colspan="2" align="center">DATA</td></tr>
<tr><td colspan="2" align="center">R/W-0</td></tr>
</table>

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

Table 32-33. Data Register (DATA) Field Descriptions

Bit	Field	Description
15-0	DATA	Transmit/receive FIFO data: EP_NUM.EP_DIR = 1: This register contains the data written by the USB device controller to be sent to the USB host during the next IN transaction. Data can be written successfully only if the EP_NUM.EP_SEL bit is asserted. EP_NUM.EP_DIR = 0: This register contains the data received by the USB core from USB host OUT or SETUP transactions. Data can be read successfully only if the EP_NUM.EP_SEL bit is asserted, or if EP_NUM.SETUP_SEL bit is asserted (for setup data).

32.3.1.4 Control Register (CTRL)

This set-only register controls the FIFO and status of the selected endpoint. A read access to this register always returns 0.

Endpoint 0 setup FIFO is always enabled and ready to accept setup data. No control register CTRL is implemented for this FIFO because the USB device controller cannot control it.

Figure 32-31. Control Register (CTRL) [address = FCF78A06h]

15 ... 8
Reserved
R-0

7	6	5 ... 3	2	1	0
CLR_HALT	SET_HALT	Reserved	SET_FIFO_EN	CLR_EP	RESET_EP
RW-0	RW-0	R-0	RW-0	RW-0	RW-0

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

Table 32-34. Control Register (CTRL) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved
7	CLR_HALT		The clear halt endpoint (non-ISO) bit only concerns non-ISO endpoints—used by the USB device controller to clear an endpoint halt condition:
		0	No action
		1	Clear halt condition
			Always read 0.
6	SET_HALT		The set halt endpoint (non-ISO) only concerns non-ISO endpoints—used by the USB device controller to halt the selected endpoint. The halted endpoint returns STALL handshakes to the USB host. The USB device controller can disable the endpoint interrupt if it does not want to be informed of STALL handshakes. Note: If the endpoint to halt is used by a DMA channel, the USB device controller must disable the DMA channel before setting halt condition for this endpoint.
		0	No action
		1	Halt endpoint
			Always read 0.
5-3	Reserved	0	Reserved. Reads return zeros, writes have no effect.
2	SET_FIFO_EN		The set FIFO enable (non-ISO) bit only concerns non-ISO endpoints. If the selected endpoint direction is IN, the USB device controller uses this bit to enable the USB device to transmit data from the FIFO at the next valid IN token. If the selected endpoint direction is OUT, the USB device controller uses this bit to enable the USB device to receive data from the USB host at the next valid OUT transaction. If not, setting the device returns a NAK handshake. ISO endpoints FIFO are always enabled. Note: The USB device controller must never enable endpoint 0 FIFO out of control transfers. For bulk and interrupt endpoints, FIFO must never be enabled when the halt feature is set or when RX FIFO is not empty. Furthermore, during EP interrupts handling, the USB device controller must have cleared the interrupt bit before setting CTRL.SET_FIFO_EN bit (to avoid masked ACK interrupts).
		0	No action
		1	FIFO enabled
			Always read 0.
1	CLR_EP		Clear endpoint: the USB device controller sets this bit to clear the selected endpoint FIFO pointers and flags. This bit resets the FIFO pointers, the FIFO empty status bit is set, and the FIFO enable bit and other FIFO flags are cleared upon completion of the FIFO reset. It also clears the previous transaction handshake status. For ISO endpoints or non-ISO double-buffered endpoints, both foreground and background FIFO are cleared.
		0	No action
		1	Clear endpoint
			Always read 0.

Table 32-34. Control Register (CTRL) Field Descriptions (continued)

Bit	Field	Value	Description
0	RESET_EP		The endpoint reset (non-control) bit only concerns non-control endpoints. The USB device controller sets this bit to reset the selected endpoint. It forces an interrupt or a bulk endpoint data PID to DATA0, clears the halt condition, and clears the FIFO (both foreground and background if endpoint is double-buffered) and previous transactions handshake status. For an ISO endpoint, it only clears the FIFO (both foreground and background).
		0	No action
		1	Reset endpoint
			Always read 0.

32.3.1.5 Status Register (STAT_FLG)

This read-only register provides a status of the FIFO and the results of the transactions handshakes for the selected endpoint. The 8 MSB are reserved for ISO endpoints, whereas the 8 LSB are reserved for non-ISO endpoints. This register cannot be read if EP_NUM.EP_SEL bit is not asserted for the endpoint. No status flag exists for the read-only set-up FIFO, which is always enabled.

NOTE: The updates for non-ISO transactions are done at the end of each non-transparent and valid transaction to a given endpoint, if no non-handled interrupt is pending on the endpoint.

The definition of a non-transparent, non-ISO IN transaction is one that responds with an ACK handshake or a STALL handshake, or optionally a NAK handshake if SYSCON1.NAK_EN is asserted to 1. An ERR handshake or a NAK handshake when SYSCON1.NAK_EN is 0 is considered transparent.

A write to this register has no effect.

Figure 32-32. Status Register (STAT_FLG) [address = FCF78A08h]

15	14	13	12	11	10	9	8
NO_RX-PACKET	MISS_IN	DATA_FLUSH	ISO_ERR	Reserved		ISO_FIFO_EMPTY	ISO_FIFO_FULL
R-0	R-0	R-0	R-0	R-0		R-0	R-0

7	6	5	4	3	2	1	0
Reserved	EP_HALTED	STALL	NAK	ACK	FIFO_EN	NON_ISO_FIFO_EMPTY	NON_ISO_FIFO_FULL
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

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

Table 32-35. Status Register (STAT_FLG) Field Descriptions

Bit	Field	Value	Description
15	NO_RXPACKET		The isochronous no packet received (ISO OUT) bit only concerns ISO OUT endpoints. This bit notifies the USB device controller that the core has not received any isochronous packet on the endpoint. This bit is updated on a SOF (start of frame).
		0	The endpoint received a packet on the previous frame.
		1	The endpoint did not receive a packet on the previous frame.
			Value after system reset or USB reset is high.

Table 32-35. Status Register (STAT_FLG) Field Descriptions (continued)

Bit	Field	Value	Description
14	MISS_IN		The isochronous missed IN token for the previous frame (ISO IN) bit only concerns ISO IN endpoints. This bit notifies the USB device controller that the core missed a valid ISO IN token during the previous frame and that TX data were flushed from the FIFO instead of being transmitted to the USB host.
		0	This bit is updated on a SOF (start of frame).
		1	The endpoint received an IN token the previous frame.
			The endpoint did not receive an IN token the previous frame and TX data were flushed.
			Value after system reset or USB reset is low.
13	DATA_FLUSH		The isochronous receive data flush (ISO OUT) bit only concerns ISO OUT endpoints. When set, this bit indicates that data were flushed from the isochronous FIFO that was moved from the foreground to the background. This happens when the USB device controller does not read all of the data from the foreground FIFO in a frame.
			This bit is updated in every frame.
		0	Not significant.
		1	Data was flushed.
			Value after system reset or USB reset is low.
12	ISO_ERR		The isochronous receive data error (ISO OUT) bit only concerns ISO OUT endpoints. When set, this bit indicates that the ISO data packet was received incorrectly. This happens when the core detects an error in the data packet (CRC, bit stuffing, PID check) or when there is an overrun condition in the FIFO. When this bit is set, the FIFO contents are automatically flushed by the core and the FIFO status is empty.
			This bit is updated in every frame.
		0	Not significant.
		1	ISO packet received with errors.
			Value after system reset or USB reset is low.
11-10	Reserved	0	Reserved
9	ISO_FIFO_EMPTY		The ISO FIFO empty bit only concerns ISO endpoints. This bit is set when the FIFO for the selected ISO endpoint is empty, either via an appropriate CTRL.CLR_EP bit or CTRL.RESET_EP bit or after successful reads from the selected FIFO.
		0	ISO FIFO is not empty.
		1	ISO FIFO is empty.
			Value after system reset or USB reset is high (FIFO empty).
8	ISO_FIFO_FULL		The ISO FIFO full bit only concerns ISO endpoints. This bit is set when the FIFO for the selected ISO endpoint is full. This condition is cleared by setting the CTRL.CLR_EP bit or CTRL.RESET_EP bit, or after one successful read (by the USB device controller or the USB host).
		0	ISO FIFO is not full.
		1	ISO FIFO is full.
			Value after system reset or USB reset is low (FIFO empty).
7	Reserved	0	Reserved
6	EP_HALTED		The endpoint halted flag (non-ISO) bit only concerns non-ISO endpoints. This bit when asserted 1 indicates that the selected endpoint is halted. The endpoint can be put into the halt state only by the USB device controller writing the endpoint halt control bit (in response to a SET_FEATURE request, for instance).
		0	The selected endpoint is not halted.
		1	The selected endpoint is halted.
			Value after system reset or USB reset is low.

Table 32-35. Status Register (STAT_FLG) Field Descriptions (continued)

Bit	Field	Value	Description
5	STALL		The transaction stall (non-ISO) bit only concerns non-ISO endpoints. This status bit is set at the end of a transaction if a STALL handshake packet was returned to the USB host, and if no non-handled interrupt is pending on the current buffer (see Section 32.3.11). The core automatically returns a STALL packet if a valid IN token is received by a halted TX endpoint, if a valid OUT transaction is received by a halted RX endpoint, or if there is a request error (endpoint 0). The bit is cleared when the USB device controller has finished handling the corresponding interrupt (at EP_NUM.EP_SEL bit deselection).
		0	No STALL handshake was returned.
		1	A STALL handshake packet was returned.
			Value after system reset or USB reset is low.
4	NAK		The transaction non-acknowledge (non-ISO) bit only concerns non-ISO endpoints with SYSCON1.NAK_EN bit asserted. This status bit is set at the end of a transaction if a NAK handshake is returned to the USB host, and if no non-handled interrupt is pending on the current buffer. The USB core automatically returns a NAK handshake to the USB host if a valid IN token is received by a TX endpoint or if a valid OUT transaction is received by an RX endpoint, and the STAT_FLG.FIFO_EN bit is not set for the endpoint. The bit is cleared when the USB device controller has finished handling the corresponding interrupt (at EP_NUM.EP_SEL bit deselection).
		0	No NAK handshake was returned (SYSCON1.NAK_EN bit is set).
		1	A NAK handshake packet was returned and SYSCON1.NAK_EN bit is set.
			Value after system reset or USB reset is low.
3	ACK		The transaction acknowledge (non-ISO) bit only concerns non-ISO endpoints. This bit is set at the end of a non-transparent valid IN transaction if the data packet was sent successfully to the USB host and the ACK handshake was received, or at the end of a non-transparent valid OUT transaction if the data packet was received successfully by the USB device and the ACK handshake was returned. The bit is cleared when the USB device controller has finished handling the corresponding interrupt (at EP_NUM.EP_SEL bit deselection).
		0	No ACK handshake packet was returned.
		1	An ACK handshake packet was returned.
			Value after system reset or USB reset is low.
2	FIFO_EN		The FIFO enable status (non-ISO) bit only concerns non-ISO endpoints. This bit is asserted when CTRL.SET_FIFO_EN is set to 1 and is cleared automatically after a transaction completes with an ACK, STALL.
		0	The non-ISO endpoint FIFO is disabled.
		1	The non-ISO endpoint FIFO is enabled.
			Value after system reset or USB reset is low.
1	NON_ISO_FIFO_EMPTY		The non-ISO FIFO empty bit only concerns non-ISO endpoints. This bit is set when the FIFO for the selected non-ISO endpoint is empty, either via an appropriate CTRL.CLR_EP bit or CTRL.RESET_EP bit or after successful reads from the selected FIFO.
		0	Non-ISO FIFO is not empty.
		1	Non-ISO FIFO is empty.
			Value after system reset or USB reset is high (FIFO empty).
0	NON_ISO_FIFO_FULL		The non-ISO FIFO full bit only concerns non-ISO endpoints. This bit is set when the FIFO for the selected non-ISO endpoint is full. This condition is cleared by setting the CTRL.CLR_EP bit or CTRL.RESET_EP bit, or after one successful read (by the USB device controller or the USB host).
		0	Non-ISO FIFO is not full.
		1	Non-ISO FIFO is full.
			Value after system reset or USB reset is low (FIFO empty).

32.3.1.6 Receive FIFO Status Register (RXFSTAT)

This read-only register indicates how many bytes are in the receive FIFO for the selected endpoint. A write to this register has no effect. The USB device controller cannot read this register if the EP_NUM.EP_SEL bit is not set for the endpoint. No receive FIFO status exists for the setup FIFO, because 8 bytes are always expected.

Figure 32-33. Receive FIFO Status Register (RXFSTAT) [address = FCF78A0Ah]

15-10	9-8
Reserved	RXF_COUNT
R-0	R-0

7-0
RXF_COUNT
R-0

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

Table 32-36. Receive FIFO Status Register (RXFSTAT) Field Descriptions

Bit	Field	Value	Description
15-10	Reserved	0	Reserved
9-0	RXF_COUNT	0-3FFh	The receive FIFO byte count 10-bit field indicates the number of bytes currently in the receive FIFO. Values after system reset or USB reset is low (all 10 bits).

32.3.1.7 System Configuration Register 1 (SYSCON1)

This read/write register provides control functions for power management and miscellaneous control for the core.

Values depend on whether the reset action comes from the USB device controller (CPU) or the USB host.

Figure 32-34. System Configuration Register 1 (SYSCON1) [address = FCF78A0Ch]

15-9	8
Reserved	CFG_LOCK
R-0	R/W-0

7	6	5	4	3	2	1	0
DATA_ENDIAN	DMA_ENDIAN	Reserved	NAK_EN	AUTODEC_DIS	SELF_PWR	SOFF_DIS	PULLUP_EN
R/W-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 at reset

Table 32-37. System Configuration Register 1 (SYSCON1) Field Descriptions

Bit	Field	Value	Description
15-9	Reserved	0	Reserved
8	CFG_LOCK		Device configuration locked bit: after the USB device controller has entered the device configuration (registers 0x20 to 0x3F), it must set the CFG_LOCK bit so that the device can be used. When the device configuration is not locked, the device is not ready to be used.
		0	Device configuration is not locked. Device is not ready
		1	Device configuration is locked.
			The value after the USB device controller hardware reset is low; after USB reset, it is unchanged (keep previous configuration).
7	DATA_ENDIAN		The data endian bit can be set by the USB device controller to select little- or big-endian format on data access (read or write). See Table 32-38.
		0	Little endian
		1	Big endian
			The value after USB device controller hardware reset is low; after USB reset, it is unchanged (keep previous configuration).
6	DMA_ENDIAN		The DMA data endian bit can be set by the USB device controller to select little- or big-endian format on data access (read or write). See Table 32-38.
		0	Little endian
		1	Big endian
			The value after USB device controller hardware reset is low; after USB reset, it is unchanged (keep previous configuration).
5	Reserved	0	Reserved
4	NAK_EN		The NAK enable bit can be set by the USB device controller to be signaled for NAK transaction handshake response. When this bit is set, STAT_FLG.NAK is set on a NAK handshake if no non-handled interrupt is pending on the endpoint, and the endpoint interrupt is asserted. In the normal mode, when cleared, NAK handshake response to the USB host is made transparent to the USB device controller and no interrupt is asserted.
		0	NAK is disabled.
		1	NAK is enabled.
			The value after system reset or USB reset is low.
			Note: If the USB device controller sets this bit, it must wait for a NAK interrupt before selecting TX endpoint to write TX data.
3	AUTODEC_DIS		The autodecode process disabled can be set to force software to handle all EP0 transactions (except the SET_ADDRESS transaction).
		0	Autodecode process is activated (see Table 32-55).
		1	Autodecode process is deactivated. The value after USB device controller hardware reset is low; after USB reset, it is unchanged.

Table 32-37. System Configuration Register 1 (SYSCON1) Field Descriptions (continued)

Bit	Field	Value	Description
2	SELF_PWR		The self-powered bit indicates to the USB host whether the device is bus-powered or self-powered. This information is needed for an autodecoded request. The USB device controller must update this bit after a SET_CONFIGURATION according to the self-powered bit D6 given in the configuration descriptor (see *USB 1.1 Specification*).
		0	Bus powered
		1	Self-powered
			The value after USB device controller hardware reset is low; after USB reset, it is unchanged.
1	SOFF_DIS		When the shutoff disable bit is set, it disables the power shutoff circuitry.
		0	Power shutoff circuitry is enabled.
		1	Power shutoff circuitry is disabled.
			The value after USB device controller hardware reset is low; after USB reset, it is unchanged.
0	PULLUP_EN		The external pullup enable bit allows the device to disconnect itself from the USB bus, forcing the host to reset and reconfigure the device. This bit can be used to prevent USB traffic when the device is not ready.
		0	Pull-up is disabled. USB host cannot detect the device. In this mode, the 48-MHz USB clock is forced off.
		1	Pull-up is enabled.
			The value after USB device controller hardware reset is low; after USB reset, it is high; and after detach, it is unchanged.

Table 32-38. Little-Endian and Big-Endian Formats

Little-Endian Format (16 bits):			**Big-Endian Format (16 bits):**	
15 8	7 0		15 8	7 0
Byte 1	Byte 0		Byte 0	Byte 1

32.3.1.8 System Configuration Register 2 (SYSCON2)

This set-only register provides miscellaneous controls for the function. A read to this register always returns 0.

Figure 32-35. System Configuration Register 2 (SYSCON2) [address = FCF78A0Eh]

15 – 8
Reserved
R-0

7	6	5	4	3	2	1 – 0
Reserved	RMT_WKP	STALL_CMD	Reserved	DEV_CFG	CLR_CFG	Reserved
R-0	W-0	W-0	R-0	W-0	W-0	R-0

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

Table 32-39. System Configuration Register 2 (SYSCON2) Field Descriptions

Bit	Field	Value	Description
15-7	Reserved	0	Reserved
6	RMT_WKP		The set-only remote wake-up bit, when written with a 1, initiates the remote wake-up sequence even if DEVSTAT.R_WK_OK bit was not previously set to 1 by the USB host. Reading this bit always returns 0. Writing 0 into this bit has no effect. To generate a resume, the software must check the remote wake-up enable value before initiating any wake-up sequence.
		0	No action.
		1	Initiates the remote wake-up sequence.
			Always read 0.
5	STALL_CMD		The set-only stall command bit only concerns non-autodecoded requests on control endpoint (EP0). This is asserted in response to a USB command where either the command itself or its data is invalid. Asserting this bit forces the non-autodecoded command to complete with a STALL handshake. It has no effect for autodecoded requests.
		0	No action.
		1	Stall current USB command.
			Always read 0.
4	Reserved	0	Reserved
3	DEV_CFG		If the USB device controller receives a SET_CONFIGURATION with a valid configuration value, and the device is in addressed state, it must write a 1 to the device configured (DEV_CFG) bit to inform the command decode that the device moves to configured state. The core sets the DEVSTAT.CFG bit to 1. If the device is already configured when the SET_CONFIGURATION request is received, the USB device controller does not have to set this bit. If the new configuration value is 0, USB device controller must set the SYSCON2.CLR_CFG bit to move to addressed state. Reading this bit always returns 0. Writing 0 into this bit has no effect.
		0	No action.
		1	Allows DEVSTAT.CFG to be set.
			Always read 0.
2	CLR_CFG		If the USB device controller receives a SET_CONFIGURATION with a configuration value of 0, and if the device is configured, it must write a 1 to the clear configured (CLR_CFG) bit to inform the command decode that the device becomes deconfigured (moves to addressed state). The core clears the DEVSTAT.CFG bit. Reading this bit always returns 0. Writing 0 into this bit has no effect.
		0	No action.
		1	Allows DEVSTAT.CFG to be cleared.
			Always read 0.
1-0	Reserved	0	Reserved

32.3.1.9 Device Status Register (DEVSTAT)

This read-only register provides a status reflecting the visible device states as defined in *USB1.1 Specification*. A write to this register has no effect.

NOTE: This register is double buffered. If IRQ_EN.DS_CHG_IE is set (interrupt enabled), the background register is moved into foreground position only after clearing any pending DS_CHG interrupt. Therefore, if there is a state change and a pending DS_CHG interrupt has not been serviced yet, then the recent state change is not visible, because the background register was only updated and not moved into the foreground position.

The values depend on whether the reset action comes from the USB device controller (CPU) or USB host. They also depend on whether the device is attached or not when the USB device controller hardware reset event occurs.

Figure 32-36. Device Status Register (DEVSTAT) [address = FCF78A10h]

15	10	9	8
Reserved		B_HNP_ ENABLE	A_HNP_ SUPPORT
R-0		R-0	R-0

7	6	5	4	3	2	1	0
A_ALT_HNP_ SUPPORT	R_WK_OK	USB_RESET	SUS	CFG	ADD	DEF	ATT
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

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

Table 32-40. Device Status Register (DEVSTAT) Field Descriptions

Bit	Field	Value	Description
15-10	Reserved	0	Reserved
9	B_HNP_ENABLE		The HNP enable for B-device bit is used for On-The-Go feature selection purposes. See the *On-The-Go Supplement* to the *USB 2.0 Specification*.
		0	Not significant.
		1	HNP enable for B-device.
			Value after system reset or USB reset is low.
8	A_HNP_SUPPORT		The B-device is directly connected to an A-device port that supports HNP. This A_HNP_SUPPORT bit is used for On-The-Go feature selection purposes. See the *On-The-Go Supplement* to the *USB 2.0 Specification*.
		0	Not significant.
		1	B-device connection to HNP capable A-device.
			Value after system reset or USB reset is low.
7	A_ALT_HNP_ SUPPORT		The B-device is connected to an A-device port that is not capable of HNP, but the A-device has an alternate port that is capable of HNP. This A_ALT_HNP_SUPPORT bit is used for On-The-Go feature selection purposes. See the *On-The-Go Supplement* to the *USB 2.0 Specification*.
		0	Not significant.
		1	B-device connection to not HNP capable A-device.
			Value after system reset or USB reset is low.
6	R_WK_OK		The remote wake-up granted bit is automatically set and cleared when the core receives a valid set/clear device feature request from the USB host. It returns a 1 when the USB host grants the function the ability to assert remote wake-up.
		0	Not significant.
		1	Remote wake-up granted.
			Value after system reset or USB reset is low.

Table 32-40. Device Status Register (DEVSTAT) Field Descriptions (continued)

Bit	Field	Value	Description
5	USB_RESET		USB reset signaling is active. The USB_RESET bit returns 1 when the USB host resets the USB bus. A valid USB reset resets the endpoint FIFOS, the other control register bits, except for SYSCON1.CFG_LOCK and the associated configuration registers (0x20 to 0x3F), IRQ_EN.DS_CHG_IE and IRQ_SRC.DS_CHG, and forces the device to the default state for DEVSTAT (this register). This bit is cleared at the end of reset. This bit, as well as other DEVSTAT bits, is double-buffered. If a pending interrupt has not been handled when a USB reset occurs and is handled only when USB reset is finished, the USB device controller does not see the USB_RESET bit going high and then low.
		0	Device is not being reset by USB host.
		1	Device is being reset by USB host.
			The value after the USB device controller hardware reset is low, and during USB reset is high (low after USB reset).
4	SUS		Suspended state bit: the device in suspended state is, at a minimum, attached to the USB, powered, has been reset by the USB host, and has not seen bus activity for 5 ms. It can also have a unique address and be configured for use. Because the device is suspended, however, the host cannot use the device function. This bit returns 1 when the USB device is in a suspended state.
		0	Not suspended
		1	Suspended
			Value after system reset or USB reset is low.
3	CFG		Configured state bit: the device is attached to the USB, powered, has been reset, has a unique address, and is configured. The host can use the function provided by the device. This bit returns 1 when the USB device has been configured after a set SYSCON2.DEV_CFG = 1. This bit remains set to 1 until the device becomes deconfigured. This bit is cleared when the core receives a valid SET_CONFIGURATION request and the USB device controller sets the SYSCON2.CLR_CFG bit. During the time this bit is not set to 1, transactions that are not for control EP0 are ignored. A GET_ENDPOINT_STATUS to a non-control endpoint is stalled.
		0	Not configured
		1	Configured
			Value after system reset or USB reset is low.
2	ADD		Addressed state bit: the device is attached to the USB, powered, has been reset, and a unique device address has been assigned. This bit (ADD) returns 1 after an ET_ADDRESS standard request. This bit remains set to 1 until the device becomes de-addressed.
		0	Not addressed
		1	Addressed
			Value after system reset or USB reset is low.
1	DEF		The default state bit returns 1 when the USB device is attached to the USB and powered, and has been reset. This bit remains set to 1 until the device becomes de-powered. The device moves into default state as soon as the USB reset is effective.
		0	Not in default
		1	Default
			The value after CPU is low, and after USB reset is high.
0	ATT		The attached state bit returns 1 when the device is attached to the USB and is powered. This bit remains set to 1 until the device powers down.
		0	Not attached
		1	Attached
			The value after USB device controller hardware reset is low (unattached) or high (attached), and after USB reset is high.

32.3.1.10 Start of Frame Register (SOF)

This read-only register provides a frame timer status for use in ISO communications. A write to this register is denied.

Figure 32-37. Start of Frame Register (SOF) [address = FCF78A12h]

15 13	12	11	10 8
Reserved	FT_LOCK	TS_OK	TS
R-0	R-0	R-0	R-0

7 0
TS
R-0

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

Table 32-41. Start of Frame Register (SOF) Field Descriptions

Bit	Field	Value	Description
15-13	Reserved	0	Reserved
12	FT_LOCK		Frame timer locked bit: the USB host sends out a start of frame (SOF) packet every millisecond. When the device does not receive a SOF packet because of a bus error, a local start of-frame that is used for ISO FIFO switch is generated. Once the core receives two valid SOF, separated by TF (time frame), it sets SOF.FT_LOCK to 1, only if TF is in frame interval IF allowed by the USB 1.1 specification (IF = [11964:12036] USB bit time). If TF is out of this interval, SOF.FT_LOCK value remains 0, and a local SOF is generated by the core. When SOF.FT_LOCK is set and the frame timer is locked to the timing TF, a local SOF is generated, if no valid SOF has been received in an interval of TF since the last valid SOF. The SOF.FT_LOCK bit is cleared if a valid SOF is received out of the interval IF. If the core receives a valid SOF in this interval, the frame timer locks to the new frame time. If the core does not receive a valid SOF, the frame timer remains locked to TF. When SOF.FT_LOCK is cleared, a local SOF is generated after the 12036 USB bit time, if no valid SOF has been received, and SOF.FT_LOCK remains 0.
		0	Frame timer is not locked.
		1	Frame timer is locked.
			The value after system reset or USB reset is low. Note: Each time the core receives a valid SOF out of the allowed interval IF, a local SOF is generated and the ISO FIFO switches.
11	TS_OK		The timestamp OK bit indicates that the timestamp in SOF.TS is valid for the current frame. It returns a 1 if a valid SOF packet was received from the USB host and a 0 otherwise.
		0	Timestamp is invalid.
		1	Timestamp is valid.
			Value after system reset or USB reset is low.
10-0	TS	0-7FFh	The timestamp number field returns the timestamp from the last USB host-valid SOF packet. The frame number is valid if SOF.TS_OK is 1. In case of an SOF miss, this value is not updated and TS_OK is cleared. The value after system reset or USB reset is low (all 11 bits).

32.3.1.11 Interrupt Enable Register (IRQ_EN)

This read/write register enables all non-DMA interrupts (control, state changed, ISO, and non-ISO).

The values depend on whether the reset action comes from the USB device controller (CPU) or the USB host.

When the USB device controller sets a bit position to 1, an interrupt is signaled to the USB device controller if the corresponding IRQ_SRC bit is asserted to 1 by the core for any IRQ_SRC bit controlled by this bit. If reset to 0, the interrupt is masked and not signaled to the USB device controller.

- 0: Interrupt is disabled.
- 1: Interrupt is enabled.

The value after system reset or USB reset is low for all bits except IRQ_EN.DS_CHG_IE bit, which remains unchanged after a reset from the USB host.

Figure 32-38. Interrupt Enable Register (IRQ_EN) [address = FCF78A14h]

15							8
Reserved							
R-0							

7	6	5	4	3	2	1	0
SOF_IE	Reserved	EPn_RX_IE	EPn_TX_IE	DS_CHG_IE	Reserved		EP0_IE
R/W-0	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 at reset

Table 32-42. Interrupt Enable Register (IRQ_EN) Field Descriptions

Bit	Field	Value	Description
15-8	Reserved	0	Reserved
7	SOF_IE	0-1	Start of frame interrupt enable
6	Reserved	0	Reserved
5	EPn_RX_IE	0-1	Receive endpoint n interrupt enable (non-ISO)
4	EPn_TX_IE	0-1	Transmit endpoint n interrupt enable (non-ISO)
3	DS_CHG_IE	0-1	Device state changed interrupt enable
2-1	Reserved	0	Reserved
0	EP0_IE	0-1	EP0 transactions interrupt enable

32.3.1.12 DMA Interrupt Enable Register (DMA_IRQ_EN)

This read/write register enables all DMA interrupts.

When the USB device controller sets a bit position to 1, an interrupt is signaled to the USB device controller if the corresponding bit in the IRQ_SRC register is asserted to 1 by the core. If reset to 0, the interrupt is masked and not signaled to the USB device controller.

- 0: Interrupt is disabled.
- 1: Interrupt is enabled.

The value after system reset or USB reset is low for all bits.

Figure 32-39. DMA Interrupt Enable Register (DMA_IRQ_EN) [address = FCF78A16h]

15-11	10	9	8
Reserved	TX2_DONE_IE	RX2_CNT_IE	RX2_EOT_IE
R-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
Reserved	TX1_DONE_IE	RX1_CNT_IE	RX1_EOT_IE	Reserved	TX0_DONE_IE	RX0_CNT_IE	RX0_EOT_IE
R-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 at reset

Table 32-43. DMA Interrupt Enable Register (DMA_IRQ_EN) Field Descriptions

Bit	Field	Value	Description
15-11	Reserved	0	Reserved
10	TX2_DONE_IE	0-1	Transmit DMA channel 2 done interrupt enable
9	RX2_CNTt_IE	0-1	Receive DMA channel 2 transactions count interrupt enable
8	RX2_EOT_IE	0-1	Receive DMA channel 2 end of transfer interrupt enable
7	Reserved	0	Reserved
6	TX1_DONE_IE	0-1	Transmit DMA channel 1 done interrupt enable
5	RX1_CNT_IE	0-1	Receive DMA channel 1 transactions count interrupt enable
4	RX1_EOT_IE	0-1	Receive DMA channel 1 end of transfer interrupt enable
3	Reserved	0	Reserved
2	TX0_DONE_IE	0-1	Transmit DMA channel 0 done interrupt enable
1	RX0_CNT_IE	0-1	Receive DMA channel 0 transactions count interrupt enable
0	RX0_EOT_IE	0-1	Receive DMA channel 0 end of transfer interrupt enable

32.3.1.13 Interrupt Source Register (IRQ_SRC)

This read/clear only register identifies and clears the source of the interrupt signaled by a set flag.

The value depends on whether the reset action comes from the USB device controller (CPU) or the USB host.

The USB device controller can clear a set bit location only by writing a 1 into the bit location (except for the setup bit, which is automatically cleared by the core). A write of 0 has no effect.

When the core sets a bit location to 1, an interrupt is signaled to the USB device controller if the interrupt was enabled.

- 0: No interrupt
- 1: Interrupt signaled

The value after the system reset or USB reset is low except for the IRQ_SRC.DS_CHG bit, which is high after a USB reset.

Figure 32-40. Interrupt Source Register (IRQ_SRC) [address = FCF78A18h]

15	11	10	9	8
Reserved		TXn_DONE	RXn_CNT	RXn_EOT
R-0		R-0	R-0	R-0

7	6	5	4	3	2	1	0
SOF	Reserved	EPn_RX	EPn_TX	DS_CHG	SETUP	EP0_RX	EP0_TX
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

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

Table 32-44. Interrupt Source Register (IRQ_SRC) Field Descriptions

Bit	Field	Value	Description
15-11	Reserved	0	Reserved
10	TXn_DONE		The transmit DMA channel n done interrupt flag (non-ISO) bit is only for non-ISO DMA transfer. This bit is never set for ISO DMA transfer. The core automatically sets this bit when a transmit DMA channel has completed the programmed transfer by servicing the last IN transaction from the USB host. This is when TXDMAn.TXn_TSC (transfer size counter) equals 0, and the last IN transaction completes with an ACK. When this bit is asserted, the USB device controller must read the DMAN_STAT register to identify the endpoint number for which the transfer completed. The endpoint interrupt IRQ_SRC.EPn_TX is never set for the assigned endpoint to TX DMA channel n.
		0	No action.
		1	Non-ISO transmit DMA transfer for a channel has ended.
			The value after system reset or USB reset is low.
9	RXn_CNT		The receive DMA channel n transactions count interrupt flag (non-ISO) bit is only for non-ISO DMA transfer. This bit is never set for ISO DMA transfer and never set if the interrupt enable bit associated with it is not set. It means that no poll operation is possible on the DMA. The core automatically sets this bit during an active Receive DMA transfer each time RXDMAn.RXn_TC equals 0 after an OUT transaction with ACK status. This bit is set after the RX DMA data have been read (end of DMA request). When this bit is asserted, the USB device controller must read DMAN_STAT register to identify the endpoint number for which the transfer completed. An RXn_CNT IT is asserted also if RXDMAn.RXn_STOP is set; in this case, both IRQ_SRC.RXn_EOT and IRQ_SRC.RXn_CNT are asserted.
		0	No action.
		1	Non-ISO receive DMA transfer for a channel has reached transactions count level.
			The value after system reset or USB reset is low.

Table 32-44. Interrupt Source Register (IRQ_SRC) Field Descriptions (continued)

Bit	Field	Value	Description
8	RXn_EOT		The receive DMA channel n end of transfer interrupt flag (non-ISO) bit is for non-ISO DMA transfer only. This bit is never set for ISO DMA transfer and never set if the interrupt enable bit associated with it is not set. It means that no poll operation is possible on the DMA. THE core automatically sets this bit when a receive DMA channel detects an end-of-transfer (EOT) packet during the last OUT transaction from the USB host. This bit is set after the RX DMA data have been read (end of DMA request). When this happens, the DMA assigned endpoint FIFO is kept disabled (STAT_FLG.FIFO_EN = 0) to avoid receiving a new packet data from the USB host. The USB device controller can grant DMA transfer again to the same endpoint by enabling the FIFO again (STAT_FLG.FIFO_EN = 1).An end of transfer is detected when the core receives a data packet that is less than the configured endpoint FIFO size (or is empty), or when RXDMAn.RXn_TC equals 0 after an OUT transaction with ACK status and the RXDMAn.RXn_STOP bit is set. When this bit is asserted, the USB device controller must read DMAN_STAT.DMAn_RX_IT_SRC to identify the endpoint number for which the transfer completed, and DMAN_STAT.DMAn_RX_SB must be informed of an odd number of bytes received during the last transaction (useful for 16-bit read access from the DATA_DMA register).The endpoint interrupt IRQ_SRC.EPn_RX is never set to RX DMA channel for the assigned endpoint.
		0	No action.
		1	Non-ISO receive DMA transfer for a channel has ended.
			The value after system reset or USB reset is low.
7	SOF		Start of frame interrupt flag bit: every millisecond, the USB host outputs a start-of-frame packet to the functions. The SOF bit reflects when a new SOF is received. Writing a 1 in the SOF bit location clears the flag. Writing a 0 has no effect. In accordance with the USB1.1 specification, if SOF is received corrupted or is not received, the core still sets this flag at the same rate (if bit SOF.FT_LOCK = 1) or after 12043 USB bit time (if bit SOF.FT_LOCK = 0).
		0	No action
		1	Start-of-frame packet received (or internal SOF)
			The value after system reset or USB reset is low.
6	Reserved	0	Reserved
5	EPn_RX		The EPn OUT transactions interrupt flag bit only concerns non-ISO endpoints. The core automatically sets the EPn_RX bit when a handshake sequence occurs for an OUT transaction to an interrupt or bulk endpoint (NAK with SYSCON1.NAK_EN set, ACK or STALL). The USB device controller must read the EPn_STAT register to identify the endpoint causing the interrupt.
		0	No action.
		1	OUT transaction detected on an endpoint.
			The value after system reset or USB reset is low.
4	EPn_TX		The EPn IN transactions interrupt flag bit only concerns non-ISO endpoints. The core automatically sets this bit when a handshake sequence occurs for an IN transaction to an interrupt or bulk endpoint (NAK with SYSCON1.NAK_EN set, ACK or STALL). The USB device controller must read the EPn_STAT register to identify the endpoint causing the interrupt.
		0	No action
		1	IN transaction detected on an endpoint value after system reset or USB reset is low.
3	DS_CHG		Device state changed interrupt flag bit: the core automatically resets the DS_CHG bit when the state of the device changes. This is when the core modifies any of the bits present in the DEVSTAT register. When this bit is cleared, the background DEVSTAT register moves into foreground position.
		0	No action.
		1	Device state change detected value after USB device controller hardware reset is low and after USB reset is high.
2	SETUP		Setup transaction interrupt flag bit: the core automatically sets the SETUP bit when a valid setup transaction completes on control endpoint for a non-autodecoded control request and automatically clears it when the USB device controller sets EP_NUM.SETUP_SEL bit when reading setup data. A write 1 to it has no effect.
		0	No action.
		1	Valid setup transaction occurred on endpoint 0.
			Value after system reset or USB reset is low.

Table 32-44. Interrupt Source Register (IRQ_SRC) Field Descriptions (continued)

Bit	Field	Value	Description
1	EP0_RX		EP0 OUT transactions interrupt flag bit: the core automatically sets the EP0_RX bit when a handshake sequence occurs for a non-autodecoded OUT transaction to control endpoint (NAK with SYSCON1.NAK_EN set, ACK or STALL).
		0	No action.
		1	OUT transaction on EP0.
			Value after system reset or USB reset is low.
0	EP0_TX		EP0 IN transactions interrupt flag bit: the core automatically sets this bit when a handshake sequence occurs for a non-autodecoded IN transaction to control endpoint (NAK with SYSCON1.NAK_EN set, ACK or STALL).
		0	No action.
		1	IN transaction on EP0.
			Value after system reset or USB reset is low.

32.3.1.14 Non-ISO Endpoint Interrupt Status Register (EPN_STAT)

This read-only register identifies the non-ISO endpoint causing an EPn interrupt. A write into it is denied.

NOTE: If a nontransparent transaction occurs before a previous one on another endpoint in the same direction that has been handled, the second interrupt is asserted only after the USB device controller clears the first one and EPN_STAT is updated with the corresponding interrupt assertion.

Figure 32-41. Non-ISO Endpoint Interrupt Status Register (EPN_STAT) [address = FCF78A1Ah]

15 — 12	11 — 8
Reserved	EPn_RX_IT_SRC
R-0	R-0

7 — 4	3 — 0
Reserved	EPn_TX_IT_SRC
R-0	R-0

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

Table 32-45. Non-ISO Endpoint Interrupt Status Register (EPN_STAT) Field Descriptions

Bit	Field	Value	Description
15-12	Reserved	0	Reserved
11-8	EPn_RX_IT_SRC		The receive endpoint interrupt source (non-ISO) bit only concerns non-ISO endpoints. When the IRQ_SRC.EPn_RX flag is set, the endpoint causing the interrupt condition is encoded in these four register bits. When the IRQ_SRC.EPn_RX flag is cleared, the four bits read as 0.
		0	No receive endpoint interrupt is pending.
		1h	EP1
		:	:
		Fh	EP15
			Values after system reset or USB reset are low (all four bits).
7-4	Reserved	0	Reserved
3-0	EPn_TX_IT_SRC		Transmit endpoint interrupt source (non-ISO) bit only concerns non-ISO endpoints. When the IRQ_SRC.EPn_TX flag is set, the endpoint that causes this flag to be set is encoded in these four register bits. When the IRQ_SRC.EPn_TX flag is cleared, the four bits read as 0.
		0	No transmit endpoint interrupt is pending.
		1h	EP1
		:	:
		Fh	EP15
			Values after system reset or USB reset are low (all 4 bits).

32.3.1.15 Non-ISO DMA Interrupt Status Register (DMAN_STAT)

This read-only register identifies the endpoint that causes a DMAn interrupt. A write into it is denied.

NOTE: If a DMA interrupt occurs before a previous one on another endpoint in the same direction that has been handled, the second interrupt is asserted only after the USB device controller clears the first one. DMAn_STAT is updated when the corresponding interrupt is asserted.

Figure 32-42. Non-ISO DMA Interrupt Status Register (DMAN_STAT) [address = FCF78A1Ch]

15 - 13	12	11 - 8
Reserved	DMAn_RX_SB	DMAn_RX_IT_SRC
R-0	R-0	R-0

7 - 4	3 - 0
Reserved	DMAn_TX_IT_SRC
R-0	R-0

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

Table 32-46. Non-ISO DMA Interrupt Status Register (DMAN_STAT) Field Descriptions

Bit	Field	Value	Description
15-13	Reserved	0	Reserved
12	DMAn_RX_SB		The DMA receive single byte (non-ISO) bit only concerns non-ISO endpoints (ISO endpoints receive a constant number of bytes). This bit is set when an IRQ_SRC.RXn_EOT interrupt is asserted and the core received an odd number of bytes during the last transaction. It is used to know the exact number of bytes received in case of a 16-bit read access from DATA_DMA register. When the IRQ_SRC.RXn_EOT flag is cleared, this bit reads as 0.
		0	No EOT DMA interrupt is pending or the core received an even number of bytes during the last transaction.
		1	An EOT DMA interrupt is pending and an odd number of bytes was received during the last transaction.
			Value after system reset or USB reset is low.
11-8	DMAn_RX_IT_SRC		The DMA receive interrupt source (non-ISO) bit only concerns non-ISO endpoints. When the IRQ_SRC.EPn_RX flag is set, the endpoint causing this flag to be set is encoded in these four register bits. When the IRQ_SRC.EPn_RX flag is cleared, the four bits read as 0.
		0	No receive DMA interrupt is pending.
		1h	EP1
		:	:
		Fh	EP15
			Values after system reset or USB reset are low (all 4 bits).
7-4	Reserved	0	Reserved
3-0	DMAn_TX_IT_SRC		The DMA transmit interrupt source (non-ISO) bit only concerns non-ISO endpoints. When the IRQ_SRC.EPn_TX flag is set, the endpoint that causes this flag to be set is encoded in these four register bits. When the IRQ_SRC.EPn_TX flag is cleared, the four bits read as 0.
		0	No transmit DMA interrupt is pending.
		1h	EP1
		:	:
		Fh	EP15
			Values after system reset or USB reset are low (all 4 bits).

32.3.1.16 DMA Receive Channels Configuration Register (RXDMA_CFG)

This read/write register enables the three possible DMA receive channels and selects the endpoint number that is assigned to each of these DMA channels. An endpoint used by an RX DMA channel must have been configured (through register EPn_RX). The RXDMA_CFG register can be filled when SYSCON1.CFG_LOCK is set.

Only one channel is serviced at a time, so it is better to configure ISO endpoints on the first channel (_EP). If the ISO endpoint is configured on channel 2, it can never be serviced with low software and a fast USB host.

The USB device controller transmit endpoint DMA channels 0, 1, and 2 are associated with device controller inputs.

NOTE: System software must not program RXDMAx_EP to endpoint numbers that are not configured as DMA endpoints.

Figure 32-43. DMA Receive Channels Configuration Register (RXDMA_CFG) [address = FCF78A20h]

15 13	12	11 8
Reserved	RX_REQ	RXDMA2_EP
R-0	R/W-0	R/W-0

7 4	3 0
RXDMA1_EP	RXDMA0_EP
R/W-0	R/W-0

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

Table 32-47. DMA Receive Channels Configuration Register (RXDMA_CFG) Field Descriptions

Bit	Field	Value	Description
15-13	Reserved	0	Reserved
12	RX_REQ		The RX DMA request active level or pulse bit allows the RXDMA request to be configurable level or pulse-sensitive. When pulse-sensitive, the request is active during 2 cycles.
		0	RX DMA request active level.
		1	RX DMA request active pulse.
			Value after system reset or USB reset is low.
11-8	RXDMA2_EP		Receive endpoint number for DMA channel 2. The endpoint number binary encoded in these four bits is the current receive endpoint selected for DMA channel 2. A zero value indicates that the DMA channel 2 is deactivated. Any other value automatically enables receive DMA transfer for the selected endpoint.
		0	Receive DMA channel 2 is deactivated.
		1h	EP1
		:	:
		Fh	EP15
			Values after system reset or USB reset are low (all 4 bits).
7-4	RXDMA1_EP		Receive endpoint number for DMA channel 1. The endpoint number binary encoded in these four bits is the current receive endpoint selected for DMA channel 1. A zero value indicates that the DMA channel 1 is deactivated. Any other value automatically enables receive DMA transfer for the selected endpoint.
		0	Receive DMA channel 1 is deactivated.
		1h	EP1
		:	:
		Fh	EP15
			Values after system reset or USB reset are low (all 4 bits).

Table 32-47. DMA Receive Channels Configuration Register (RXDMA_CFG) Field Descriptions (continued)

Bit	Field	Value	Description
3-0	RXDMA0_EP		Receive endpoint number for DMA channel 0. The endpoint number binary encoded in these four bits is the current receive endpoint selected for DMA channel 0. A zero value indicates that the DMA channel 0 is deactivated. Any other value automatically enables receive DMA transfer for the selected endpoint.
		0	Receive DMA channel 0 is deactivated.
		1h	EP1
		:	:
		Fh	EP15
			Values after system reset or USB reset are low (all 4 bits).

32.3.1.17 DMA Transmit Channels Configuration Register (TXDMA_CFG)

This read/write register enables the three possible DMA transmit channels and selects the endpoint number that is assigned to each. An endpoint used by a TX DMA channel must have been configured (through register EPn_TX). TXDMA_CFG register can be filled when SYSCON1.CFG_LOCK is set.

Only one channel is serviced at a time, so it is better to configure ISO endpoints on the first channel (TXDMA0_EP). If, the/your ISO endpoint is configured on the channel 2 and it can never be serviced with low software and a fast USB host.

USB device controller transmit endpoint DMA channels 0, 1, and 2 are associated with device controller inputs.

Figure 32-44. DMA Transmit Channels Configuration Register (TXDMA_CFG) [address = FCF78A22h]

15 13	12	11 8
Reserved	TX_REQ	TXDMA2_EP
R-0	R/W-0	R/W-0

7 4	3 0
TXDMA1_EP	TXDMA0_EP
R/W-0	R/W-0

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

Table 32-48. DMA Transmit Channels Configuration Register (TXDMA_CFG) Field Descriptions

Bit	Field	Value	Description
15-13	Reserved	0	Reserved
12	TX_REQ		The TX DMA request active level or pulse bit allows the TXDMA request to be configurable level or pulse-sensitive. When pulse-sensitive, the request is active for two cycles.
		0	TX DMA request active level.
		1	TX DMA request active pulse value after system reset or USB reset is low.
11-8	TXDMA2_EP		Transmit endpoint number for DMA channel 2: the endpoint number binary encoded in these four bits is the current transmit endpoint selected for DMA channel 2. A zero value indicates that the DMA channel 2 is deactivated. Any other value automatically enables transmit DMA transfer for the selected endpoint.
		0	Transmit DMA channel 2 is deactivated.
		1h	EP1
		:	:
		Fh	EP15
			Values after system reset or USB reset are low (all four bits).

Table 32-48. DMA Transmit Channels Configuration Register (TXDMA_CFG) Field Descriptions (continued)

Bit	Field	Value	Description
7-4	TXDMA1_EP		Transmit endpoint number for DMA channel 1: the endpoint number binary encoded in these four bits is the current transmit endpoint selected for DMA channel 1. A zero value indicates that the DMA channel 1 is deactivated. Any other value automatically enables transmit DMA transfer for the selected endpoint.
		0	Transmit DMA channel 1 is deactivated.
		1h	EP1
		:	:
		Fh	EP15
			Values after system reset or USB reset are low (all four bits).
3-0	TXDMA0_EP		Transmit endpoint number for DMA channel 0: the endpoint number binary encoded in these four bits is the current transmit endpoint selected for DMA channel 0. A zero value indicates that the DMA channel 0 is deactivated. Any other value automatically enables transmit DMA transfer for the selected endpoint.
		0	Transmit DMA channel 0 is deactivated.
		1h	EP1
		:	:
		Fh	EP15
			Values after system reset or USB reset are low (all four bits).

32.3.1.18 DMA FIFO Data Register (DATA_DMA)

This register is the entry point to write or to read data into/from an endpoint used in a DMA transfer through DMA channel 0, 1 or 2.

Figure 32-45. DMA FIFO Data Register (DATA_DMA) [address = FCF78A24h]

15	8
DATA_DMA	
R/W-0	

7	0
DATA_DMA	
R/W-0	

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

Table 32-49. DMA FIFO Data Register (DATA_DMA) Field Descriptions

Bit	Field	Description
15-0	DATA_DMA	DMA FIFO data. When an RX DMA request is active for a channel (only one active at a given time), this register contains the data that the core received from the USB host OUT transaction using this channel. Data can be accessed by the main DMA controller engine (read access) in response to the DMA request for the channel. When a TX DMA request is active for a channel (only one active at a given time), this register contains the data written by the main DMA controller engine (write access) in response to a DMA request for the transmit channel to be sent to the USB host during the next IN transaction. Note: It is possible for both an RX DMA request and a TX DMA request to be active at the same time. In this case, the main DMA controller engine can access both transmit endpoint and receive endpoint FIFO. A read access to DATA_DMA register affects the endpoint that caused the RX DMA request to be active, and a write access affects the endpoint that caused the TX DMA request to be active. Note: The USB device controller must not access this register directly; however, there is no hardware mechanism to prevent such access. No access into this register must be made out of DMA request handling.

32.3.1.19 Transmit DMA Control Register n (TXDMAn)

This read/write register controls the operation of the transmit DMA channel n (n = 0, 1, 2).

**Figure 32-46. Transmit DMA Control Register n (TXDMAn)
[address = FCF78A28h to FCF78A2Ch]**

15	14	13	10	9	8
TXn_EOT	TXn_START	Reserved		TXn_TSC	
R/W-0	R/W-0	R-0		R/W-0	

7	0
TXn_TSC	
R/W-0	

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

Table 32-50. Transmit DMA Control Register n (TXDMAn) Field Descriptions

Bit	Field	Value	Description
15	TXn_EOT		Transmit DMA channel n end of transfer: the TXn_EOT bit can be either 0 or 1 for BULK DMA transfer. When the USB device controller sets it to 1, this bit signals the core that the transfer size set in TXDMAn.TXn_TSC is in bytes. A TX one interrupt (IRQ_SRC.TXn_DONE) is asserted with the last IN transaction. If the number of bytes set in TXDMAn.TXn_TSC is a multiple of the endpoint buffer size, the TX done interrupt is asserted only after an IN transaction with an empty data packet. When cleared, the transfer size set in TXn_TSC is in full buffer size for the endpoint selected (BULK only). A TX done interrupt is asserted when the last buffer is sent with the last IN transaction. This mode is to be used for a partial bulk transfer of a large file exceeding 1023 bytes.
		0	DMA transfer size is in buffers.
		1	DMA transfer size is in bytes.
			Value after system reset or USB reset is low.
14	TXn_START		Transmit DMA channel n start. The USB device controller sets this bit to tell the device that the main DMA system is ready to transmit the number of bytes or buffers. Once set, the DMA transfer cannot be interrupted, unless the USB device controller clears the endpoint in the TXDMA_CFG register. A write 0 into this bit has no effect and a read to this bit always returns 0. The IRQ_SRC.TXn_DONE interrupt bit is asserted when the DMA transfer ends.
		0	No action.
		1	DMA transfer start.
			Always reads 0.
13-10	Reserved	0	Reserved
9-0	TXn_TSC	0-3FFh	Transmit DMA channel n transfer size counter. The binary encoded value from 0 to 1023, which the USB device controller writes into this register, corresponds to the number of bytes or number of buffer transfers (function of TXDMAn.TXn_EOT) to be transmitted by the transmit DMA channel n. When read, the register reflects the number of bytes/buffers the USB device has still to transmit. Read mode is only provided for software debug purposes. Note: For ISO transfer, the user must verify that the set value does not exceed the ISO FIFO size for the endpoint. There is no hardware mechanism to prevent this situation. If this situation occurs, results are unpredictable. Note: For bulk transfer, when TXDMAn.TXn_EOT = 0, a set value of TXDMAn.TXn_TSC = 0 means 1024 buffers and not 0. The counter then operates in the following way: 000, 3FF, 3FE, 0001, 000, stop. When TXDMAn.TXn_EOT = 1, a set value of TXDMAn.TXn_TSC = 0 a NULL packet is sent in response to the next IN token. Values after system reset or USB reset are low (all 10 bits).

32.3.1.20 Receive DMA Control Register n (RXDMAn)

This read/write register monitors incoming OUT transactions during DMA transfer on channel n (n=0,1,2).

Figure 32-47. Receive DMA Control Register n (RXDMAn) [address = FCF78A30h to FCF78A34h]

15	14	8
RXn_STOP	Reserved	
R/W-0	R-0	

7	0
RXn_TC	
R/W-0	

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

Table 32-51. Receive DMA Control Register n (RXDMAn) Field Descriptions

Bit	Field	Value	Description
15	RXn_STOP	0-1	Receive DMA channel n transfer stop. When this bit is set, an RXn_EOT interrupt is asserted to the USB device controller after n OUT transactions where n is the encoded binary value + 1 programmed into RXDMAn.RXn_TC field. This register is used when no smaller than buffer size packet is received at an end-of-transfer (EOT), and the USB device controller expects a given amount of data for the transfer. Note: At end of transfer, the DMA channel is disabled and all OUT transactions received to the assigned endpoint are sent NAK by the core. The USB device controller must set CTRL.SET_FIFO_EN for the endpoint to reenable the channel. Values after system reset or USB reset are low.
14-8	Reserved	0	Reserved
7-0	RXn_TC	0-FFh	Receive DMA channel n transactions count. The USB device controller can ask for an interrupt each n OUT transactions where n is the encoded binary value + 1 programmed into RXDMAn.RXn_TC field. This register must be programmed to the desired transaction watermark limit prior to enabling the DMA transfer for the receive DMA channel n. Note: A reached watermark does not disable an active DMA transfer if RXDMAn.RXn_STOP was not set. If RXDMAn.RXn_STOP was set for the transfer both RXn_CNT and RXn_EOT interrupts are asserted. A read to that register returns the number of transactions remaining before the IRQ_SRC.RXn_CNT interrupt flag is asserted. This read mode is only provided for software debug purposes. Values after system reset or USB reset are low (all 8 bits).

32.3.1.21 Endpoint 0 Configuration Register (EP0)

This read/write register gives the device configuration for control endpoint 0.

Values depend on whether the reset action comes from USB device controller (CPU) or the USB host.

Figure 32-48. Endpoint 0 Configuration Register (EP0) [address = FCF78A40h]

15 14	13 12	11	10 8
Reserved	EP0_SIZE	Reserved	EP0_PTR
R-0	R/W-0	R-0	R/W-0

7 0
EP0_PTR
R/W-0

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

Table 32-52. Endpoint 0 Configuration Register (EP0) Field Descriptions

Bit	Field	Value	Description
15-14	Reserved	0	Reserved
13-12	EP0_SIZE		Endpoint 0 FIFO size. This field contains the endpoint 0 FIFO size value and must match the value sent by the USB device controller to the USB host during the GET_DEVICE_DESCRIPTOR request preceding configuration phase. Status flags (STAT_FLG.NON_ISO_FIFO_EMPTY and STAT_FLG.NON_ISO_FIFO_FULL) and overrun and underrun conditions are based on this value for all IN and OUT transactions to endpoint 0.The USB device controller must fill this field before setting SYSCON1.CFG_LOCK.
		0	8 bytes
		1	16 bytes
		2h	32 bytes
		3h	64 bytes
			Values after USB device controller hardware reset or USB reset are unchanged (which means that value is unknown until first write access).
11	Reserved	0	Reserved
10-0	EP0_PTR		Endpoint 0 pointer. This field contains the address of the endpoint 0 pointer. Value 0 is not permitted (reserved for setup FIFO). Note: The pointer value must be set to a value < FFh, because the memory size is 2K bytes and a pointer coded value = FFh corresponds to 2040 bytes. Addressing upper bytes results in memory overlap.
		0	Address = BASE (not permitted)
		1h	Address = BASE + 8 bytes
		2h	Address = BASE + 16 bytes
		3h	Address = BASE + 24 bytes
		:	:
		FFh	Address = BASE + 2040 bytes
			Values after USB device controller hardware reset or USB reset are unchanged (which means that values are unknown until first write access).

32.3.1.22 Receive Endpoint n Configuration Register (EPn_RX)

This read/write register gives the device configuration for non-control receive endpoint n (1:15). The endpoints size fields must match values sent by the USB device controller to the USB host in response to the GET_CONFIGURATION_DESCRIPTOR during configuration phase.

The USB device controller must fill this field before setting SYSCON1.CFG_LOCK and must not change the values once SYSCON1.CFG_LOCK is set.

Values depend on whether the reset action comes from USB device controller (CPU) or USB host.

Figure 32-49. Receive Endpoint n Configuration Register (EPn_RX) [address = FCF78A42h to FCF78A5Eh]

15	14	13	12	11	10	8
EPn_RX_VALID	EPn_RX_SIZE_DB	EPn_RX_SIZE		EPn_RX_ISO	EPn_RX_PTR	
R/W-0	R/W-0	R/W-0		R/W-0	R/W-0	

7	0
EPn_RX_PTR	
R/W-0	

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

Table 32-53. Receive Endpoint n Configuration Register (EPn_RX) Field Descriptions

Bit	Field	Value	Description
15	EPn_RX_VALID		The receive endpoint n valid bit must be set by the USB device controller to allow receive endpoint n to be used for USB transfers as part of the device configuration. If not set, all transactions to this endpoint are ignored by the core.
		0	Receive endpoint n does not exist for this configuration.
		1	Receive endpoint n is part of the device configuration.
			Values after USB device controller hardware reset are low, after USB reset is unchanged.
14	EPn_RX_SIZE_DB		Receive non-ISO (or ISO) endpoint n double-buffer (DB). This bit is only for non-ISO endpoints. For ISO endpoints, which are always double-buffered, this bit is endpoint size MSB. This bit must be set by the USB device controller to allow double buffering for receive non-ISO endpoint n. This is used to reduce number of transactions resulting in NAK handshake.
		0	No double buffer for non-ISO receive endpoint n.
		1	Double buffer used for non-ISO receive endpoint n.
			Values after USB device controller hardware reset or USB reset are unchanged (which means that values are unknown until first write access).

Table 32-53. Receive Endpoint n Configuration Register (EPn_RX) Field Descriptions (continued)

Bit	Field	Value	Description
13-12	EPn_RX_SIZE		The receive endpoint n size field contains the endpoint n FIFO size value. Status flags (STAT_FLG.NON_ISO_FIFO_EMPTY, STAT_FLG.NON_ISO_FIFO_FULL, STAT_FLG. ISO_FIFO_EMPTY, and STAT_FLG.ISO_FIFO_FULL) and overrun and underrun conditions are based on this value for all OUT transactions to endpoint n.
			Non-ISO (bits 13-12):
		0	8 bytes
		1h	16 bytes
		2h	32 bytes
		3h	64 bytes
			ISO (bits 14-12):
		0	8 bytes
		1h	16 bytes
		2h	32 bytes
		3h	64 bytes
		4h	128 bytes
		5h	256 bytes
		6h	512 bytes
		7h	1023 bytes
			Note: For ISO endpoints, a size of 1023 bytes takes up the whole memory and prevents programming with a 2K-byte USB device controller.
			Values after USB device controller hardware reset or USB reset are unchanged (which means that values are unknown until first write access).
11	EPn_RX_ISO		The receive ISO endpoint n field must be set if the receive endpoint n type is isochronous in the desired device configuration. If not set, the endpoint type is bulk or interrupt (the hardware does not distinguish bulk type from interrupt).
		0	Receive endpoint n type is bulk or interrupt.
		1	Receive endpoint n type is isochronous.
			Values after USB device controller hardware reset or USB reset are unchanged.
10-0	EPn_RX_PTR		The receive endpoint n pointer field contains the address of the receive endpoint n pointer. Value 0x000 is not permitted (reserved for setup FIFO).
			Note: For ISO endpoints or for non-ISO endpoints that allow double-buffering, 2 × RX buffer size must be reserved for ping-pong.
		0	Address = BASE (not permitted)
		1h	Address = BASE + 8 bytes
		2h	Address = BASE + 16 bytes
		3h	Address = BASE + 24 bytes
		:	:
		FFh	Address = BASE + 2040 bytes
			Values after USB device controller hardware reset or USB reset are unchanged (which means that values are unknown until first write access).
			Note: Pointer value must be set to a value < FFh, because the memory size is 2K bytes and a pointer coded value = FFh corresponds to 2040 bytes. Addressing upper bytes results in memory overlap.

32.3.1.23 Transmit Endpoint n Configuration Register (EPn_TX)

This read/write register gives the device configuration for non-control transmit endpoint n (1:15). The endpoints size fields must match the values sent by the USB device controller to the USB host in response to the GET_CONFIGURATION_DESCRIPTOR during configuration phase.

NOTE: The USB device controller must fill this field before setting SYSCON1.CFG_LOCK and must not change the values once SYSCON1.CFG_LOCK is set.

Figure 32-50. Transmit Endpoint n Configuration Register (EPn_TX) [address = FCF78A62h to FCF78A7Eh]

15	14	13	12	11	10	8
EPn_TX_ VALID	EPn_TX_SIZE_ DB	EPn_TX_SIZE		EPn_TX_ISO	EPn_TX_PTR	
R/W-0	R/W-0	R/W-0		R/W-0	R/W-0	

7	0
EPn_TX_PTR	
R/W-0	

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

Table 32-54. Transmit Endpoint n Configuration Register (EPn_TX) Field Descriptions

Bit	Field	Value	Description
15	EPn_TX_VALID		The transmit endpoint n valid bit must be set by the USB device controller to allow transmit endpoint n to be used for USB transfers as part of the device configuration. If not set, all transactions to this endpoint are ignored by the core.
		0	Transmit endpoint n does not exist for this configuration.
		1	Transmit endpoint n is part of the device configuration.
			Values after USB device controller hardware reset are low, after USB reset is unchanged.
14	EPn_TX_SIZE_DB		Transmit non-ISO (or ISO) endpoint n double buffer. This bit is only for non-ISO endpoints used in DMA mode. For ISO endpoints, which are always double buffered, this bit is endpoint size MSB. For non-ISO endpoints that are not used in a DMA transfer, double buffering is not provided. This bit must be set by the USB device controller to allow double buffering for transmit non-ISO endpoint n, when used in a DMA transfer. This is used to reduce the number of transactions resulting in NAK handshake.
		0	No double buffer for non-ISO transmit endpoint n.
		1	Double buffer used for non-ISO transmit endpoint n.
			Values after system reset or USB reset is unchanged. Or transmit ISO endpoint n size[2]

Table 32-54. Transmit Endpoint n Configuration Register (EPn_TX) Field Descriptions (continued)

Bit	Field	Value	Description
13-12	EPn_TX_SIZE		The transmit endpoint n size field contains the endpoint n FIFO size value. Status flags (FIFO_EMPTY, FIFO_FULL) and underrun condition are based on this value for all IN transactions to endpoint n.
			Non-ISO (bits 13-12):
		0	8 bytes
		1h	16 bytes
		2h	32 bytes
		3h	64 bytes
			ISO (bits 14-12):
		0	8 bytes
		1h	16 bytes
		2h	32 bytes
		3h	64 bytes
		4h	128 bytes
		5h	256 bytes
		6h	512 bytes
		7h	1023 bytes
			Note: For ISO endpoints, a size of 1023 bytes takes up the whole memory and prevents programming with a 2K-byte USB device controller.
			Values after USB device controller hardware reset or USB reset are unchanged (which means that values are unknown until first write access).
11	EPn_TX_ISO		The transmit ISO endpoint n field must be set if the transmit endpoint n type is isochronous in the desired device configuration. If not set, the endpoint type is bulk or interrupt (the hardware does not distinguish bulk type from interrupt).
		0	Transmit endpoint n type is bulk or interrupt.
		1	Transmit endpoint n type is isochronous.
			Value after USB reset is unchanged.
10-0	EPn_TX_PTR		The transmit endpoint n pointer field contains the address of the transmit endpoint n pointer.
			For ISO endpoints or for non-ISO endpoints that allow double-buffering, (2 × TX buffer size) must be reserved for ping-pong.
		0	Address = BASE
		1h	Address = BASE + 8 bytes
		2h	Address = BASE + 16 bytes
		3h	Address = BASE + 24 bytes
		:	:
		FFh	Address = BASE + 2040 bytes
			Values after USB device controller hardware reset or USB reset are unchanged (which means that values are unknown until first write access).
			Pointer value must be set to a value < FFh, because the memory size is 2K bytes and a pointer coded value = FFh corresponds to 2040 bytes. Addressing upper bytes results in memory overlap.

32.3.2 USB Device Transactions

There is an interrupt to the CPU at the end of an USB transaction if the CPU has actions to perform. Isochronous transactions are an exception because isochronous interrupt information is available at start of frame interrupts. The CPU ISR code determines which endpoint and direction caused the interrupt and acts appropriately. The following subsections describe in detail the activities surrounding USB transactions that are not part of a DMA transfer. Cases where a transaction occurs before the previous one has been handled by the CPU are not taken into account in this part. The information is organized so that each section deals with one type and direction of transaction, such as:

- Non-isochronous, non-setup OUT transactions
- Non-isochronous IN transactions
- Isochronous OUT transactions
- Isochronous IN transactions

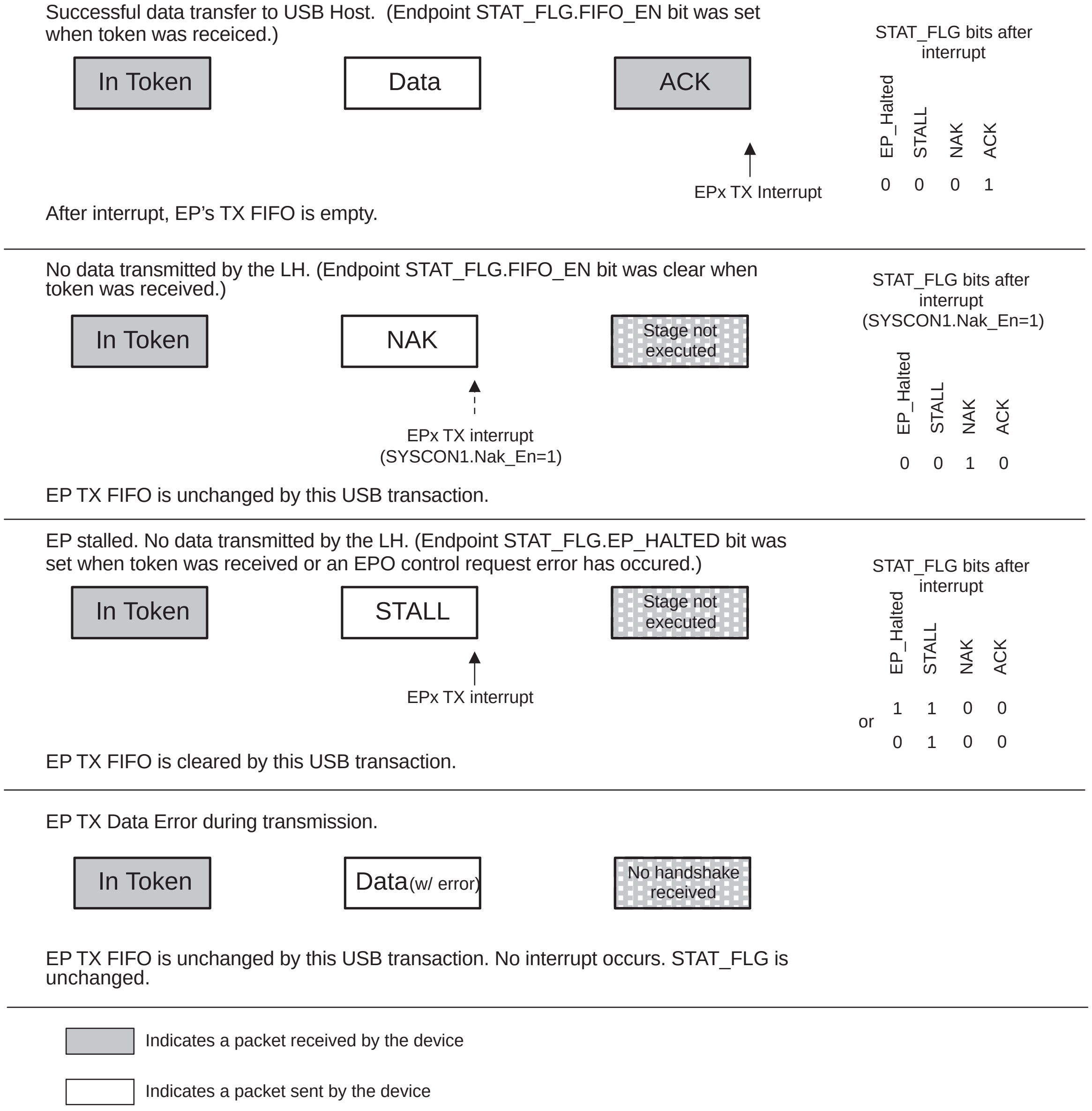
This allows each section to focus only on a specific style of transaction without adding in the confusion of special cases related to other styles.

32.3.3 Non-Isochronous, Non-Setup OUT (USB HOST→CPU) Transactions

Non-isochronous, non-setup OUT transactions refer to USB transactions where data is moved from the USB host to the CPU, the USB handshaking protocols are in effect, and data transmission is ensured. These types of transactions apply to all OUT transactions on bulk and interrupt endpoint types, and to non-setup transactions on control endpoints.

Figure 32-51 shows the various USB protocol conditions that can occur during non-isochronous, non-setup OUT transactions. The diagram shows the three phases that can occur in an OUT transaction, the direction of information flow for each phase, when endpoint interrupts are generated, and the resulting STAT_FLG bits for the endpoint. The top three cases show the normal USB handshaking modes: acknowledge (good data received), NAK (device not ready to receive data), and STALL (device in a condition where the endpoint cannot handle OUT transactions). The last case is an abnormal instance where the token packet or the data packet was received with errors. The RX FIFO only contains valid receive data under the first (acknowledge) case.

Figure 32-51. Non-Isochronous, Non-Control OUT Transaction Phases and Interrupts



Successful data transfer to USB Host. (Endpoint STAT_FLG.FIFO_EN bit was set when token was receiced.)

After interrupt, EP’s TX FIFO is empty.

No data transmitted by the LH. (Endpoint STAT_FLG.FIFO_EN bit was clear when token was received.)

EP TX FIFO is unchanged by this USB transaction.

EP stalled. No data transmitted by the LH. (Endpoint STAT_FLG.EP_HALTED bit was set when token was received or an EPO control request error has occured.)

EP TX FIFO is cleared by this USB transaction.

EP TX Data Error during transmission.

EP TX FIFO is unchanged by this USB transaction. No interrupt occurs. STAT_FLG is unchanged.

32.3.3.1 Non-Isochronous, Non-Control OUT Endpoint Handshaking Conditions

An endpoint CTRL.SET_FIFO_EN bit provides the main control for the endpoint ability to allow successful OUT transaction data reception for the endpoint. If at the beginning of an OUT transaction to an endpoint, the endpoint STAT_FLG.FIFO_EN bit is 1, the USB module is allowed to accept the OUT transaction data to the RX FIFO, and, when the transaction completes, the USB module can return ACK to the PC to indicate that the data was received correctly (this is the top case shown in Figure 32-51). If, however, the endpoint STAT_FLG.FIFO_EN bit was 0 at the beginning of an OUT transaction to the endpoint, the USB module returns NAK during the handshake phase to indicate that the endpoint did not accept the data (the second case shown in Figure 32-51).

The USB host need not send a whole RX FIFO worth of data to the endpoint during an OUT transaction. In this case, the RX FIFO is not full when the endpoint RX interrupt is generated. The CPU code must be careful not to read too much data. CPU code must read RXFSTAT.RXF_COUNT value before reading data from the RX FIFO.

At the end of an USB OUT transaction to an endpoint where the data is accepted (ACKed), the hardware clears the endpoint STAT_FLG.FIFO_EN bit. Once the CPU software has dealt with the OUT transaction data in the endpoint RX FIFO, it must re-enable the endpoint OUT transaction reception by setting the endpoint CTRL.SET_FIFO_EN bit. CPU software can use the endpoint CTRL.SET_FIFO_EN bit as a receive flow control mechanism.

32.3.3.1.1 Acknowledged Transactions (ACK)

At completion of an OUT transaction to an endpoint, the USB module issues an endpoint-specific interrupt to the CPU and STAT_FLG is updated. In response to the endpoint interrupt, the CPU must read EPN_STAT register to identify the endpoint causing the interrupt, then write a 1 to the interrupt bit to clear it. The CPU must then set EP_NUM.EP_NUM to the endpoint number and EP_NUM.EP_SEL to 1, and then read the endpoint status from STAT_FLG. STAT_FLG.ACK is set to indicate that the endpoint received a transaction to which the USB module signaled ACK handshaking.

If STAT_FLG.FIFO_EMPTY is cleared, the transaction sent 1 or more bytes of data (but less than or equal to the physical size of the endpoint RX FIFO), and the data is in the endpoint RX FIFO. The CPU knows the number of bytes to read from RX FIFO by reading RXFSTAT.RXF_COUNT value. The CPU can then read RX data from DATA register. Once the CPU has read the data from the FIFO, it sets the CTRL.SET_FIFO_EN bit to allow the next USB OUT transaction to the endpoint to be placed into the RX FIFO, and then clears the EP_NUM.EP_SEL bit. This clears the STAT_FLG.ACK bit for this endpoint to allow next transaction status to be written into the STAT_FLG register.

32.3.3.1.2 Non-Acknowledged Transactions (NAK)

The device can be configured via the SYSCON1.NAK_EN either to inform the CPU of a NAKed transaction or not. If SYSCON1.NAK_EN is cleared, no interrupt is asserted to the CPU if an OUT transaction completes with a NAK handshake and STAT_FLG.NAK bit is not set. If SYSCON1.NAK_EN is set, the USB module issues an endpoint-specific interrupt to the CPU at completion of an OUT transaction to an endpoint, and STAT_FLG.NAK bit is set. In response to the endpoint interrupt, the CPU must read EPN_STAT register to identify the endpoint causing the interrupt, and then write a 1 to the interrupt bit to clear it. The CPU must then set EP_NUM.EP_NUM to the endpoint number and EP_NUM.EP_SEL to 1, and then read the endpoint status from STAT_FLG. STAT_FLG.NAK is set to indicate that the endpoint received a transaction to which the USB module signaled NAK handshaking.

The CPU must set the CTRL.SET_FIFO_EN bit to allow the next USB OUT transaction to the endpoint to be placed into the RX FIFO, and then clear the EP_NUM.EP_SEL bit. This clears the STAT_FLG.NAK bit for this endpoint to allow the next transaction status to be written into the STAT_FLG register.

32.3.3.2 Non-Isochronous, Non-Control OUT Transaction Error Conditions

32.3.3.2.1 STALLed Transactions

The USB module responds to an endpoint OUT transaction with a STALL handshake to indicate an error condition on the endpoint either if the endpoint STAT_FLG.EP_HALTED bit is set or if a request error occurs (control transactions only). When an endpoint OUT transaction is given a STALL handshake, the endpoint STAT_FLG.STALL bit is set and an endpoint-specific interrupt is generated for the endpoint. STAT_FLG.FIFO_EN is of lower priority than STAT_FLG.EP_HALTED; when the STAT_FLG.EP_HALTED bit is set and transactions to the RX endpoint are stalled, regardless of the STAT_FLG.FIFO_EN value. If STAT_FLG.FIFO_EN is set, the STAT_FLG.FIFO_EN bit is automatically cleared at the end of the STALLed transaction and RX FIFO is cleared.

In response to the endpoint interrupt, the CPU must read the EPN_STAT register to identify the endpoint causing the interrupt, and then write a 1 to the interrupt bit to clear it. The CPU must then set EP_NUM.EP_NUM to the endpoint number and EP_NUM.EP_SEL to 1, and then read the endpoint status from STAT_FLG. STAT_FLG.STALL is set to indicate that the endpoint received a transaction to which the USB module signaled STALL handshaking.

If STAT_FLG.EP_HALTED has been set by the CPU and can be removed, the CPU must set CTRL.CLR_HALT to clear the condition and set CTRL.SET_FIFO_EN to allow the next USB OUT transaction to the endpoint to be placed into the RX FIFO. If STAT_FLG.EP_HALTED has been set in response to a SET_FEATURE request sent by the USB host, or if the bit is cleared (control transaction only), the CPU has no action to perform and must clear the EP_NUM.EP_SEL bit. This clears the STAT_FLG.STALL bit for this endpoint and allows the next transaction status to be written into the STAT_FLG register.

32.3.3.2.2 Packet Errors

In case of a receive data error during an endpoint OUT transaction (token or data packet), the USB module does not provide a handshake during the handshake phase of the transaction and no interrupt is asserted to the CPU (the fourth case shown in Figure 32-51). Additionally, the endpoint RX FIFO is not filled, and STAT_FLG.FIFO_EN bit is not cleared. If the CPU clears the RX FIFO during the data packet of an OUT transaction, no handshake is returned to the USB host to signal an error.

32.3.3.2.3 Sequence Bit Errors

If the core does not receive expected DATA PID during an OUT transaction, the module automatically returns ACK handshake to the USB host, regardless of the STAT_FLG.FIFO_EN bit (per the USB specification). Data is ignored, and no interrupt is asserted to the CPU.

This error occurs if ACK handshake from previous OUT transaction is received corrupted by the USB host.

32.3.3.3 Non-Isochronous, Non-Control OUT Endpoint FIFO Error Conditions

If the USB host attempts to fill more data into an endpoint RX FIFO than the FIFO can hold, a FIFO overrun occurs. The USB module does not provide a handshake during the handshake phase of the transaction and no interrupt is asserted to the CPU. Additionally, the endpoint RX FIFO is not filled, and STAT_FLG.FIFO_EN bit is not cleared.

The CPU must not read more data from RX FIFO than the value indicated by RXFSTAT.RXF_COUNT.

32.3.4 Non-Isochronous IN (CPU→USB HOST) Transactions

Non-isochronous IN transactions refer to USB transactions in which data is moved from the CPU to the USB host, where the USB handshaking protocols are in effect, and data transmission is ensured. These transactions are the IN transactions that occur on control, bulk, and interrupt endpoints. These transactions do not ensure USB bandwidth.

To provide data for an endpoint IN transaction, the CPU code writes the transmit data into the endpoint transmit FIFO. CPU code must first wait until the USB is done with any previous TX data for the endpoint (if data had previously been written to the TX FIFO). This must be done by proper response to endpoint-specific transmit interrupts. When an IN transaction to the endpoint occurs, if the endpoint STAT_FLG.FIFO_EN bit is set, the USB module sends any data that is in the endpoint TX FIFO during the data phase. If the TX FIFO is empty and STAT_FLG.FIFO_EN is set when an IN transaction to the endpoint occurs, a 0-byte data packet is sent.

Once the endpoint previous transmit activity is taken care of, the CPU code gains access to endpoint FIFO and status by setting the EP_NUM.EP_SEL bit. Then the CPU can write the new transmit data to the endpoint TX FIFO via the DATA register (being careful not to overflow the FIFO). Once all of the transmit data has been written to the endpoint FIFO, CPU code sets the CTRL.SET_FIFO_EN bit to allow the USB to use the endpoint TX FIFO, and then clears the EP_NUM.EP_SEL bit. The data in the endpoint TX FIFO is sent to the USB host the next time an IN transaction to the endpoint occurs.

Figure 32-52 shows the various USB protocol conditions that can occur during non-isochronous IN transactions. It diagrams the three phases of the IN transaction, the direction of information flow for each phase, when endpoint-specific interrupts are generated, and the resulting STAT_FLG bits for the endpoint. The top three cases show the normal USB handshaking: acknowledge (data sent by USB module and received properly by the USB host), NAK (device not ready to send data to USB host), and STALL (device in a condition where the endpoint cannot handle IN transactions). The last case shows an abnormal case in which there is an error either in the token packet received by the core or in the data packet received by the USB host.

32.3.4.1 Non-Isochronous IN Endpoint Handshaking

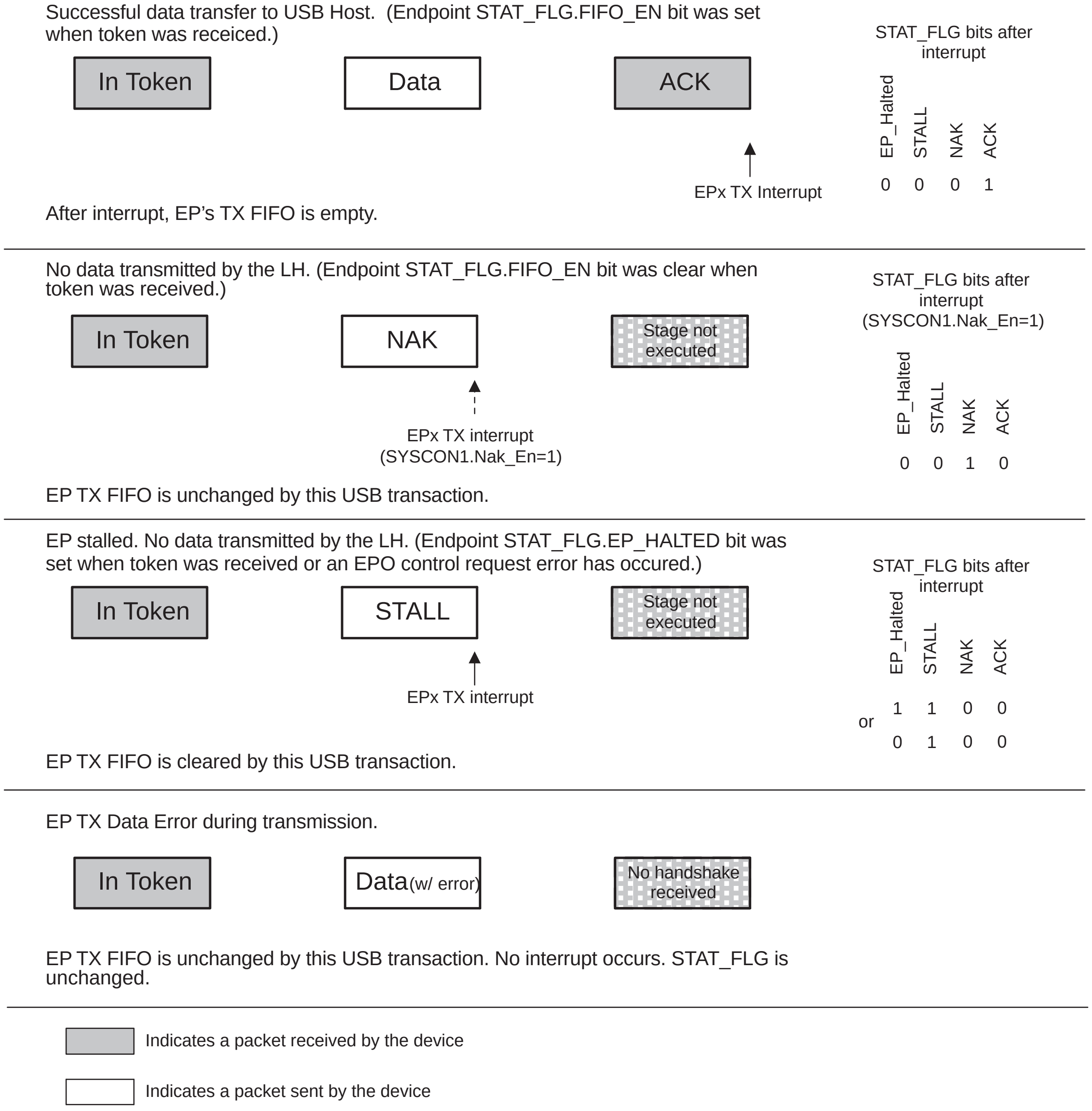
Per USB specifications for IN transactions, the USB host can provide only one of two handshakes to the USB function during the handshake phase: ACK or no handshake at all. The first indicates successful transfer (first case shown in Figure 32-52), and the second indicates that the host received a garbled data packet (last case shown in Figure 32-52).

32.3.4.1.1 Acknowledged Transactions (ACK)

When the endpoint IN transaction completes on the USB bus with an ACK handshake, the endpoint generates an endpoint-specific interrupt to the CPU (see first case in Figure 32-52). In response to the endpoint interrupt, the CPU must read the EPN_STAT register to identify the endpoint causing the interrupt, and then write a 1 to the interrupt bit to clear it. The CPU must then set EP_NUM.EP_NUM to the endpoint number, EP_NUM.EP_DIR to 1 (to signal an IN endpoint), and EP_NUM.EP_SEL to 1, and then read the endpoint status from STAT_FLG. STAT_FLG.ACK is set to indicate that the endpoint received an ACK handshake from the USB host, and the TX FIFO is empty (because any data that was in the TX FIFO was transmitted during the IN transaction).

If the CPU has more data to transmit to the USB host, it must fill the TX FIFO following the process indicated above. It must then clear EP_NUM.EP_SEL bit. This clears the STAT_FLG.ACK bit for this endpoint to allow next transaction status to be written into the STAT_FLG register.

Figure 32-52. Non-Isochronous IN Transaction Phases and Interrupts


Successful data transfer to USB Host. (Endpoint STAT_FLG.FIFO_EN bit was set when token was receiced.)
In Token
Data
ACK
EPx TX Interrupt
STAT_FLG bits after interrupt
EP_Halted
STALL
NAK
ACK
0 0 0 1
After interrupt, EP's TX FIFO is empty.
No data transmitted by the LH. (Endpoint STAT_FLG.FIFO_EN bit was clear when token was received.)
In Token
NAK
Stage not executed
EPx TX interrupt (SYSCON1.Nak_En=1)
STAT_FLG bits after interrupt (SYSCON1.Nak_En=1)
EP_Halted
STALL
NAK
ACK
0 0 1 0
EP TX FIFO is unchanged by this USB transaction.
EP stalled. No data transmitted by the LH. (Endpoint STAT_FLG.EP_HALTED bit was set when token was received or an EPO control request error has occured.)
In Token
STALL
Stage not executed
EPx TX interrupt
STAT_FLG bits after interrupt
EP_Halted
STALL
NAK
ACK
1 1 0 0
or
0 1 0 0
EP TX FIFO is cleared by this USB transaction.
EP TX Data Error during transmission.
In Token
Data(w/ error)
No handshake received
EP TX FIFO is unchanged by this USB transaction. No interrupt occurs. STAT_FLG is unchanged.
Indicates a packet received by the device
Indicates a packet sent by the device

32.3.4.1.2 Non-Acknowledged Transactions (NAK)

For the case in which the CPU is not ready to provide transmit data for transactions to an IN endpoint, the core provides a NAK handshake to the host for any USB IN transaction to that endpoint . Readiness to transmit data is signaled via the endpoint STAT_FLG.FIFO_EN bit; when 1 it indicates that data in the TX FIFO can be sent to the USB host. When the endpoint STAT_FLG.FIFO_EN bit is 0 and an IN transaction to the endpoint occurs, NAK handshake is sent, indicating that the CPU is not ready to handle the request.

If SYSCON1.NAK_EN bit is cleared, when the NAK handshake is sent in the data packet portion of the transaction to the IN endpoint, STAT_FLG is not updated and no endpoint-specific interrupt to the CPU is generated. If the SYSCON1.NAK_EN bit is set, when the NAK handshake is sent in the data packet portion of the transaction to the IN endpoint, the STAT_FLG.NAK bit is set and an endpoint-specific interrupt to the CPU is generated.

In response to the endpoint interrupt, the CPU must read the EPN_STAT register to identify the endpoint causing the interrupt, and then write a 1 to the interrupt bit to clear it. The CPU must then set EP_NUM.EP_NUM to the endpoint number, EP_NUM.EP_DIR to 1 (to signal an IN endpoint), and EP_NUM.EP_SEL to 1, and then read the endpoint status from STAT_FLG. STAT_FLG.NAK is set to indicate that the endpoint sent a NAK handshake to the USB host. If the CPU has data to transmit to the USB host, it must fill the TX FIFO following the process indicated above. The CPU must then clear the EP_NUM.EP_SEL bit. That clears the STAT_FLG.NAK bit for this endpoint to allow the next transaction status to be written into the STAT_FLG register. Signaling NAK does not cause the endpoint TX FIFO to be cleared (because the CPU still retains control of the FIFO).

Signaling NAK handshake for several endpoint transactions in a row can cause the PC host to discard the transaction, so NAK is not necessarily a good mechanism in cases where the CPU is not able to service a request for long periods of time.

32.3.4.2 Non-Isochronous IN Transaction Error Conditions

32.3.4.2.1 STALLed Transactions

The USB module sends a STALL handshake to the USB host during the data phase of the transaction to the IN endpoint either if the endpoint STAT_FLG.EP_HALTED flag is set, or if a request error occurs (control transaction only). A USB STALL handshake indicates that the device endpoint is in a condition in which it is not able to transfer data and instructs the USB host not to retry the transaction. The device typically requires intervention via some other mechanism to clear the condition, usually a control transfer via endpoint 0. The CPU can set the endpoint EP_HALTED bit by selecting the endpoint by writing the appropriate value in EP_NUM register, and then setting the endpoint CTRL.Set_HALT bit, and clear it by selecting the endpoint, and then setting the endpoint CTRL.Clr_HALT bit. When the endpoint EP_HALTED bit is set, the endpoint signals STALL for its IN transactions until the HALT condition is cleared. When the STALL handshake is sent in response to a transaction to the endpoint, the STAT_FLG.STALL bit is set, and an endpoint-specific interrupt to the CPU is generated.

In response to the endpoint interrupt, the CPU must read the EPN_STAT register to identify the endpoint causing the interrupt, and then write a 1 to the interrupt bit to clear it. The CPU must then set EP_NUM.EP_NUM to the endpoint number, EP_NUM.EP_DIR to 1 (to signal an IN endpoint), and EP_NUM.EP_SEL to 1, and then read the endpoint status from STAT_FLG. STAT_FLG.STALL is set to indicate that the endpoint sent a STALL handshake to the USB host. The CPU must then clear EP_NUM.EP_SEL bit. This clears the STAT_FLG.STALL bit for this endpoint and allows the next transaction status to be written into the STAT_FLG register.

Except for control endpoint 0, separate endpoint halt bits are defined for each direction; so for a given endpoint number, the TX can be halted when the RX is not.

32.3.4.2.2 Packet Errors

If an error (CRC, bit stuffing, or PID check) occurs during the token packet of a USB IN transaction to a non-isochronous endpoint, the USB block ignores the transaction. No endpoint-specific interrupt to the CPU occurs for transactions with corrupted packets. If the CPU clears the TX FIFO during the data packet of an IN transaction, a bit stuffing error is forced.

If the USB host returns no handshake after an IN transaction (in case of an error during transmission), the USB device controller module detects after a time-out that an error has occurred. The data to transmit is still in the TX FIFO to be re-sent during next IN transaction, STAT_FLG.FIFO_EN is not cleared, and no interrupt is asserted to the CPU.

32.3.4.3 Non-Isochronous IN Endpoint FIFO Error Conditions

The CPU cannot write more data into the TX FIFO than the configured FIFO size.

32.3.5 Isochronous OUT (USB HOST→CPU) Transactions

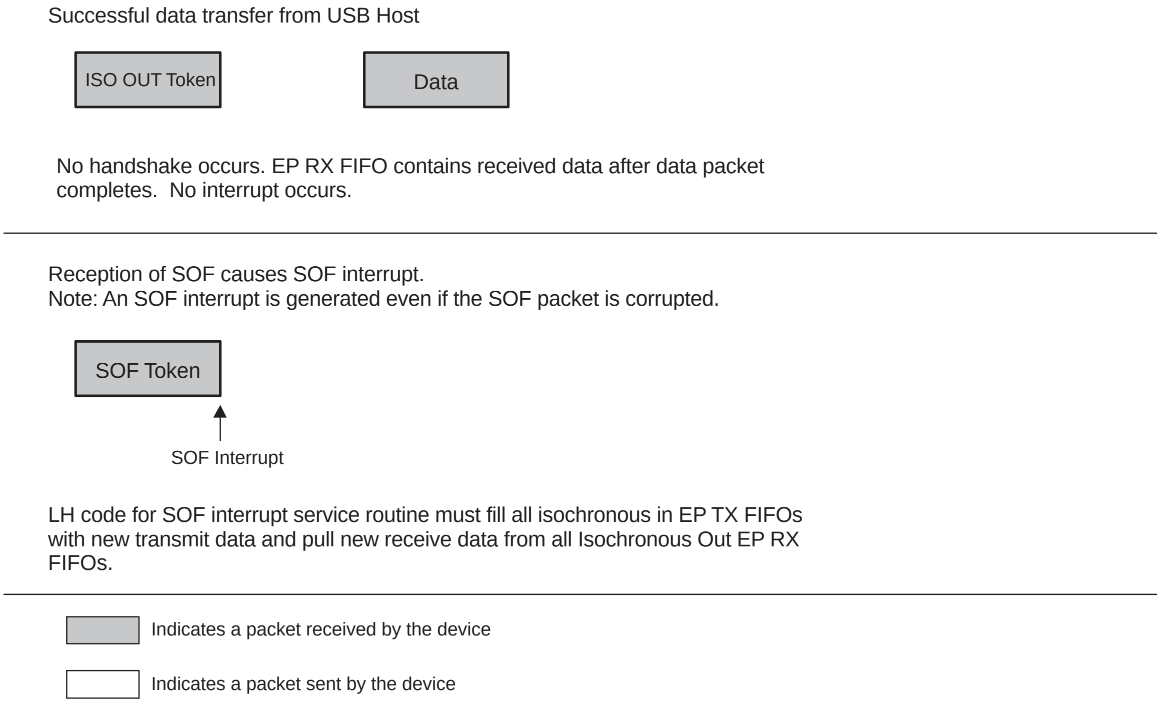
Isochronous OUT transactions are USB transactions in which a given amount of data is transferred from the USB host to the USB device controller module device every 1-ms USB frame. No USB handshaking is provided, and no endpoint-specific interrupt to the CPU is generated at completion of an isochronous OUT transaction. The CPU is responsible for handling isochronous OUT data at each start of frame (SOF) interrupt.

At every SOF interrupt, for each isochronous OUT endpoint, CPU code must select the endpoint by writing the appropriate value in the EP_NUM register and check STAT_FLG.ISO_FIFO_EMPTY. If the RX FIFO contains data, code must read the RXFSTAT.RXF_COUNT value (if the number of bytes to read from RX FIFO is not known), read all the bytes from RX FIFO via the data register, and then clear the EP_NUM.EP_SEL bit.

Because the USB transaction for the isochronous endpoint can occur at any time during the USB 1-ms frame, the USB interface implements a double-buffering of the endpoint receive data FIFO. The endpoint includes two FIFOs, each of which is the length of the configured isochronous endpoint. At all times, one of the two FIFOs is foreground and the other is background. The USB interface side of the USB module is allowed to write to the background RX FIFO, and the CPU is allowed to read to the foreground RX FIFO. The designations foreground and background are swapped at each start of frame (SOF). Isochronous endpoint FIFOs in the background are always enabled to the USB, whereas the foreground FIFOs are enabled to the CPU.

Figure 32-53 shows the two phases (ISO OUT token and data) of an isochronous OUT data transfer in the top portion of the figure. No endpoint-specific interrupt to the CPU is generated for the isochronous OUT transaction. The data for isochronous endpoints are instead handled by the CPU at each start of frame (SOF) interrupt, which is shown as the second case in Figure 32-53.

Figure 32-53. Isochronous OUT Transaction Phases and Interrupts


Successful data transfer from USB Host
ISO OUT Token
Data
SOF Token
SOF Interrupt
Indicates a packet received by the device
Indicates a packet sent by the device

No handshake occurs. EP RX FIFO contains received data after data packet completes. No interrupt occurs.

Reception of SOF causes SOF interrupt.
Note: An SOF interrupt is generated even if the SOF packet is corrupted.

LH code for SOF interrupt service routine must fill all isochronous in EP TX FIFOs with new transmit data and pull new receive data from all Isochronous Out EP RX FIFOs.

32.3.5.1 Isochronous OUT Endpoint Handshaking

Because isochronous endpoint transactions have no handshake packets, the STAT_FLG.STALL, STAT_FLG.NAK, and STAT_FLG.ACK bits for isochronous endpoints always return 0. Because there is no handshake, the endpoint-specific interrupt for isochronous endpoints is not used.

32.3.5.2 Isochronous OUT Transaction Error Conditions

If the CPU fails to read all of the data in the ISO OUT endpoint foreground FIFO by the time the foreground and background FIFOs are switched (at the next SOF), the endpoint FIFO that is being switched to the background is flushed, and the STAT_FLG.DATA_FLUSH bit is asserted for the duration of the next frame.

There is no special indication for the case in which the USB host does not provide a transaction to an ISO OUT endpoint during a frame, but once the FIFO that was background in that frame is foreground, the FIFO is empty (a 0-length data ISO OUT transaction also results in an empty FIFO and cannot be distinguished from a missed ISO OUT transaction).

If an ISO OUT transaction occurs with data error (CRC, PID check, or bit stuffing), the RX FIFO is empty at the next SOF interrupt, and the STAT_FLG.ISO_ERR bit is asserted for the duration of the next frame.

32.3.5.3 Isochronous OUT Endpoint FIFO Error Conditions

The CPU must never read more data than the value given by RXFSTAT.RXF_COUNT.

If the USB host sends more data than the FIFO can contain, the FIFO is cleared and the STAT_FLG.ISO_ERR is set at the next SOF interrupt. A properly configured USB system does not do this.

NOTE: Both foreground and background isochronous FIFOs are cleared when the CTRL.CLR_EP bit is set.

32.3.6 Isochronous IN (CPU→USB HOST) Transactions

Isochronous IN transactions are USB transactions in which a given amount of data is transferred from the USB device controller module device to the USB host every 1-ms USB frame. No handshaking is provided.

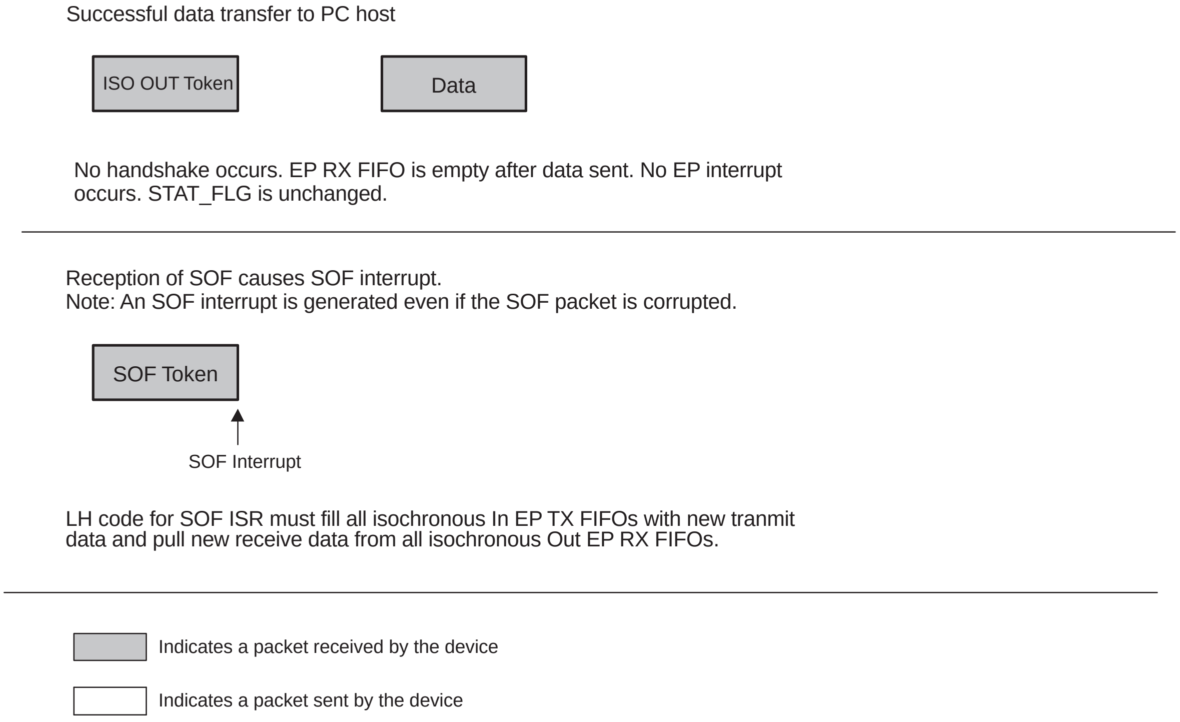
The USB module provides double-buffering of data for ISO IN endpoints; the background FIFO is used as the source of data for IN transactions to the ISO endpoint, and the foreground FIFO can be written to by the CPU. When an IN transaction to an ISO endpoint occurs, the USB module sends all data found in the endpoint background TX FIFO. The CPU is responsible for providing new data to the isochronous IN endpoint foreground TX FIFO at each start of frame interrupt.

In response to the SOF interrupt, for each isochronous IN endpoint, CPU code selects the endpoint (via the EP_NUM register), and then fills the endpoint TX FIFO (via the DATA register). Once all the transmit data have been written to the FIFO, the CPU code must clear the EP_NUM.EP_SEL bit.

Because the USB transaction for the isochronous endpoint can occur at any time during the USB 1-ms frame, the USB interface implements a double-buffering of the endpoint transmit data FIFO. The endpoint includes two FIFOs, each of which is the length of the configured isochronous endpoint. At all times, one of the two FIFOs is foreground and the other is background. The USB interface side of the USB module is allowed to read from the background TX FIFO, and the CPU is allowed to write to the foreground TX FIFO. The designations foreground and background are swapped, and the new background TX FIFO is cleared at each start of frame (SOF). Because ISO endpoints implement double-buffering, ISO endpoints do not control access to the FIFOs via a CTRL.SET_FIFO_EN bit; the CTRL.SET_FIFO_EN and the STAT_FLG.FIFO_EN bits are not implemented for ISO IN endpoints.

Figure 32-54 shows the transaction phases associated with isochronous IN transactions and the SOF transaction. No endpoint-specific interrupt to the CPU is generated as a result of an isochronous IN transaction, and there is no handshake phase. The SOF transaction causes an SOF interrupt to the CPU; it is assumed that the CPU refills the isochronous IN endpoint transmit FIFO at each SOF interrupt.

Figure 32-54. Isochronous IN Transaction Phases and Interrupts


Successful data transfer to PC host
ISO OUT Token
Data
No handshake occurs. EP RX FIFO is empty after data sent. No EP interrupt occurs. STAT_FLG is unchanged.
Reception of SOF causes SOF interrupt.
Note: An SOF interrupt is generated even if the SOF packet is corrupted.
SOF Token
SOF Interrupt
LH code for SOF ISR must fill all isochronous In EP TX FIFOs with new tranmit data and pull new receive data from all isochronous Out EP RX FIFOs.
Indicates a packet received by the device
Indicates a packet sent by the device

32.3.6.1 Isochronous IN Endpoint Handshaking

Because isochronous endpoint transactions have no handshake packets, the STAT_FLG.STALL, STAT_FLG.NAK, and STAT_FLG.ACK bits for isochronous endpoints always return 0. Because there is no handshake, there is no endpoint-specific interrupt to the CPU to report handshake results for isochronous endpoints.

32.3.6.2 Isochronous IN Transaction Error Conditions

If the USB host did not successfully complete an ISO IN transaction in the previous frame, and if data were present in TX FIFO to be sent at the IN transaction, the STAT_FLG.MISS_IN bit is asserted for the duration of the following frame. If the ISO IN endpoint is cleared in the middle of a USB transaction to the background FIFO, the macro forces a bit stuffing error for the ISO transaction.

32.3.6.3 Isochronous IN Endpoint FIFO Error Conditions

If the CPU attempts to overfill the configured endpoint FIFO, data written to DATA register after the TX FIFO is full is lost, but any data that was successfully put into the FIFO is transmitted when that FIFO is the background FIFO and an IN transaction for that endpoint occurs. Because an ISO TX FIFO is cleared automatically on the toggle from background to foreground, there is no reason to clear the FIFO. However, if the CPU does not wish to send the data it wrote, clearing the endpoint is the only mechanism to do this.

32.3.7 Control Transfers on Endpoint 0

Control transfers on endpoint 0 include control write and control read transfers. Control write and control read transfers are each composed of two or more transactions to endpoint 0. Additionally, the USB device controller module is capable of autodecoding some control write and control read transfers. These operations are summarized in Figure 32-55 and Figure 32-56. An IN or an OUT transaction is received out of a control request. This transaction is automatically stalled by the core.

Non-autodecoded control read and control write transfers are sets of transactions that occur on endpoint 0 and have specific USB protocol meaning but are not handled automatically by the core. The USB device controller block automatically provides an ACK handshake for the setup stage transaction, but the data and status stage transaction handshaking is accomplished using the usual RX and TX control bits that affect transaction handshaking. A general USB interrupt to the CPU occurs at the end of each transaction of each stage of a control transfer. The CPU must perform the following actions to act on non-autodecoded control transfers:

- Process the setup phase setup USB interrupt. The CPU reads the control transfer command from the setup FIFO and decodes the command. For control reads, the CPU fetches the requested read data and places it (or as much of the read data as fits) into the endpoint 0 FIFO, and then enables the endpoint 0 FIFO. For control writes, the CPU code only enables the endpoint 0 FIFO. CPU code also sets any flags needed for processing endpoint 0 USB interrupts during the control transfer.
- Process the data phase endpoint 0 general USB interrupt(s). For control reads, the data phase general USB endpoint 0 TX interrupt indicates that the previously provided transmit data has been sent. Any additional data must be written to the endpoint 0 FIFO. For control writes, the write data must be pulled from the endpoint 0 FIFO, and when all control write data is available, interpret the write data and act on the write request. After handling the last data phase interrupt, the CPU must set the endpoint 0 control bits to signal the desired status to the host.
- Process the status stage endpoint 0 general USB interrupt. The CPU provides its completion status back to the USB host during this stage, either via status in the data phase of the transaction (for control write transfers) or via the handshake phase of the transaction (for control read transfers).

Figure 32-55. Stages and Transaction Phases of Autodecoded Control Transfers

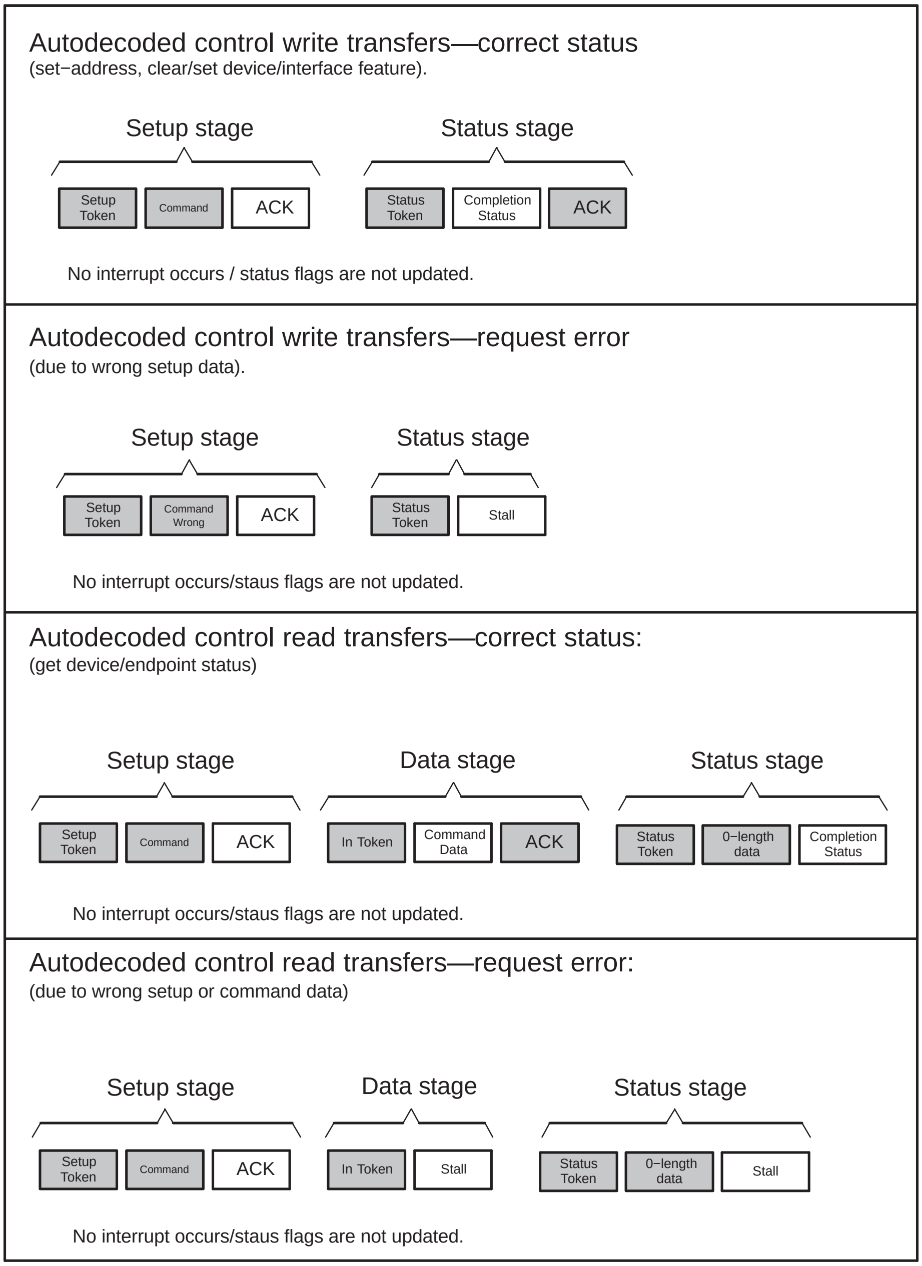
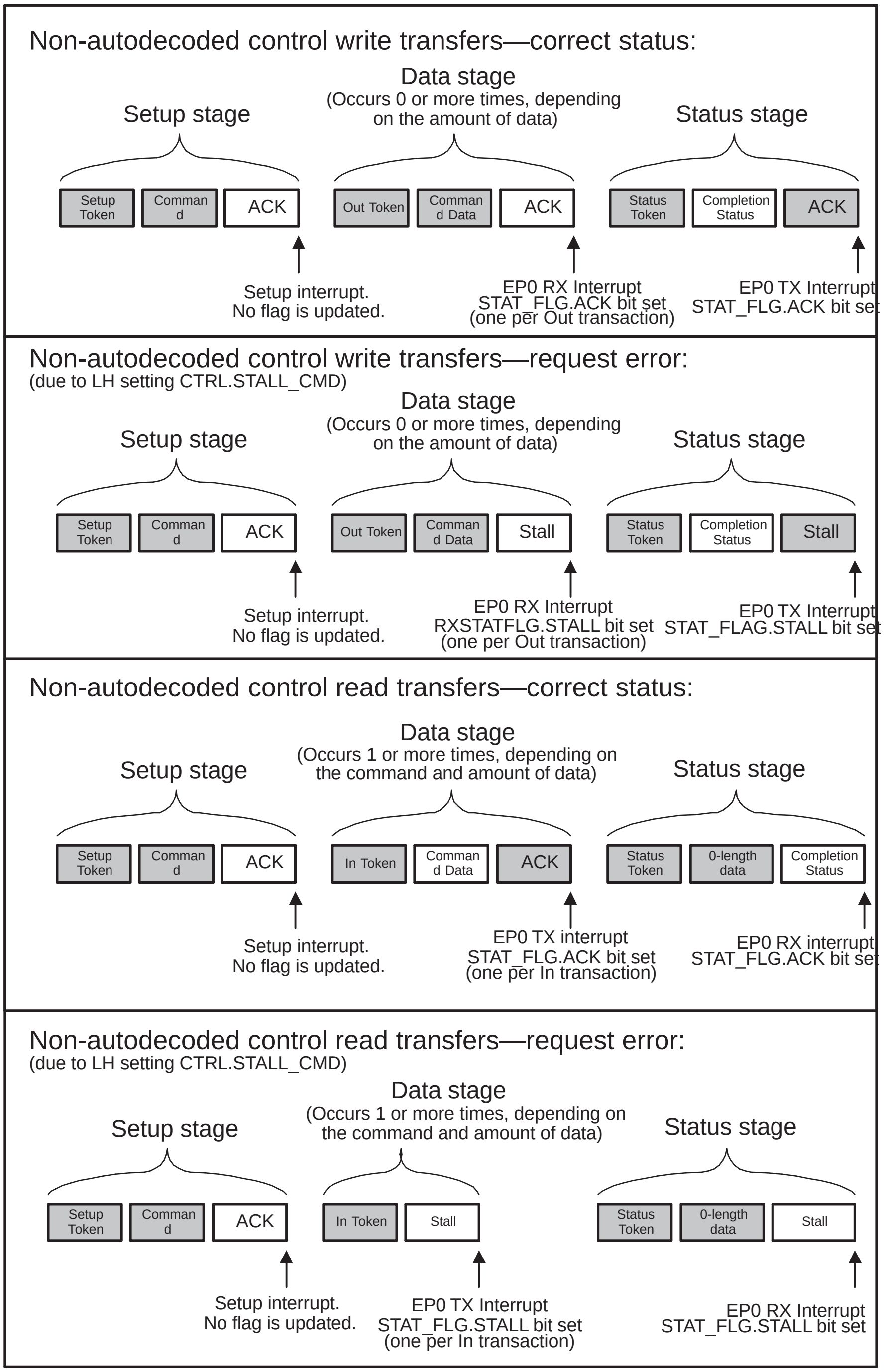

Autodecoded control write transfers—correct status
(set−address, clear/set device/interface feature).
Setup stage
Status stage
Setup Token
Command
ACK
Status Token
Completion Status
ACK
No interrupt occurs / status flags are not updated.
Autodecoded control write transfers—request error
(due to wrong setup data).
Setup stage
Status stage
Setup Token
Command Wrong
ACK
Status Token
Stall
No interrupt occurs/staus flags are not updated.
Autodecoded control read transfers—correct status:
(get device/endpoint status)
Setup stage
Data stage
Status stage
Setup Token
Command
ACK
In Token
Command Data
ACK
Status Token
0−length data
Completion Status
No interrupt occurs/staus flags are not updated.
Autodecoded control read transfers—request error:
(due to wrong setup or command data)
Setup stage
Data stage
Status stage
Setup Token
Command
ACK
In Token
Stall
Status Token
0−length data
Stall
No interrupt occurs/staus flags are not updated.

Figure 32-56. Stages and Transaction Phases of Non-Autodecoded Control Transfers


Non-autodecoded control write transfers—correct status:
Setup stage
Data stage
(Occurs 0 or more times, depending on the amount of data)
Status stage
Setup Token
Comman d
ACK
Out Token
Comman d Data
ACK
Status Token
Completion Status
ACK
Setup interrupt. No flag is updated.
EP0 RX Interrupt STAT_FLG.ACK bit set (one per Out transaction)
EP0 TX Interrupt STAT_FLG.ACK bit set
Non-autodecoded control write transfers—request error:
(due to LH setting CTRL.STALL_CMD)
Setup stage
Data stage
(Occurs 0 or more times, depending on the amount of data)
Status stage
Setup Token
Comman d
ACK
Out Token
Comman d Data
Stall
Status Token
Completion Status
Stall
Setup interrupt. No flag is updated.
EP0 RX Interrupt RXSTATFLG.STALL bit set (one per Out transaction)
EP0 TX Interrupt STAT_FLAG.STALL bit set
Non-autodecoded control read transfers—correct status:
Setup stage
Data stage
(Occurs 1 or more times, depending on the command and amount of data)
Status stage
Setup Token
Comman d
ACK
In Token
Comman d Data
ACK
Status Token
0-length data
Completion Status
Setup interrupt. No flag is updated.
EP0 TX interrupt STAT_FLG.ACK bit set (one per In transaction)
EP0 RX interrupt STAT_FLG.ACK bit set
Non-autodecoded control read transfers—request error:
(due to LH setting CTRL.STALL_CMD)
Setup stage
Data stage
(Occurs 1 or more times, depending on the command and amount of data)
Status stage
Setup Token
Comman d
ACK
In Token
Stall
Status Token
0-length data
Stall
Setup interrupt. No flag is updated.
EP0 TX Interrupt STAT_FLG.STALL bit set (one per In transaction)
EP0 RX Interrupt STAT_FLG.STALL bit set

Autodecoded control read and control write transfers are sets of transactions that occur on endpoint 0 that have specific USB protocol meaning and are handled automatically by the USB device controller block without any intervention by the CPU. The USB device controller block handles all handshaking automatically and without regard to the endpoint 0 control bits that affect normal (non-control transfer) transaction handshaking. No interrupt is asserted to the CPU during autodecoded control transfers.

If no USB1.1 specification defined request is associated with the data of the setup phase, request is stalled by the core and the CPU is not informed of its occurrence (autodecoded).

When a setup token is identified, the USB decode module must monitor the setup stage data packet, decode it, and determine whether it is an autodecoded or a non-autodecoded transfer and a control read or a control write. If it is a valid non-autodecoded request, the setup FIFO is immediately cleared and control of the FIFO is immediately taken away from the CPU (if the CPU had control of the FIFO). New setup data are placed into the setup FIFO, and the setup interrupt flag is set (IRQ_SRC.setup).

In response to the setup interrupt, the CPU must select the setup FIFO by setting the EP_NUM.SETUP_SEL bit. This clears the IRQ_SRC.SETUP flag. The CPU must then read 8 bytes from the setup FIFO, clear the EP_NUM.SETUP_SEL bit, and confirm that IRQ_SRC.SETUP bit has not been reset by a new setup transaction. If the IRQ_SRC.SETUP flag is asserted, the CPU must discard the previously read data and handle the new setup packet as explained above. Thus the CPU never misses a new occurring setup transaction (per USB 1.1 specification).

32.3.7.1 Autodecoded Control Write Transfers

For set address control write transfers, the USB address provided in the setup token is captured to the USB module device address register. If new address is different from 0, the device moves into addressed state (DEVSTAT.ADD set) if it was not already addressed.

For set and clear feature control writes, the appropriate feature information bit is set or cleared. When a set or clear feature transfer occurs to set or clear the device remote wake-up feature, the DEVSTAT.R_WK_OK bit is set or cleared, as appropriate. If a set or clear interface feature occurs, the request is automatically stalled by the core because no feature is defined for interface (see the USB 1.1 specification).

Per the USB 1.1 specifications, a SET_ADDRESS request is effective after the status stage of the request, even if the status stage does not end with an ACK handshake. SET/CLEAR_FEATURE requests are effective after setup stage, even if no status stage occurs.

32.3.7.1.1 Autodecoded Control Write Transfer Handshaking

The USB device controller module automatically provides ACK handshaking for all transactions of all stages of autodecoded control write transfers, except if a corrupted packet is received, which the USB module ignores. The CTRL.SET_FIFO_EN and SYSCON2.STALL_CMD bits have no effect on handshaking.

32.3.7.1.2 Autodecoded Control Write Transfer Error Conditions

If the token packet or the data packet of a setup stage transaction has an error (bad CRC, PID check, or bit stuffing error), the USB block ignores the transaction. The USB block does not provide ACK handshaking in this case.

32.3.7.2 Autodecoded Control Read Transfers

Autodecoded control reads include the standard device requests GET_ENDPOINT and DEVICE_STATUS. These control read transfers access information that is kept in registers inside the USB module, so CPU code is not involved in filling the read data into the TX FIFO.

The USB module returns the currently selected appropriate status information (depending on the wIndex value in the setup stage data packet) during the data phase of the single IN transaction of the data stage, and provides ACK as the handshake for the status stage handshake phase. The CPU receives no interrupt.

32.3.7.2.1 Autodecoded Control Read Transfer Handshaking

The USB device controller module automatically provides ACK handshaking for all transactions of all stages of autodecoded control read transfers, except if a corrupted token packet is received, which the USB module ignores. The CTRL.FIFO_EN and SYSCON2.STALL_CMD bits have no effect on the handshaking. If the status packet has a DATA0 PID instead of a DATA1 PID, status is STALLed and no interrupt is asserted to the CPU. If the setup packet has a DATA1 PID instead of a DATA0 PID, setup transaction is ignored (error).

32.3.7.2.2 Autodecoded Control Read Transfer Error Conditions

If the token phase or the data phase of a setup stage transaction has an error (bad CRC, PID check, or bit stuffing error), the USB block ignores the transaction. The USB block does not provide ACK handshaking in this case.

Data errors during the data stage of autodecoded control write transfers are handled in the standard way; any data stage transaction from the host where a data error occurs is ignored.

It is possible that the USB host sends a GET_ENDPOINT/DEVICE_STATUS request with a bad parameter. If the autodecode mechanism senses a bad parameter in the setup stage data phase, the autodecode mechanism causes a STALL handshake to be signaled during the data phase of the data stage and during the status stage.

32.3.7.3 Non-Autodecoded Control Write Transfers

Non-autodecoded control write transfers include the SET_/CLEAR_ENDPOINT feature, SET_CONFIGURATION, SET_INTERFACE, SET_DESCRIPTOR and class- or vendor-specific control write transfers. Non-autodecoded control write transfers consist of two or three stages [setup, data (optional), and status].

The setup stage of a valid non-autodecoded control write transfer consists of one SETUP transaction from USB host to USB device. At the end of the setup stage handshake, the USB module generates an CPU general USB interrupt with the IRQ_SRC.SETUP flag set. The CPU must respond to this general USB interrupt by setting EP_NUM.SETUP_SEL bit, which clears the setup interrupt flag. The CPU must then read 8 bytes from the setup FIFO via the DATA register, clear EP_NUM.EP_SEL bit, and check the IRQ_SRC.SETUP flag. If the IRQ_SRC.SETUP flag is set, the CPU must discard the setup data it has just read and handle the new setup data packet following the same scheme. If the IRQ_SRC.SETUP flag is cleared, the CPU code interprets this request information and performs any application-specific activity needed because of the setup stage request. If there is one or more data stage for the transfer, the CPU must set the CTRL.SET_FIFO_EN bit for endpoint 0 to allow the core to accept RX data from the coming OUT transaction.

The data stage for non-autodecoded control writes consists of zero or more OUT transactions. Transaction handshaking and interrupt generation as for non-isochronous, non-control OUT endpoints applies. The CPU can cause NAK, STALL, or ACK signaling for the data stage transactions. If ACK was signaled on a given general USB interrupt, the CPU must respond by reading the data from the endpoint 0 RX FIFO and saving it for processing.

After completion of the data stage, a status stage IN transaction occurs. The USB module provides handshaking to the USB host based on the endpoint 0 handshaking control bit STAT_FLG.FIFO_EN. The CPU can delay signaling completion of the control write transfer by forcing NAK handshaking to the host during the status stage (by holding STAT_FLG.FIFO_EN 0), or causing ACK handshaking by setting the CTRL.SET_FIFO_EN bit with an empty endpoint 0 FIFO. An endpoint 0 TX general USB interrupt is sent to the CPU at completion of the status stage.

After a SET_CONFIGURATION request, the device moves in addressed or configured state as soon as the CPU sets the SYSCON2.DEV_CFG or SYSCON2.CLR_CFG bits.

32.3.7.3.1 Specific CPU Required Actions

If the device receives a valid set endpoint halt feature request, it must set the appropriate CTRL.SET_HALT control bit.

If the device receives a valid CLEAR_ENDPOINT halt feature request, it must set the appropriate CTRL.RESET_EP bit to clear the halt condition, FIFO flags, and reset data PID to DATA0 for the endpoint. If the specified endpoint number is 0, the CPU has only to set CTRL.CLR_HALT bit to clear the halt condition.

If the device receives a valid SET_CONFIGURATION request, it must reset all endpoints by setting the CTRL.RESET_EP control bits, set the SYSCON1.SELF_PWR bit to the appropriate value, and then set halt conditions for endpoints not used by the default interface set for the configuration. If the device was addressed when the SET_CONFIGURATION was received, the CPU must write 1 to the SYSCON2.DEV_CFG bit to allow the device to move into the configured state (DEVSTAT.CFG bit set). If the device was configured when the SET_CONFIGURATION was received, and the new configuration value is 0, the CPU must write 1 to the SYSCON2.CLR_CFG bit to allow the device to move back into the addressed state (DEVSTAT.CFG bit cleared).

If the device receives a valid set interface request, it must reset all endpoints used by the interface set, by setting CTRL.RESET_EP control bits, and then set halt conditions for endpoints not used by this interface.

Other CPU required actions are specific to the request and not detailed in this document.

32.3.7.3.2 Non-Autodecoded Control Write Transfer Handshaking

Setup stage transactions that are valid are signaled ACK. Transactions with invalid setup stage token or data packets are ignored and receive no handshake packet from the USB module, and there is no interrupt generated.

Data stage handshaking for non-autodecoded control write transfers is dependent on the endpoint 0 STAT_FLG.FIFO_EN, STAT_FLG.EP_HALTED, and SYSCON2.STALL_CMD bits. The CPU can delay completion of any transaction of the data stage by signaling NAK (via CTRL.SET_FIFO_EN bit not set). The USB specification requires that once STALL is signaled in a control transfer, it must be signaled on that endpoint until the next setup token is received. Either the SYSCON2.STALL_CMD or the CTRL.SET_HALT (reflected in the STAT_FLG.EP_HALTED register bit) register bits provide this functionality. STAT_FLG.EP_HALTED does not reflect the forced STALL caused by SYSCON2.STALL_CMD; it retains its previous value.

Status stage handshaking is controlled by the endpoint 0 STAT_FLG.FIFO_EN and SYSCON2.STALL_CMD bits. Successful completion of a non-autodecoded control write transfer is indicated by the USB device controller module returning a zero length data payload for the data phase of the status stage and an ACK handshake from the host for the handshake phase of the status stage. Although NAK handshaking can be used to indicate delays in completion of the requested control write, the USB host can choose to abort the control write after some number of NAKs.

32.3.7.3.3 Non-Autodecoded Control Write Transfer Error Conditions

If an error occurs while dealing with the control write, which the CPU cannot deal with itself, it must signal STALL to the USB host for all subsequent transactions until a new setup token to endpoint 0 occurs. This is true for both data stage and status stage transactions. This is most conveniently done by setting endpoint 0 SYSCON2.STALL_CMD bit, which causes stalling of all the remaining transactions of all remaining stages of a non-autodecoded control transfer, up to the reception of the next valid SETUP command.

Error conditions are handled as for BULK/INTERRUPT transactions. If a packet is received corrupted, the core ignores the transaction and no interrupt is asserted.

32.3.7.4 Non-Autodecoded Control Read Transfers

Non-autodecoded control read transfers include the GET_INTERFACE_STATUS, GET_CONFIGURATION, GET_INTERFACE, GET_DESCRIPTOR, SYNCH_FRAME and class- or vendor-specific control read transfers. Non-autodecoded control read transfers consist of three stages (setup, data, and status).

The setup stage of a valid non-autodecoded control read transfer consists of one SETUP transaction from USB host to USB device. At the end of the setup stage handshake, the USB module generates a CPU general USB interrupt with the IRQ_SRC.SETUP flag set. The CPU must respond to this general USB interrupt by setting the EP_NUM.SETUP_SEL bit, which clears the setup interrupt flag. The CPU must then read 8 bytes from the setup FIFO via the DATA register, clear the EP_NUM.EP_SEL bit, and check the IRQ_SRC.SETUP flag. If the IRQ_SRC.SETUP flag is set, the CPU must discard the setup data it has just read and handle the new setup data packet following the same scheme. If the IRQ_SRC.SETUP flag is cleared, the CPU code interprets this request information and then prepares data for the IN transaction that follow. This includes placing the data being requested (or the first few bytes, if more than one FIFO worth of data is being returned) into the endpoint 0 FIFO, and setting the CTRL.SET_FIFO_EN bit.

The data stage of a control read transfer consists of one or more IN transactions. Transaction handshaking and interrupt generation as for non-isochronous, non-control IN endpoints applies; the CPU can cause NAK, STALL, or ACK signaling for the data stage transactions. At endpoint 0 TX general USB interrupts, CPU code must move more data to the endpoint 0 FIFO, until the last bytes of the requested data have been provided. Although SETUP packets have a defined payload length, the USB host can cancel the transaction at any time, without the status stage, and resend another SETUP command. The CPU code must be able to operate correctly in this situation.

After completion of the data stage, a status stage OUT transaction occurs. The USB host sends a zero-length data packet, and the CPU code must return its completion status for the control read standard request via standard handshaking mechanisms.

NOTE: In the case of returning exactly what the host requested and that request was a multiple of the maximum packet size, no zero-length packet is required. A zero-length packet is required only when the amount of data the device has to return is less than the amount requested by the host and the amount returned is a multiple of the maximum packet size (source USB forum).

32.3.7.4.1 Non-Autodecoded Control Read Transfer Handshaking

Handshaking for the setup stage of non-autodecoded control read transfers is forced by the USB module to always be ACK, unless there is a data error in the packet, in which case the USB module ignores the transaction. If the setup packet has a DATA1 PID instead of a DATA0 PID, the setup transaction is ignored (error).

Data stage handshaking for non-autodecoded control read transfers is dependent on the endpoint 0 STAT_FLG.FIFO_EN, STAT_FLG.EP_HALTED, and SYSCON2.STALL_CMD bits. The handshaking information is used during the data phase of the data stage transaction. The USB specification requires that once STALL is signaled in a control transfer, it must be signaled until the next setup token is received. The SYSCON2.STALL_CMD and CTRL.SET_HALT (reflected through the STAT_FLG.EP_HALTED register bit) register bits provide this functionality. STAT_FLG.EP_HALTED does not reflect the forced STALL caused by SYSCON2.STALL_CMD; it retains its previous value.

The status stage is controlled by the CTRL.FIFO_EN andSYSCON2.STALL_CMD bits.

Successful completion of non-autodecoded control read transfers is indicated by the host sending an OUT token followed by an empty packet and the USB device controller responding with ACK. If the data packet sent by the USB host during the status stage of a control read request is not empty, the OUT transaction is accepted by the core, but OUT data is not put into the endpoint 0 RX FIFO. If the status packet has a DATA0 PID instead of a DATA1 PID, a STALLed is returned by the core and an interrupt is asserted.

32.3.7.4.2 Non-Autodecoded Control Read Transfer Error Conditions

If an error occurs while dealing with the control read, which the CPU cannot deal with itself, it must signal STALL to the USB host for all subsequent transactions until a new setup token to endpoint 0 occurs. This is true for both data stage and status stage transactions. This is most conveniently done by setting endpoint 0 SYSCON2.STALL_CMD bit, which causes stalling of all the remaining transactions of all remaining stages of a non-autodecoded control transfer, up to the reception of the next valid SETUP command.

Error conditions are handled as for BULK/INTERRUPT transactions. The USB device controller module responds to control read status stage transactions that have a bad token or bad data by not sending a handshake packet. In both cases, the transaction is ignored and no general USB interrupt is generated to the CPU.

32.3.7.5 Autodecoded Versus Non-Autodecoded Control Requests

Table 32-55. Autodecoded Versus Non-Autodecoded Control Requests [1] [2] [3] [4]

Request	Recipient	Status	CPU Required Action	Device Behavior if Device Is Not Configured
GET_ STATUS	Device	Autodecoded (function of AUTODEC_DIS register bit)	None	Device status is returned (SYSCON1.SELF_PWR and DEVSTAT.R_WK_OK bits).
	Interface	Non-autodecoded	The CPU must stall the command (via SYSCON2.STALL_CMD bit) if interface number is not correct. No feature is defined for interface.	Command is passed to the CPU.
	Endpoint	Autodecoded(function of AUTODEC_DIS register bit)	None	The core automatically stalls the command if endpoint number is different from 0.
CLEAR/SET FEATURE	Device	Autodecoded (function of AUTODEC_DIS register bit)	None (DS_CHG IT is asserted to the CPU after any DEVSTAT.R_WK_OK bit modification).	The core handles the request.
	Interface	Autodecoded (function of AUTODEC_DIS register bit)	None. (No feature is defined in USB 1.1 spec for interface. These requests are stalled).	Command is stalled anyway.
	Endpoint	Non-autodecoded	The CPU must stall the command (via SYSCON2.STALL_CMD bit) if endpoint number/type/direction is not correct. The CPU must reset the EP after having handled the pending transactions (if CLEAR) or set halt condition (if SET). For EP 0, CPU must only clear or set halt condition; FIFO and data PID is always correct for next setup.	Command is passed to the CPU.
SET_ ADDRESS	Device	Autodecoded	None (see [5]) (Whether the device is addressed or not is available in DEVSTAT register. A valid SET_ADDRESS request with address number from 0 generates a DS_CHG interrupt to the CPU).	Default: device moves in the addressed state if address number is different from 0. Addressed: device takes the new address value or moves in default state if address number is 0. Configured: request is STALLed.

[1] Transactions on endpoints other than zero are ignored if the device is not configured (addressed state).

[2] If some endpoints are not used by the interface currently set, transactions on these endpoints are not ignored; the CPU must set halt feature for the endpoint. This does not happen if USB host works correctly.)

[3] If endpoint 0 is halted, per *USB 1.1 Specification* (see 9.4.5: Get_Status), all requests are stalled except GET_STATUS, CLEAR_FEATURE, and SET_FEATURE requests.)

[4] Requests are handled with respect to USB 1.1 when specified as such, but many device reactions are not specified by USB 1.1.)

[5] During a SET_ADDRESS autodecoded command, only the seven LSBs are significant for the new address (decimal value from 0 to 127); all others are discarded.

Table 32-55. Autodecoded Versus Non-Autodecoded Control Requests [(1)] [(2)] [(3)] [(4)] (continued)

Request	Recipient	Status	CPU Required Action	Device Behavior if Device Is Not Configured
GET_ DESCRIPTOR	All	Non-autodecoded	The CPU must write descriptor data into endpoint 0 FIFO.	Command is passed to the CPU.
SET_ DESCRIPTOR	All	Non-autodecoded	The CPU must stall the command (via SYSCON2.STALL_CMD bit) if it does not support set descriptor requests.	Command is passed to the CPU.
GET/SET CONFIGURATION	Device	Non-autodecoded	The CPU must stall the command (via SYSCON2.STALL_CMD bit) if configuration number is not correct. If the request is SET_CONFIG, the CPU must reset all endpoints, halt endpoints not used by the default interface setting, set SYSCON1.SELF_PWR value if device is self-powered for the configuration set, and then set SYSCON2.DEV_CFG bit (if config nb is not 0), or set SYSCON2.CLR_CFG bit (if config nb is 0) before allowing status stage to complete. The device moves to configured state (if DEV_CFG set), or moves to addressed state (if CLR_CFG set) and a DS_CHG interrupt is asserted to the CPU.	Command is passed to the CPU.
GET/SET INTERFACE	Interface	Non-autodecoded	The CPU must stall the command (via SYSCON2.STALL_CMD bit) if interface/setting number is not correct. If the request is SET_INTER-FACE, the CPU must reset endpoints used by the interface and halt endpoints not used by the interface setting before allowing status stage to complete.	Command is passed to the CPU.
SYNCH_ FRAME	Endpoint	Non-autodecoded	The CPU must stall the command if it does not support SYNCH_FRAME request, else write requested data in the endpoint 0 FIFO.	Command is passed to the CPU.

32.3.7.6 Note on Control Transfers Data Stage Length

The control transfer data stage length is indicated in the setup data packet.

During control reads, if the USB host requests more data than indicated in the setup packet, unexpected IN transaction is STALLed, causing STALL handshake for all remaining transactions of the transfer until next SETUP. If the USB host requires less data than indicated in the setup packet, the transfer is not STALLed. However, if the host moves to status stage earlier than expected for a non-autodecoded request, the OUT status stage is NAKed because the CPU has not enabled the RX FIFO.

During control writes, if the USB host sends more bytes than indicated in the setup packet, the transfer is STALLed. If the USB host sends fewer bytes than were expected, the request is accepted. But if the USB host moves to status stage earlier than expected for a non-autodecoded request, the IN status stage is NAKed because the CPU has not enabled the TX FIFO.

32.3.8 USB Device Initialization

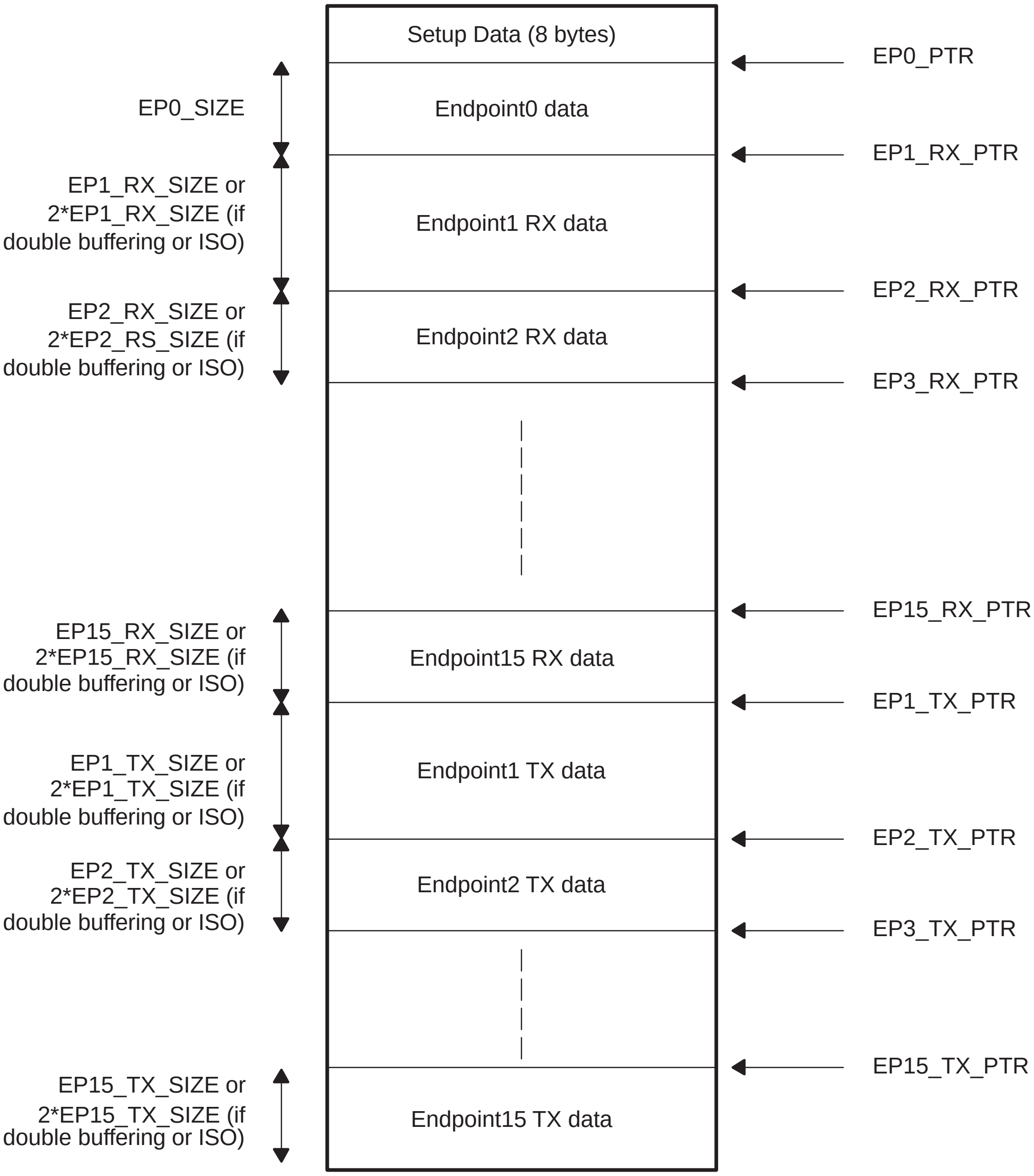
To allow communication between the device and a USB host, the CPU must configure the device by filling the configuration registers.

For each endpoint, the CPU must write on the dedicated register:

- Endpoint size
- Whether double-buffering is allowed for endpoint or not
- Endpoint type (ISO or non-ISO)
- Address of the pointer

System software must choose how to allocate the 2040 available bytes of USB device controller RAM to the USB endpoints. Receive endpoint size and type are configured using the EP1_RX through EP15_RX registers. Transmit endpoint size and type are configured using the EP1_TX through EP15_TX registers. Figure 32-57 shows an example of the RAM organization, obtained by following the flowchart shown in Figure 32-58.

Figure 32-57. Example of RAM Organization


Setup Data (8 bytes)
EP0_PTR
EP0_SIZE
Endpoint0 data
EP1_RX_PTR
EP1_RX_SIZE or 2*EP1_RX_SIZE (if double buffering or ISO)
Endpoint1 RX data
EP2_RX_PTR
EP2_RX_SIZE or 2*EP2_RS_SIZE (if double buffering or ISO)
Endpoint2 RX data
EP3_RX_PTR
EP15_RX_PTR
EP15_RX_SIZE or 2*EP15_RX_SIZE (if double buffering or ISO)
Endpoint15 RX data
EP1_TX_PTR
EP1_TX_SIZE or 2*EP1_TX_SIZE (if double buffering or ISO)
Endpoint1 TX data
EP2_TX_PTR
EP2_TX_SIZE or 2*EP2_TX_SIZE (if double buffering or ISO)
Endpoint2 TX data
EP3_TX_PTR
EP15_TX_PTR
EP15_TX_SIZE or 2*EP15_TX_SIZE (if double buffering or ISO)
Endpoint15 TX data

Once the endpoints are configured, the CPU must set the SYSCON1.CFG_LOCK bit. If this bit is not set, all transactions are ignored by the core. Then, when the CPU is ready to communicate with the USB host, it must set the SYSCON1.PULLUP_EN bit. The CPU can wait until the DS_CHG attach interrupt has been detected and handled before setting the SYSCON1.PULLUP_EN bit. The USB host cannot detect the device until this bit is set.

Figure 32-58 and Figure 32-59 show flowcharts for the configuration phase.

Figure 32-58. Device Configuration Routine

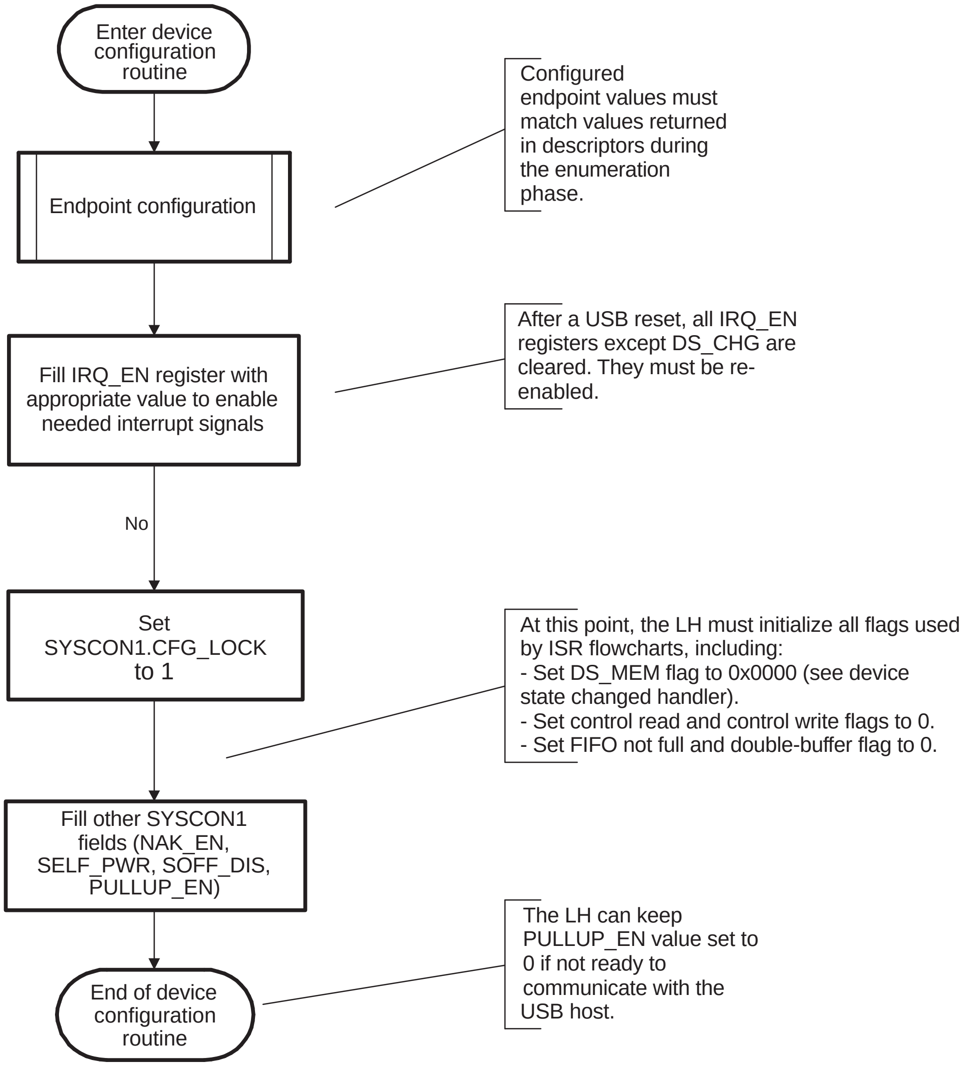
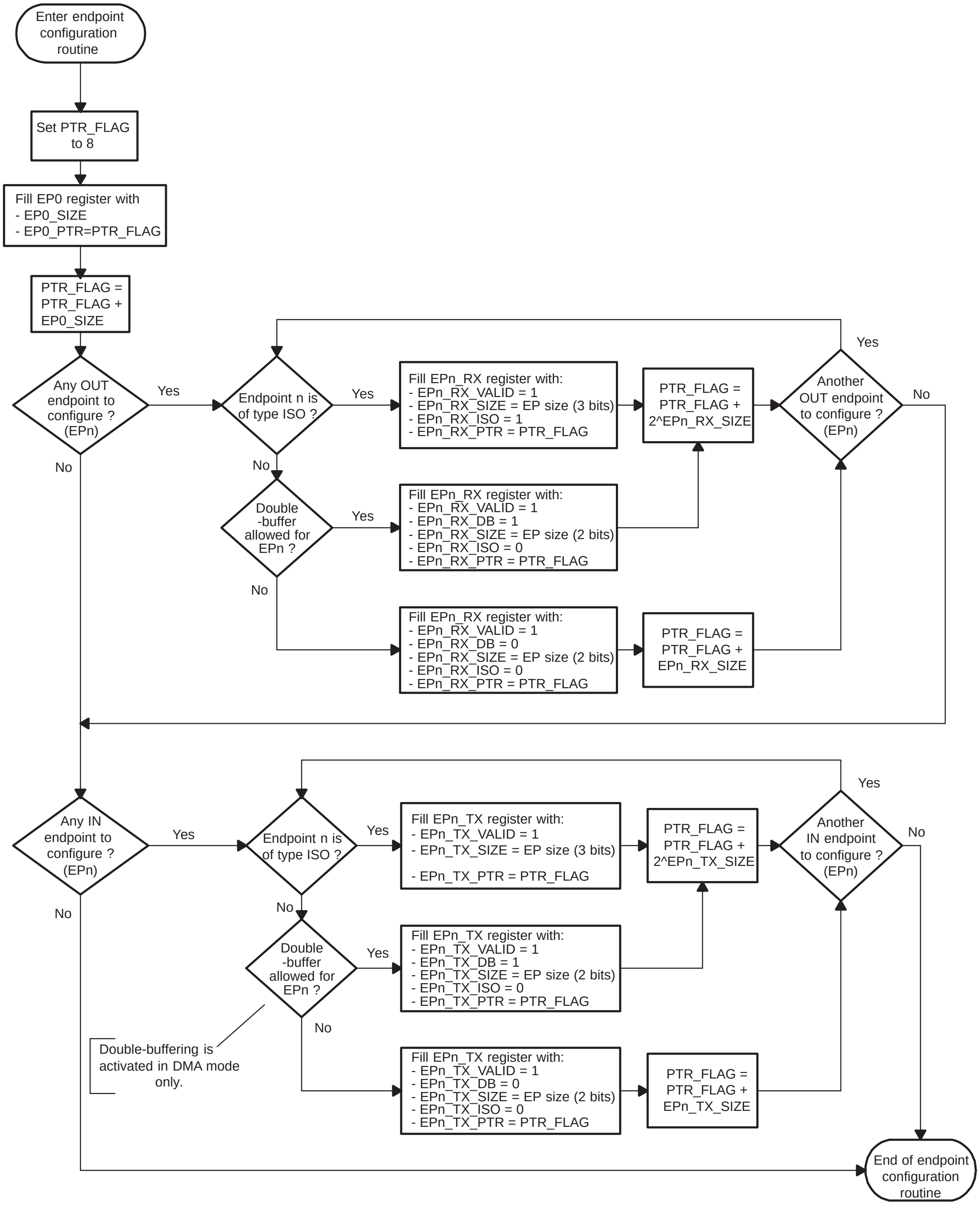

Enter device configuration routine
Endpoint configuration
Fill IRQ_EN register with appropriate value to enable needed interrupt signals
No
Set SYSCON1.CFG_LOCK to 1
Fill other SYSCON1 fields (NAK_EN, SELF_PWR, SOFF_DIS, PULLUP_EN)
End of device configuration routine
Configured endpoint values must match values returned in descriptors during the enumeration phase.
After a USB reset, all IRQ_EN registers except DS_CHG are cleared. They must be re-enabled.
At this point, the LH must initialize all flags used by ISR flowcharts, including:
- Set DS_MEM flag to 0x0000 (see device state changed handler).
- Set control read and control write flags to 0.
- Set FIFO not full and double-buffer flag to 0.
The LH can keep PULLUP_EN value set to 0 if not ready to communicate with the USB host.

Figure 32-59. Endpoint Configuration Routine


Enter endpoint configuration routine
Set PTR_FLAG to 8
Fill EP0 register with
- EP0_SIZE
- EP0_PTR=PTR_FLAG
PTR_FLAG = PTR_FLAG + EP0_SIZE
Any OUT endpoint to configure ? (EPn)
Yes
No
Endpoint n is of type ISO ?
Yes
No
Fill EPn_RX register with:
- EPn_RX_VALID = 1
- EPn_RX_SIZE = EP size (3 bits)
- EPn_RX_ISO = 1
- EPn_RX_PTR = PTR_FLAG
PTR_FLAG = PTR_FLAG + 2^EPn_RX_SIZE
Another OUT endpoint to configure ? (EPn)
Yes
No
Double -buffer allowed for EPn ?
Yes
No
Fill EPn_RX register with:
- EPn_RX_VALID = 1
- EPn_RX_DB = 1
- EPn_RX_SIZE = EP size (2 bits)
- EPn_RX_ISO = 0
- EPn_RX_PTR = PTR_FLAG
Fill EPn_RX register with:
- EPn_RX_VALID = 1
- EPn_RX_DB = 0
- EPn_RX_SIZE = EP size (2 bits)
- EPn_RX_ISO = 0
- EPn_RX_PTR = PTR_FLAG
PTR_FLAG = PTR_FLAG + EPn_RX_SIZE
Any IN endpoint to configure ? (EPn)
Yes
No
Endpoint n is of type ISO ?
Yes
No
Fill EPn_TX register with:
- EPn_TX_VALID = 1
- EPn_TX_SIZE = EP size (3 bits)
- EPn_TX_PTR = PTR_FLAG
PTR_FLAG = PTR_FLAG + 2^EPn_TX_SIZE
Another IN endpoint to configure ? (EPn)
Yes
No
Double -buffer allowed for EPn ?
Yes
No
Fill EPn_TX register with:
- EPn_TX_VALID = 1
- EPn_TX_DB = 1
- EPn_TX_SIZE = EP size (2 bits)
- EPn_TX_ISO = 0
- EPn_TX_PTR = PTR_FLAG
Double-buffering is activated in DMA mode only.
Fill EPn_TX register with:
- EPn_TX_VALID = 1
- EPn_TX_DB = 0
- EPn_TX_SIZE = EP size (2 bits)
- EPn_TX_ISO = 0
- EPn_TX_PTR = PTR_FLAG
PTR_FLAG = PTR_FLAG + EPn_TX_SIZE
End of endpoint configuration routine

32.3.9 Preparing for Transfers

To avoid NAK handshakes for the first transaction on an endpoint, the CPU must prepare the endpoint FIFO for receiving or transferring data. After the first transaction, the FIFO is enabled during the interrupt handling (ISR flowchart).

For receive endpoints, this phase consists of enabling the FIFO to receive data from the USB host. If double buffering is allowed for the endpoint, setting the CTRL.SET_FIFO_EN bit enables both FIFOs. Therefore, it is not possible to allow a single transaction when double buffering is used.

The CPU enters the prepare for USB RX transfers routine, presented once after the enumeration phase, and then properly reacts to EP interrupts. Whether double-buffering is allowed or not is transparent to the CPU, unless both FIFOs are cleared through a CTRL.CLR_EP or CTRL.RESET_EP. In that case, and in the case where the CPU finishes to handle an interrupt without having set the CTRL.SET_FIFO_EN bit, the CPU must reenter the prepare for USB RX transfers routine.

For transmit endpoints, the CPU enters the prepare for endpoint n TX transfer routine, presented each time a new file must be transmitted from endpoint n to USB host. The CPU must not enter this routine until data written into TX FIFO from the previous transfer have all been received successfully by the USB host (ACK interrupt received), unless TX FIFO is cleared through the CTRL.CLR_EP or CTRL.RESET_EP bits (see Figure 32-60 and Figure 32-61).

NOTE: This does not apply to endpoint 0, which is not used before a setup interrupt occurs. At setup interrupt, the CPU reacts appropriately, and enables EP0 FIFO only if necessary.

To ensure proper usage of the module, you cannot prepare data on different endpoints at the same time. You can enter a routine once the routine is not being used by another endpoint (no parallelism); the EP_NUM register can be accessed via different routines.

Figure 32-60. Prepare for USB RX Transfers Routine

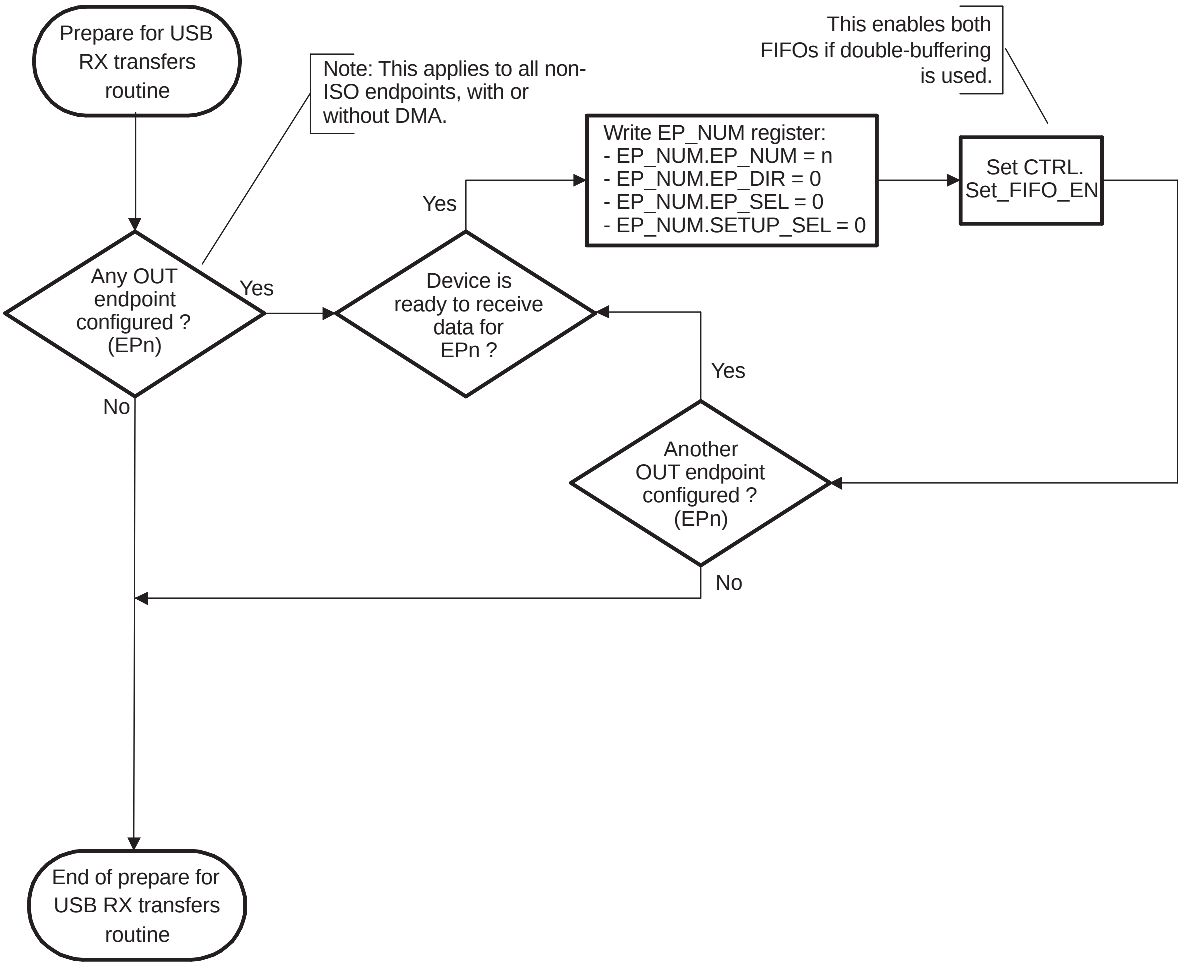
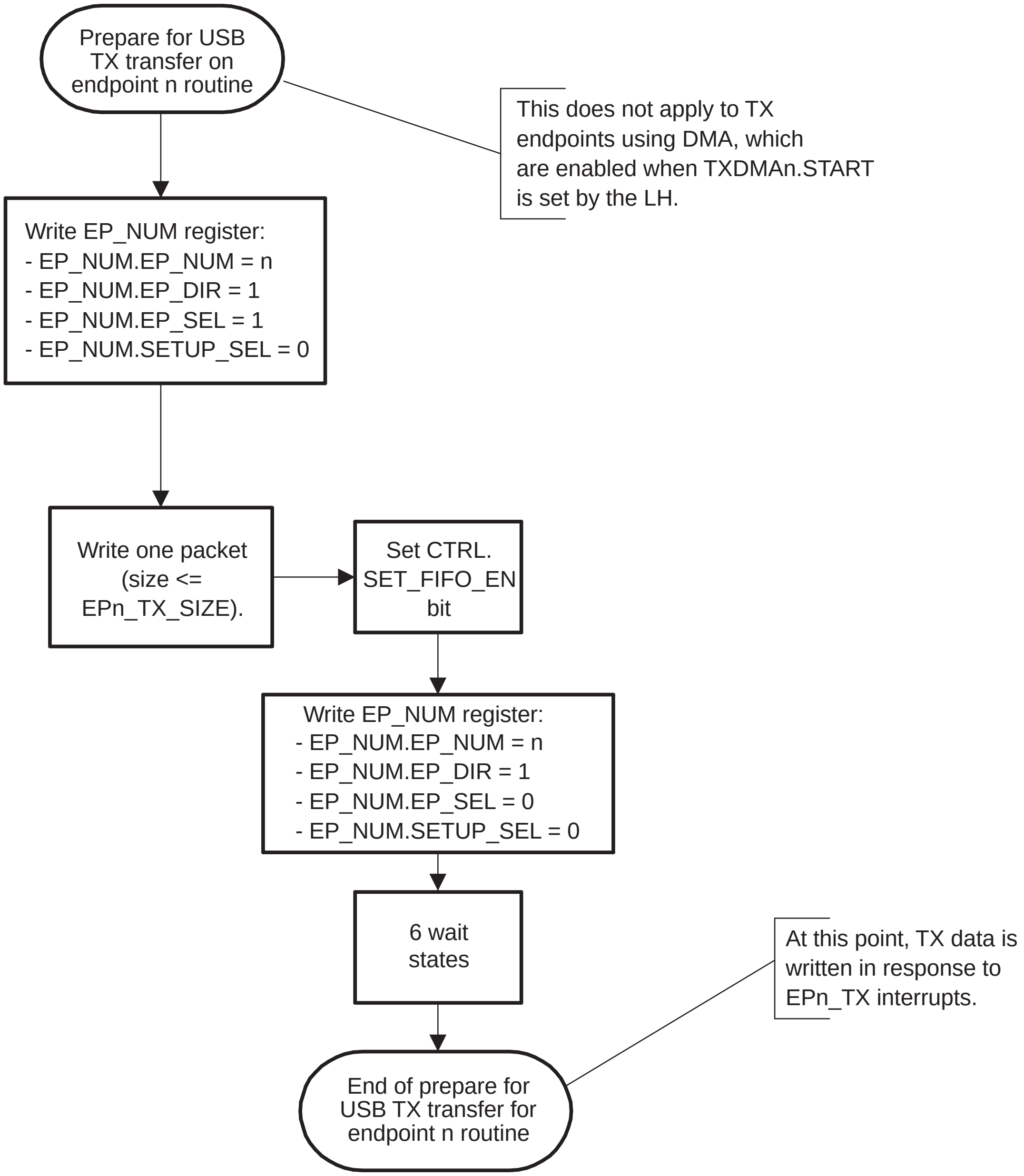

Prepare for USB RX transfers routine
Note: This applies to all non-ISO endpoints, with or without DMA.
This enables both FIFOs if double-buffering is used.
Write EP_NUM register:
- EP_NUM.EP_NUM = n
- EP_NUM.EP_DIR = 0
- EP_NUM.EP_SEL = 0
- EP_NUM.SETUP_SEL = 0
Set CTRL. Set_FIFO_EN
Yes
Any OUT endpoint configured ? (EPn)
Yes
Device is ready to receive data for EPn ?
Yes
No
Another OUT endpoint configured ? (EPn)
No
End of prepare for USB RX transfers routine

Figure 32-61. Prepare for TX Transfer on Endpoint n Routine


Prepare for USB TX transfer on endpoint n routine
This does not apply to TX endpoints using DMA, which are enabled when TXDMAn.START is set by the LH.
Write EP_NUM register:
- EP_NUM.EP_NUM = n
- EP_NUM.EP_DIR = 1
- EP_NUM.EP_SEL = 1
- EP_NUM.SETUP_SEL = 0
Write one packet (size <= EPn_TX_SIZE).
Set CTRL. SET_FIFO_EN bit
Write EP_NUM register:
- EP_NUM.EP_NUM = n
- EP_NUM.EP_DIR = 1
- EP_NUM.EP_SEL = 0
- EP_NUM.SETUP_SEL = 0
6 wait states
At this point, TX data is written in response to EPn_TX interrupts.
End of prepare for USB TX transfer for endpoint n routine

32.3.10 USB Device Interrupt Service Routine (ISR) Flowcharts

The flowcharts in this section give general operational guidelines for USB device ISR processing. System-architecture-specific details are left to the engineers who write the CPU and USB host code. One USB-specific interrupt register is provided (IRQ_SRC), including:

- General USB interrupts (including endpoint 0, DMA and device states interrupts) on Vectored Interrupt Manager (VIM) channel # 69 (by default)
- Non-ISO endpoint-specific interrupt on VIM channel # 70 (by default)
- Start of frame (SOF) interrupt for ISO transactions on VIM channel # 68 (by default)

The general USB interrupt ISR must handle non-autodecoded control transfers on endpoint 0 and some specialty interrupts generated because of USB device state modifications or DMA transfers. The ISR for the endpoint-specific interrupt must handle interrupts from the USB module that are generated because of USB activity for non-isochronous endpoints. The SOF ISR is responsible for handling isochronous endpoints and, if needed by the application, tracking the USB frame number. Many flowcharts are presented in this chapter to provide guidelines for how to handle the interrupts related to the USB device controller module. The flowcharts in this part suppose that the SYSCON1.NAK_EN bit is cleared.

NOTE: A key assumption behind the flowcharts presented here is that the application provides separate buffers for each direction of endpoint, except for endpoint 0. The flowcharts read from these application buffers for IN transactions on TX endpoints and write to these application buffers for OUT transactions on RX endpoints.

The USB device controller does not support reentrant interrupts. Each USB device controller must be handled completely before handling another USB device controller interrupt. This restriction occurs because there is only one EP_NUM register, so endpoint control operations must be completed before working with another endpoint or endpoint direction.

32.3.11 Important Note on USB Device Interrupts

When an endpoint interrupt is asserted, the CPU writes the EP_NUM register with the EP_NUM.EP_SEL bit set to 1. The CPU must finish the interrupt handling before clearing the EP_NUM.EP_SEL bit, because clearing this bit clears the corresponding status bit in the STAT_FLG register (ACK, NAK, STALL). When an interrupt is pending on an endpoint, the CPU must not select and then unselect the endpoint without handling the interrupt, because this clears the pending transaction status flags. The CPU does not need to set EP_NUM.EP_SEL to 1 when setting CTRL.SET_FIFO_EN, CTRL.SET_HALT, and CTRL.CLR_HALT bits.

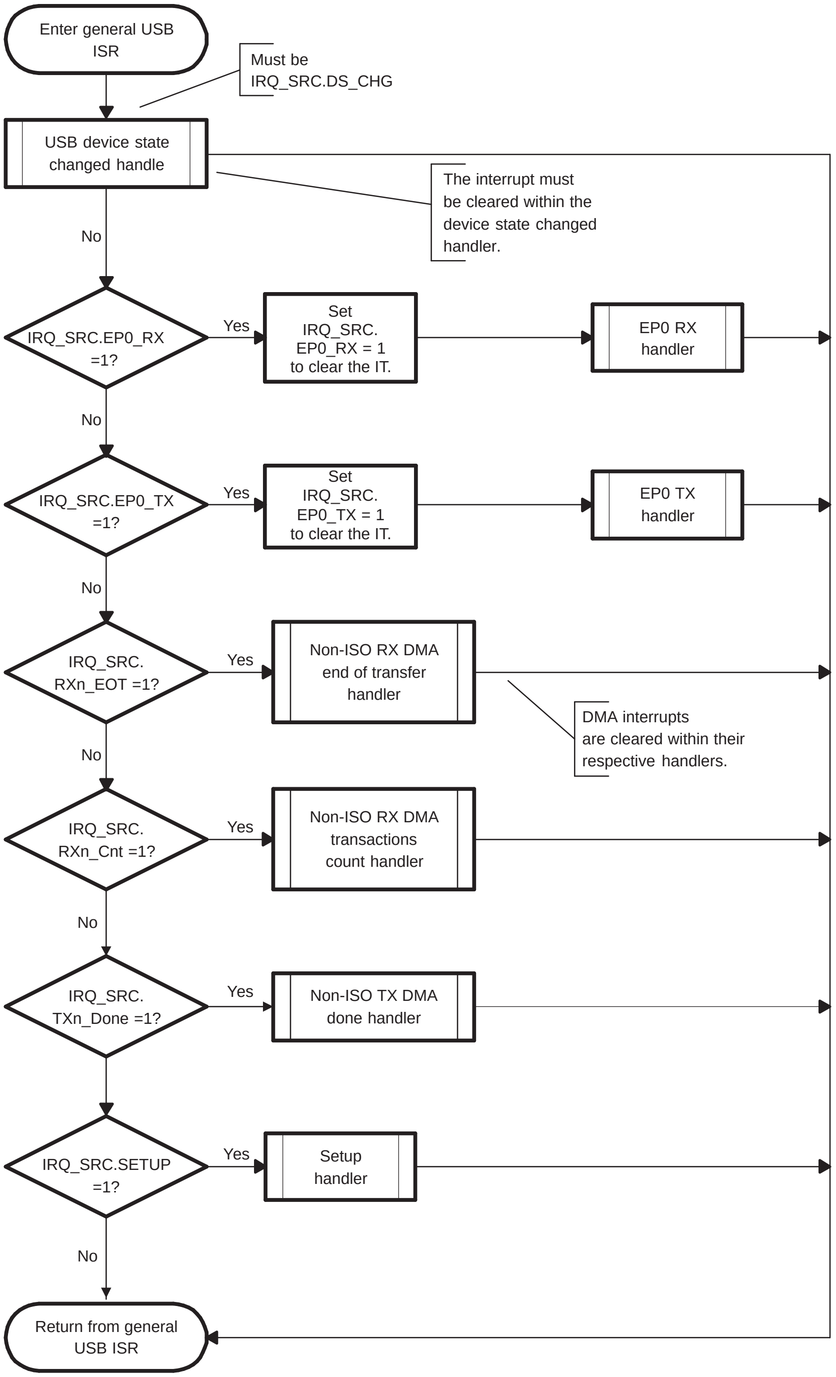
The endpoint status (STAT_FLG register) is updated at the end of each USB transaction if the previous transaction has been handled. If a pending interrupt has not been handled when a new non-transparent transaction occurs, status flags are not updated (and NAK is returned, even if FIFO was enabled, or STALLed if the EP halt feature was set), so that the CPU never misses an ACKed transaction. If double-buffering is used for an endpoint, STAT_FLG is updated if there is zero or one interrupt pending for the endpoint, and is not updated if there are already two interrupts pending on the endpoint.

The CPU does not need to set the SYSCON1.NAK_EN bit during normal operation. However, for debugging process, this bit can be set when the CPU finishes handling an EP interrupt without having set the corresponding CTRL.SET_FIFO_EN bit. During TX transaction, if the SYSCON1.NAK_EN bit is set, the CPU must wait for a NAK interrupt to write the TX data, to avoid a possible conflict caused by the NAK interrupt received while the CPU was writing the TX data.

32.3.12 Parsing General USB Device Interrupt

The general USB interrupt (VIM channel # 69 by default) ISR must parse the interrupt identifier register IRQ_SRC to determine the types of general USB interrupts that are active. These include interrupts relating to USB device state modifications (USB reset, suspend/resume, during enumeration phase) and control transfers on endpoint 0 or non-ISO DMA transfers in either receive or transmit mode. Multiple interrupts can be active at any time, and all must be dealt with by the ISR before returning from the ISR. Figure 32-62 shows an appropriate flowchart for parsing the general USB interrupts.

Figure 32-62. General USB Interrupt ISR Source Parsing Flowchart


Enter general USB ISR
Must be IRQ_SRC.DS_CHG
USB device state changed handle
The interrupt must be cleared within the device state changed handler.
No
IRQ_SRC.EP0_RX =1?
Yes
Set IRQ_SRC. EP0_RX = 1 to clear the IT.
EP0 RX handler
No
IRQ_SRC.EP0_TX =1?
Yes
Set IRQ_SRC. EP0_TX = 1 to clear the IT.
EP0 TX handler
No
IRQ_SRC. RXn_EOT =1?
Yes
Non-ISO RX DMA end of transfer handler
DMA interrupts are cleared within their respective handlers.
No
IRQ_SRC. RXn_Cnt =1?
Yes
Non-ISO RX DMA transactions count handler
No
IRQ_SRC. TXn_Done =1?
Yes
Non-ISO TX DMA done handler
IRQ_SRC.SETUP =1?
Yes
Setup handler
No
Return from general USB ISR

32.3.13 Setup Interrupt Handler

A separate interrupt flag exists for setup transactions so that the CPU cannot miss a setup transaction, even if it occurs during the data or status phase of another transfer (case of an aborted transfer). The setup parsing function captures the control transfer request information for use in determining which USB bus activity is needed and in controlling how the CPU must generate or respond to the control transfer. This information includes the following:

- bmRequestType
- bmRequest

- wValue
- wIndex
- wLength

The setup interrupt handler shown in Figure 32-63 is responsible for processing setup transactions occurring on endpoint 0. It calls the routine that parses the control transfer request information, shown in Figure 32-64 to set flags that the rest of the ISR code can use to control proper response to control transfers. Two flags are set by the setup interrupt handler, to be used during endpoint 0 interrupt handlers:

- Control read flag
- Control write flag

Figure 32-63. Setup Interrupt Handler

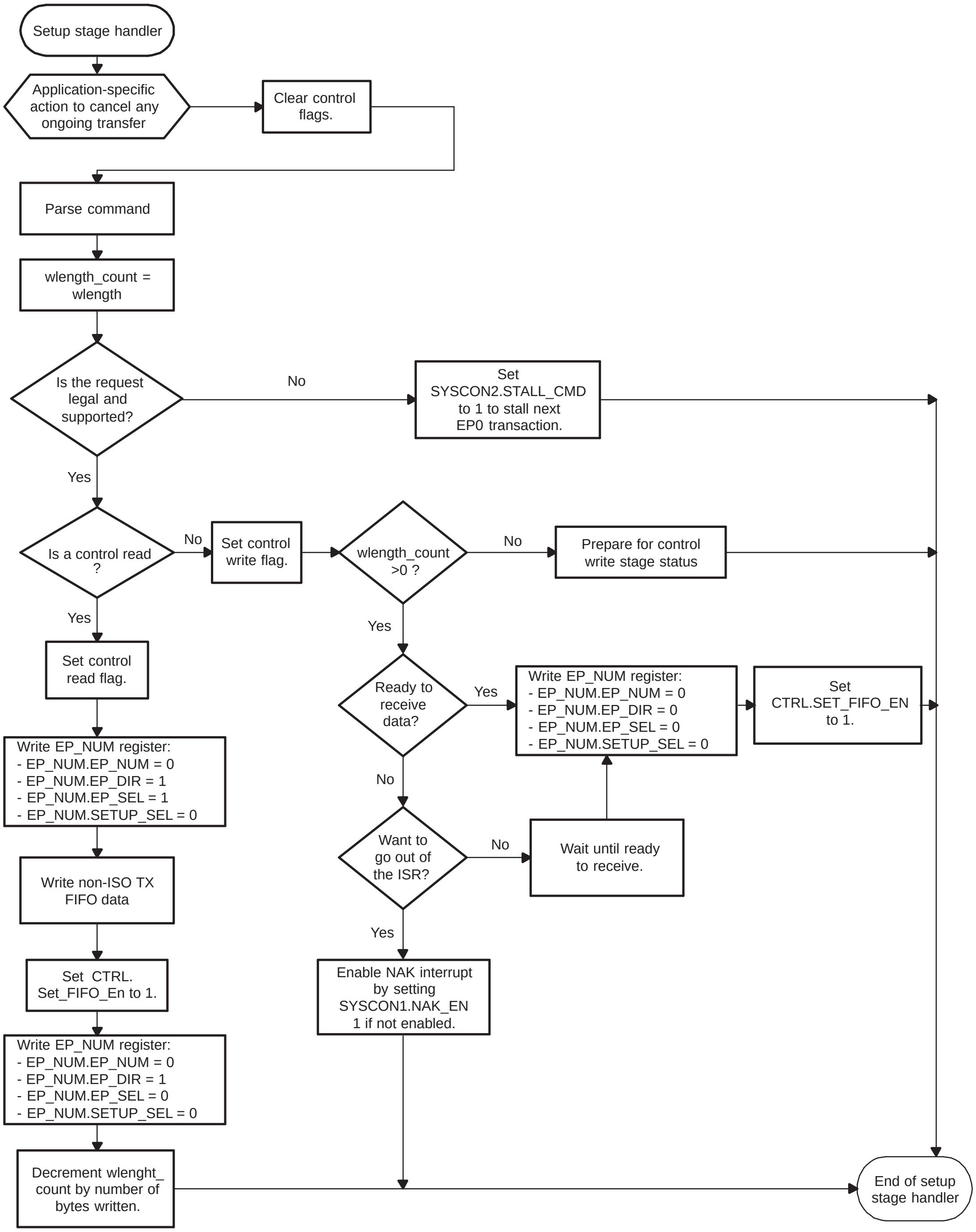
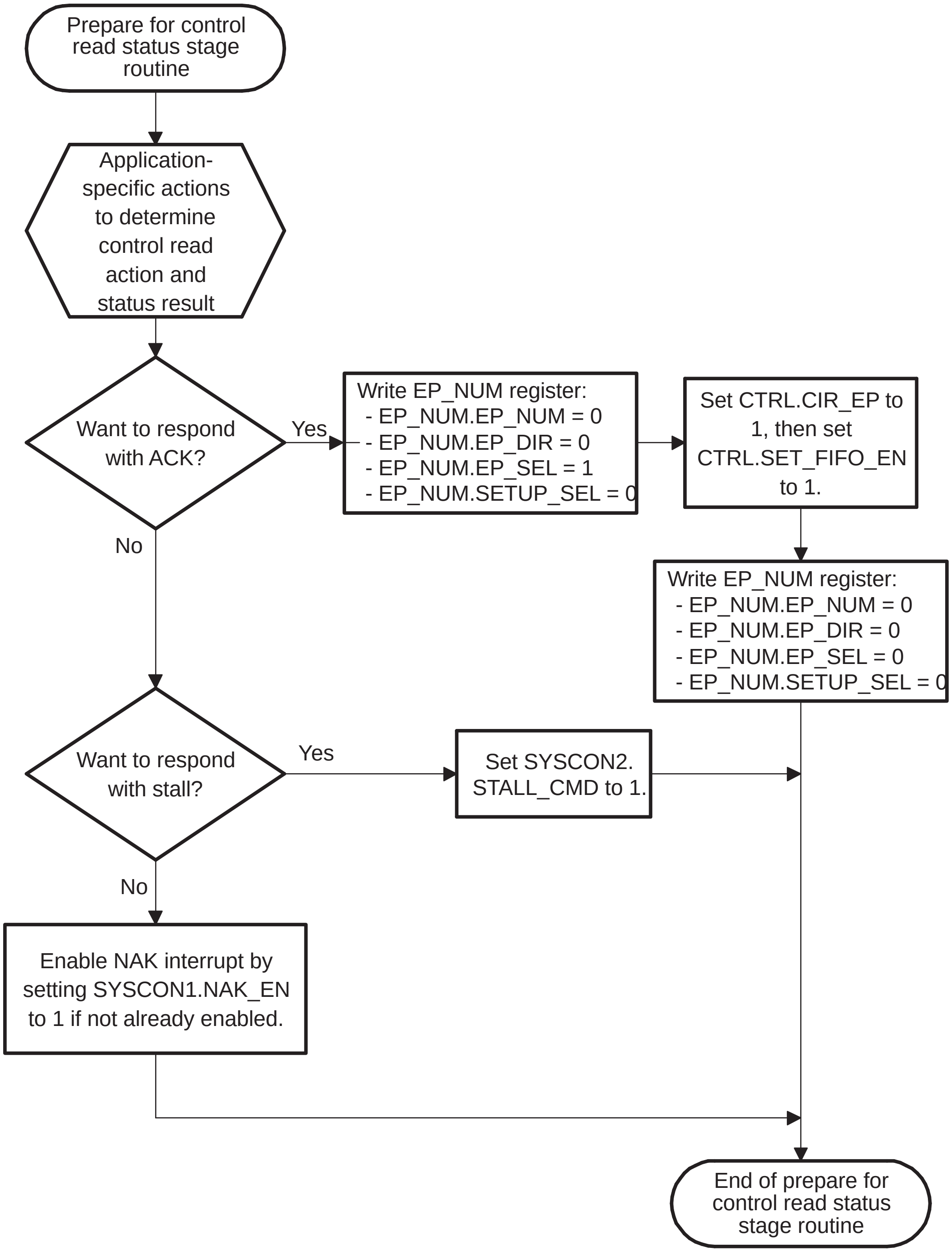


Figure 32-64. Parse Command Routine (Setup Stage Control Transfer Request)


Prepare for control read status stage routine
Application-specific actions to determine control read action and status result
Want to respond with ACK?
Yes
Write EP_NUM register:
- EP_NUM.EP_NUM = 0
- EP_NUM.EP_DIR = 0
- EP_NUM.EP_SEL = 1
- EP_NUM.SETUP_SEL = 0
Set CTRL.CIR_EP to 1, then set CTRL.SET_FIFO_EN to 1.
Write EP_NUM register:
- EP_NUM.EP_NUM = 0
- EP_NUM.EP_DIR = 0
- EP_NUM.EP_SEL = 0
- EP_NUM.SETUP_SEL = 0
No
Want to respond with stall?
Yes
Set SYSCON2. STALL_CMD to 1.
No
Enable NAK interrupt by setting SYSCON1.NAK_EN to 1 if not already enabled.
End of prepare for control read status stage routine

32.3.14 Endpoint 0 RX Interrupt Handler

Figure 32-65 shows the endpoint 0 RX portion of the general USB interrupt handler, which must handle general USB interrupts related to control OUT transactions on endpoint 0. No EP0 interrupt is generated for autodecoded control transfers.

Figure 32-65. Endpoint 0 RX Interrupt Handler

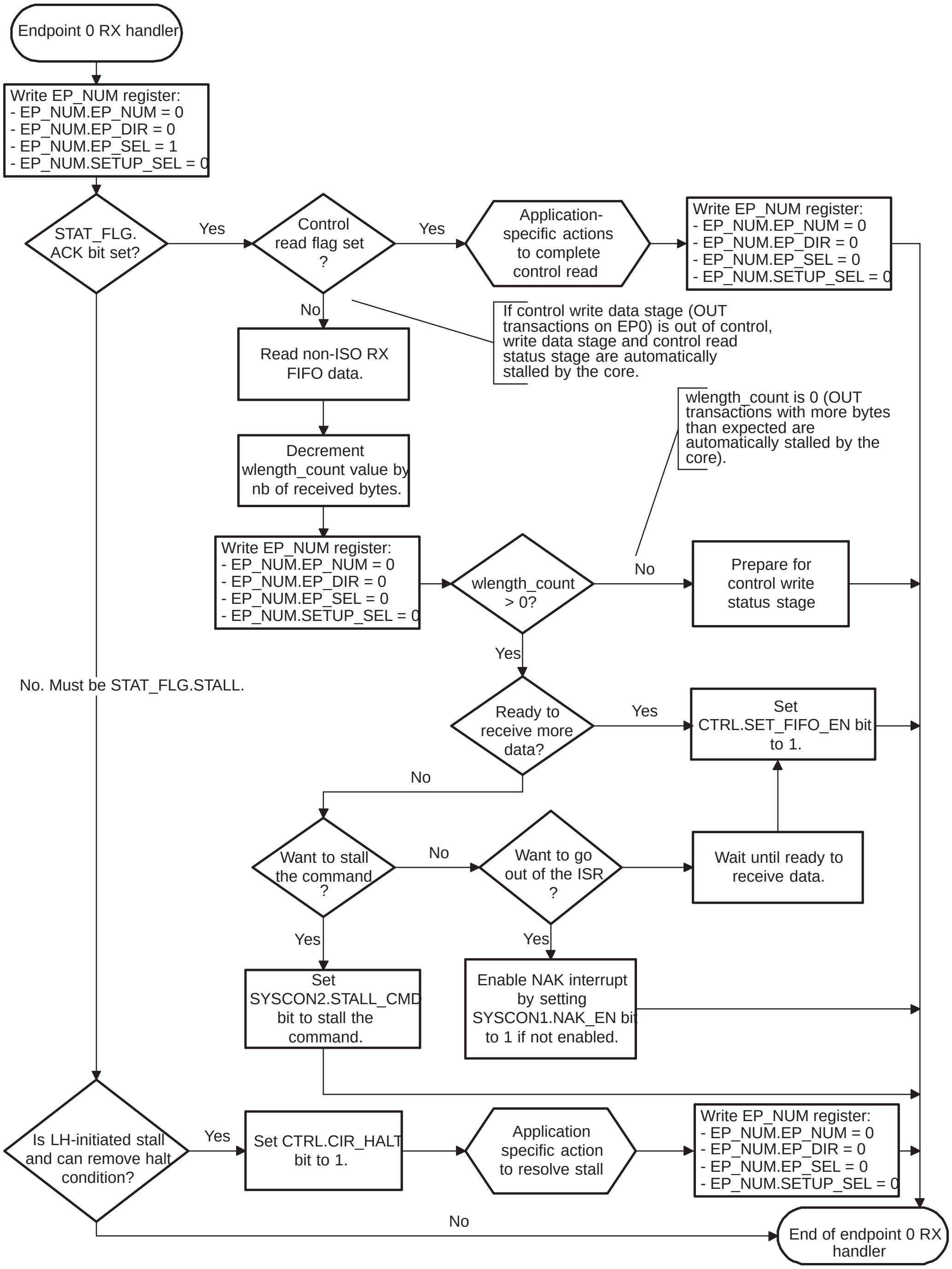
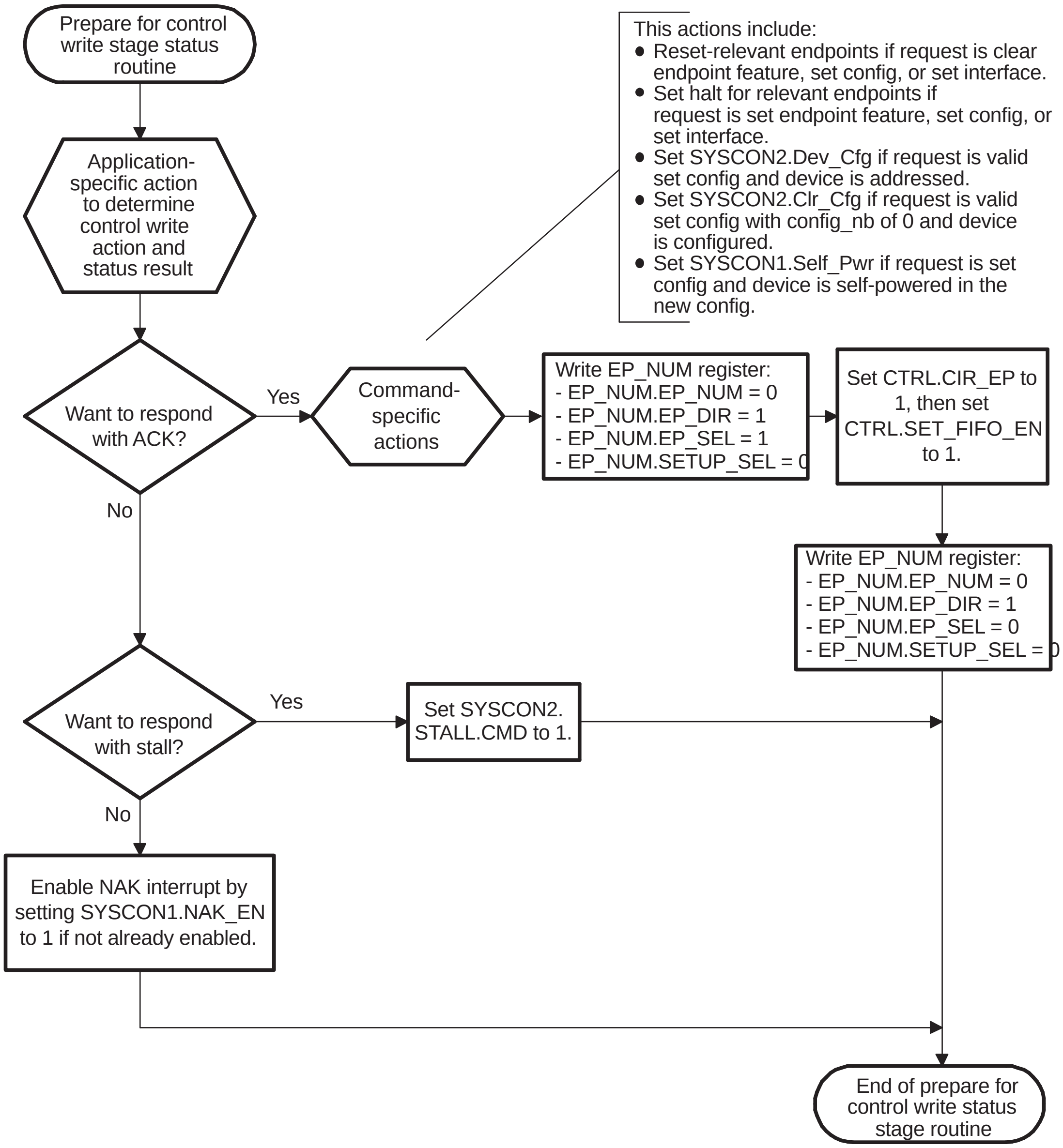

Endpoint 0 RX handler
Write EP_NUM register:
- EP_NUM.EP_NUM = 0
- EP_NUM.EP_DIR = 0
- EP_NUM.EP_SEL = 1
- EP_NUM.SETUP_SEL = 0
STAT_FLG. ACK bit set?
Yes
Control read flag set ?
Yes
Application-specific actions to complete control read
Write EP_NUM register:
- EP_NUM.EP_NUM = 0
- EP_NUM.EP_DIR = 0
- EP_NUM.EP_SEL = 0
- EP_NUM.SETUP_SEL = 0
No
Read non-ISO RX FIFO data.
If control write data stage (OUT transactions on EP0) is out of control, write data stage and control read status stage are automatically stalled by the core.
wlength_count is 0 (OUT transactions with more bytes than expected are automatically stalled by the core).
Decrement wlength_count value by nb of received bytes.
Write EP_NUM register:
- EP_NUM.EP_NUM = 0
- EP_NUM.EP_DIR = 0
- EP_NUM.EP_SEL = 0
- EP_NUM.SETUP_SEL = 0
wlength_count > 0?
No
Prepare for control write status stage
Yes
No. Must be STAT_FLG.STALL.
Ready to receive more data?
Yes
Set CTRL.SET_FIFO_EN bit to 1.
No
Want to stall the command ?
No
Want to go out of the ISR ?
Wait until ready to receive data.
Yes
Yes
Set SYSCON2.STALL_CMD bit to stall the command.
Enable NAK interrupt by setting SYSCON1.NAK_EN bit to 1 if not enabled.
Is LH-initiated stall and can remove halt condition?
Yes
Set CTRL.CIR_HALT bit to 1.
Application specific action to resolve stall
Write EP_NUM register:
- EP_NUM.EP_NUM = 0
- EP_NUM.EP_DIR = 0
- EP_NUM.EP_SEL = 0
- EP_NUM.SETUP_SEL = 0
No
End of endpoint 0 RX handler

Figure 32-66. Prepare for Control Write Status Stage Routine


Prepare for control write stage status routine
Application-specific action to determine control write action and status result
This actions include:
● Reset-relevant endpoints if request is clear endpoint feature, set config, or set interface.
● Set halt for relevant endpoints if request is set endpoint feature, set config, or set interface.
● Set SYSCON2.Dev_Cfg if request is valid set config and device is addressed.
● Set SYSCON2.Clr_Cfg if request is valid set config with config_nb of 0 and device is configured.
● Set SYSCON1.Self_Pwr if request is set config and device is self-powered in the new config.
Want to respond with ACK?
Yes
Command-specific actions
Write EP_NUM register:
- EP_NUM.EP_NUM = 0
- EP_NUM.EP_DIR = 1
- EP_NUM.EP_SEL = 1
- EP_NUM.SETUP_SEL = 0
Set CTRL.CIR_EP to 1, then set CTRL.SET_FIFO_EN to 1.
No
Write EP_NUM register:
- EP_NUM.EP_NUM = 0
- EP_NUM.EP_DIR = 1
- EP_NUM.EP_SEL = 0
- EP_NUM.SETUP_SEL = 0
Want to respond with stall?
Yes
Set SYSCON2. STALL.CMD to 1.
No
Enable NAK interrupt by setting SYSCON1.NAK_EN to 1 if not already enabled.
End of prepare for control write status stage routine

To support the SET_DESCRIPTOR command, when in commspecific actions, you must first reset all endpoints by selecting the endpoint and then clear it, unlock the configuration, charge the new one, lock the configuration, go in prepare for transfers, and enable all interrupts (see Figure 32-66).

32.3.15 Endpoint 0 TX Interrupt Handler

Figure 32-67 shows the TX portion of the general USB interrupt handler, which must handle general USB interrupts related to control IN transactions on endpoint 0.

The endpoint 0 TX interrupt handler must be able to move data into the endpoint 0 TX FIFO when the application buffer for endpoint 0 TX data is not empty and an endpoint 0 TX interrupt occurs signaling an ACKed non-autodecoded endpoint 0 IN transaction. This data can be control read data stage information or control write status stage handshaking information (see Figure 32-68).

Figure 32-67. Endpoint 0 TX Interrupt Handler

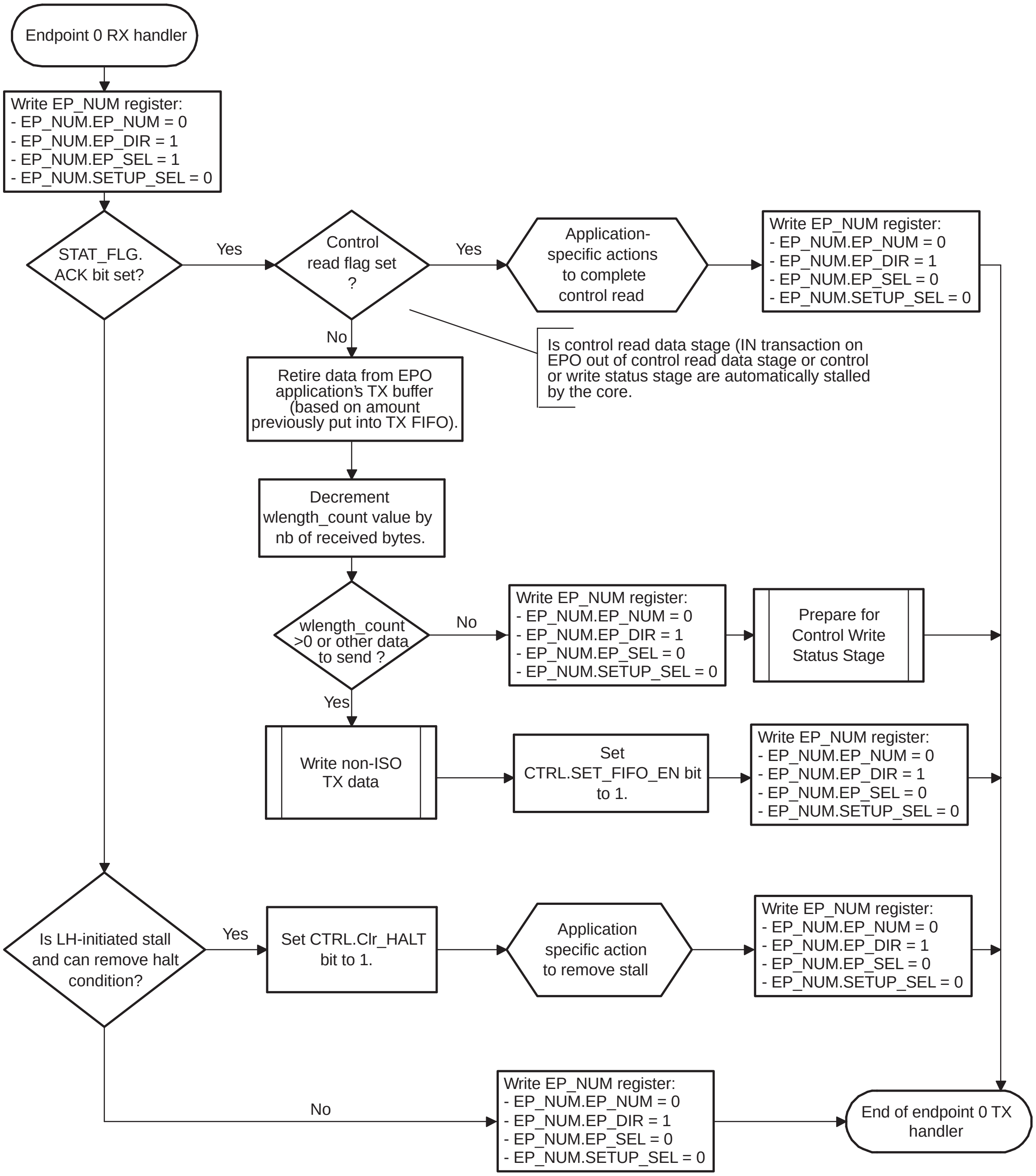
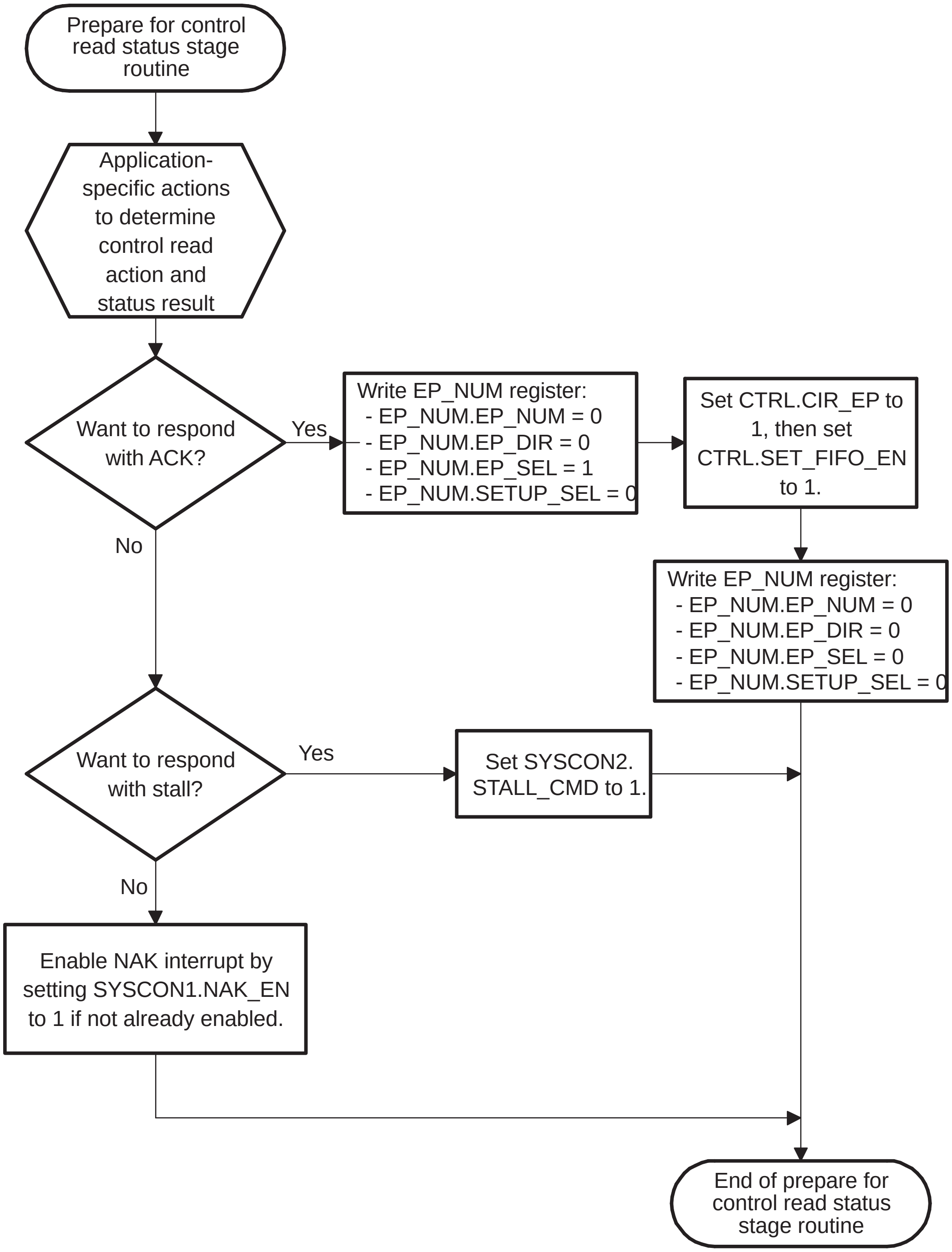


Figure 32-68. Prepare for Control Read Status Stage Routine


Prepare for control read status stage routine
Application-specific actions to determine control read action and status result
Want to respond with ACK?
Yes
Write EP_NUM register:
- EP_NUM.EP_NUM = 0
- EP_NUM.EP_DIR = 0
- EP_NUM.EP_SEL = 1
- EP_NUM.SETUP_SEL = 0
Set CTRL.CIR_EP to 1, then set CTRL.SET_FIFO_EN to 1.
Write EP_NUM register:
- EP_NUM.EP_NUM = 0
- EP_NUM.EP_DIR = 0
- EP_NUM.EP_SEL = 0
- EP_NUM.SETUP_SEL = 0
No
Want to respond with stall?
Yes
Set SYSCON2. STALL_CMD to 1.
No
Enable NAK interrupt by setting SYSCON1.NAK_EN to 1 if not already enabled.
End of prepare for control read status stage routine

32.3.16 Device States Changed Handler

This section describes how USB device states and transitions states are decoded by the USB device controller and how they can be handled.

The state machine (see Figure 32-69) shows how the USB device controller device moves from one state to another state with respect to the USB1.1 specification. Attach/unattach transition is not shown in the transition flow.

Because SET_CONFIGURATION is not decoded by the core, the CPU has the responsibility to distinguish a SET_CONFIGURATION with a nonvalid configuration value from other SET_CONFIGURATION requests and to set SYSCON2.DEV_CFG only if the configuration value is valid (value 0 is nonvalid), when the device is in addressed state. When the device is in configured state, the CPU has the responsibility to set SYSCON2.CLR_CFG if the configuration number is 0 so that the device moves to addressed state.

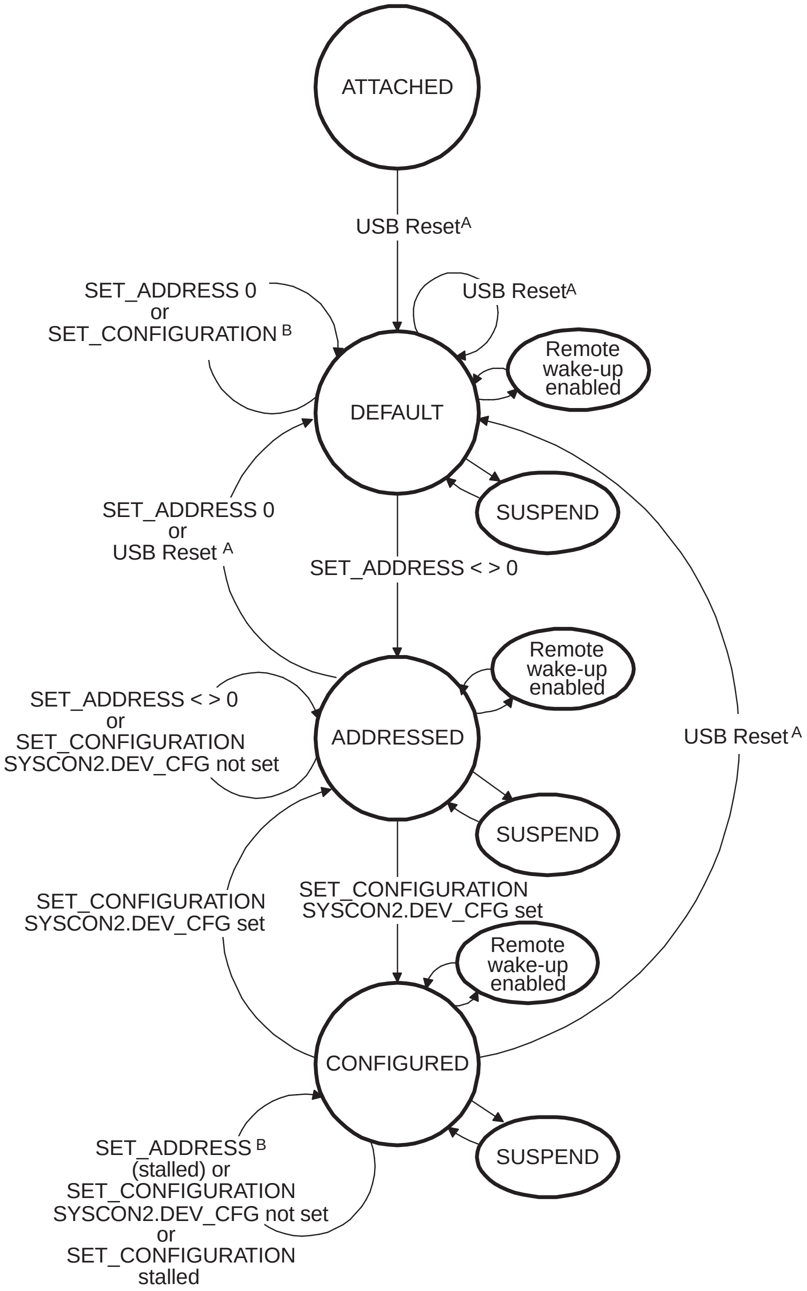
Device states are visible in the DEVSTAT register and are decoded as follows.

- Attached: The device is attached to the USB and powered.
- Default: The device is attached to the USB, is powered, and has been reset.
- Addressed: The device is attached to the USB, is powered, has been reset, and an address has been assigned. The device moves into the addressed state after a SET_ADDRESS request with an address number other than 0.
- Configured: The device is attached to the USB, is powered, has been reset, has an address other than 0, and is configured. The device moves into the configured state after a valid SET_CONFIGURATION request, only if the CPU has set the SYSCON2.DEV_CFG bit (meaning the configuration is valid).
- Suspended: The device is at minimum default and has not seen bus activity for 5 ms.
- Reset: When set, the device is receiving a valid USB host reset.
- R_WK_OK: This bit is set (cleared) automatically after a valid SET_DEVICE_FEATURE (CLEAR_DEVICE_FEATURE) request from the USB host.

Any change in the DEVSTAT register bits triggers a device change interrupt (IRQ_SRC.DS_CHG), if enabled.

The device moves to addressed state after the status stage of a valid SET_ADDRESS, even if the status stage ACK handshake is received corrupted or not sent by the USB host. A SET_DEVICE_FEATURE or a CLEAR_DEVICE_FEATURE is effective after setup transaction, even if no status stage occurs. A SET_CONFIGURATION request is effective before status stage, when the CPU sets the SYSCON2.CLR_CFG or SYSCON2.DEV_CFG bit (see Figure 32-70).

Figure 32-69. USB Device Controller Device State Transitions

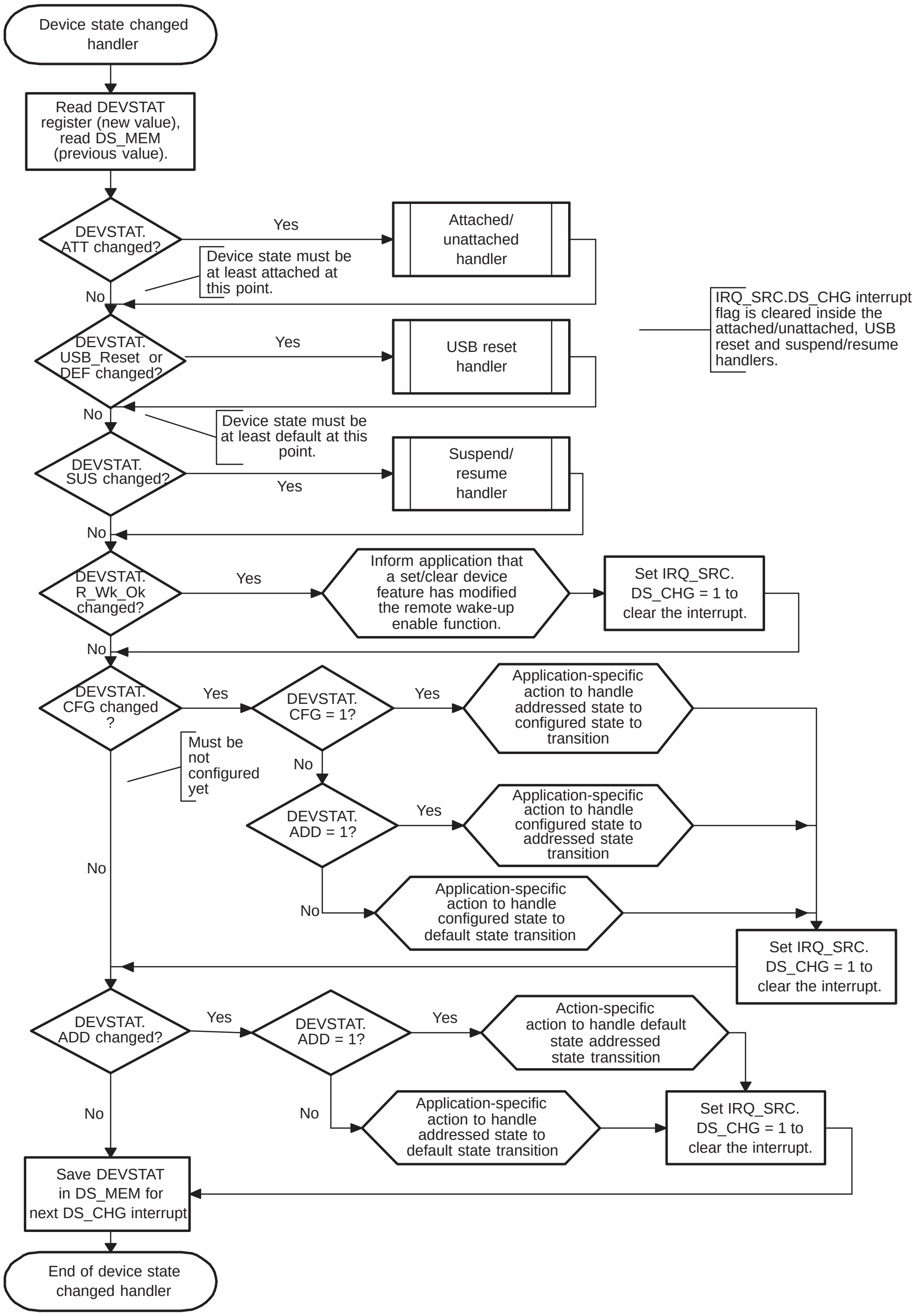


Behavior not specified by USB 1.1 specifications

A USB reset generates two interrupts (when USB reset is asserted and then when USB reset completes.)

B No interrupt is asserted by the core for transitions shown with dashed lines.

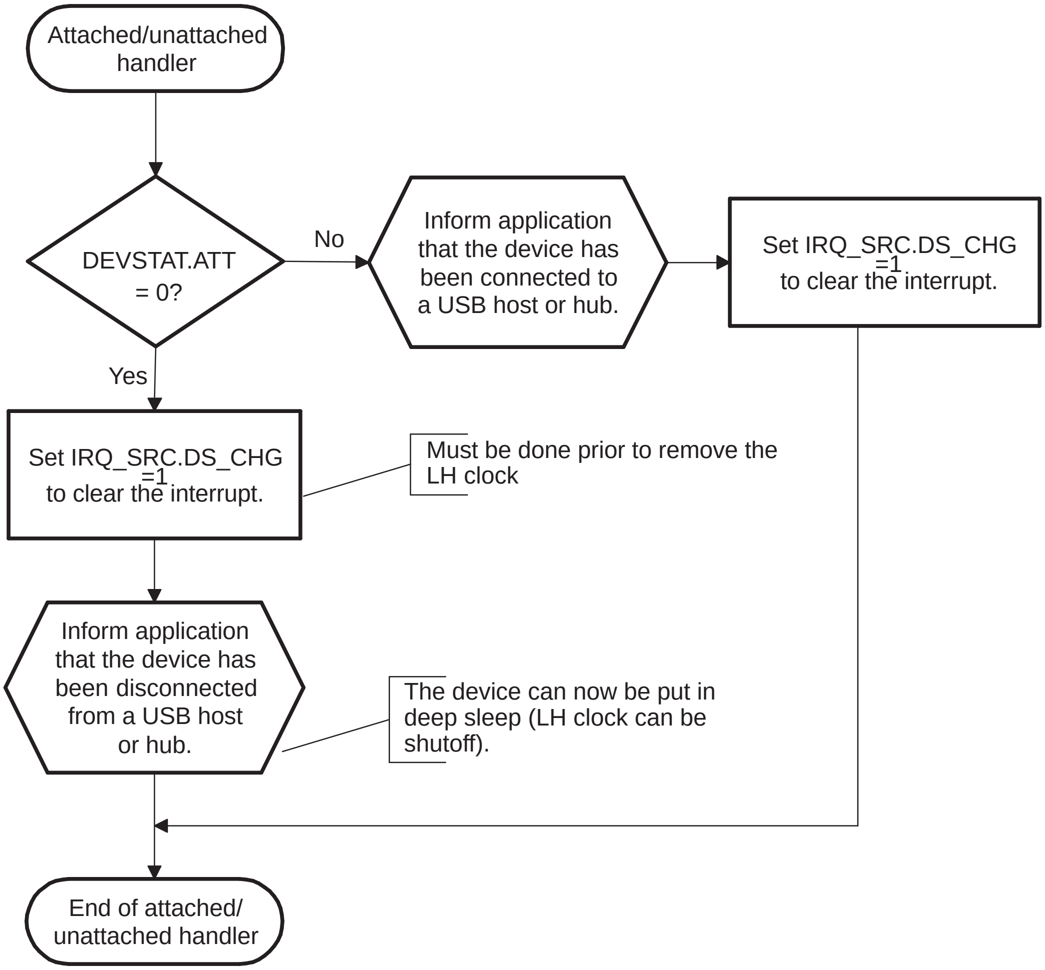
Figure 32-70. Typical Operation for USB Device State Changed Interrupt Handler


Device state changed handler
Read DEVSTAT register (new value), read DS_MEM (previous value).
DEVSTAT. ATT changed?
Yes
Attached/ unattached handler
Device state must be at least attached at this point.
No
DEVSTAT. USB_Reset or DEF changed?
Yes
USB reset handler
IRQ_SRC.DS_CHG interrupt flag is cleared inside the attached/unattached, USB reset and suspend/resume handlers.
No
Device state must be at least default at this point.
DEVSTAT. SUS changed?
Yes
Suspend/ resume handler
No
DEVSTAT. R_Wk_Ok changed?
Yes
Inform application that a set/clear device feature has modified the remote wake-up enable function.
Set IRQ_SRC. DS_CHG = 1 to clear the interrupt.
No
DEVSTAT. CFG changed ?
Yes
DEVSTAT. CFG = 1?
Yes
Application-specific action to handle addressed state to configured state to transition
Must be not configured yet
No
DEVSTAT. ADD = 1?
Yes
Application-specific action to handle configured state to addressed state transition
No
No
Application-specific action to handle configured state to default state transition
Set IRQ_SRC. DS_CHG = 1 to clear the interrupt.
DEVSTAT. ADD changed?
Yes
DEVSTAT. ADD = 1?
Yes
Action-specific action to handle default state addressed state transsition
No
No
Application-specific action to handle addressed state to default state transition
Set IRQ_SRC. DS_CHG = 1 to clear the interrupt.
Save DEVSTAT in DS_MEM for next DS_CHG interrupt
End of device state changed handler

32.3.17 Device States Attached/Unattached Handler

Device attached/unattached interrupts occur when the device detects that its VBUS has changed. System software can disable the USB device controller clock after IRQ_SRC.DS_CHG is cleared after servicing an unattached interrupt. Disabling the USB device controller clock before IRQ_SRC.DS_CHG is cleared can result in improper functionality for future USB device controller interrupts (see Figure 32-71).

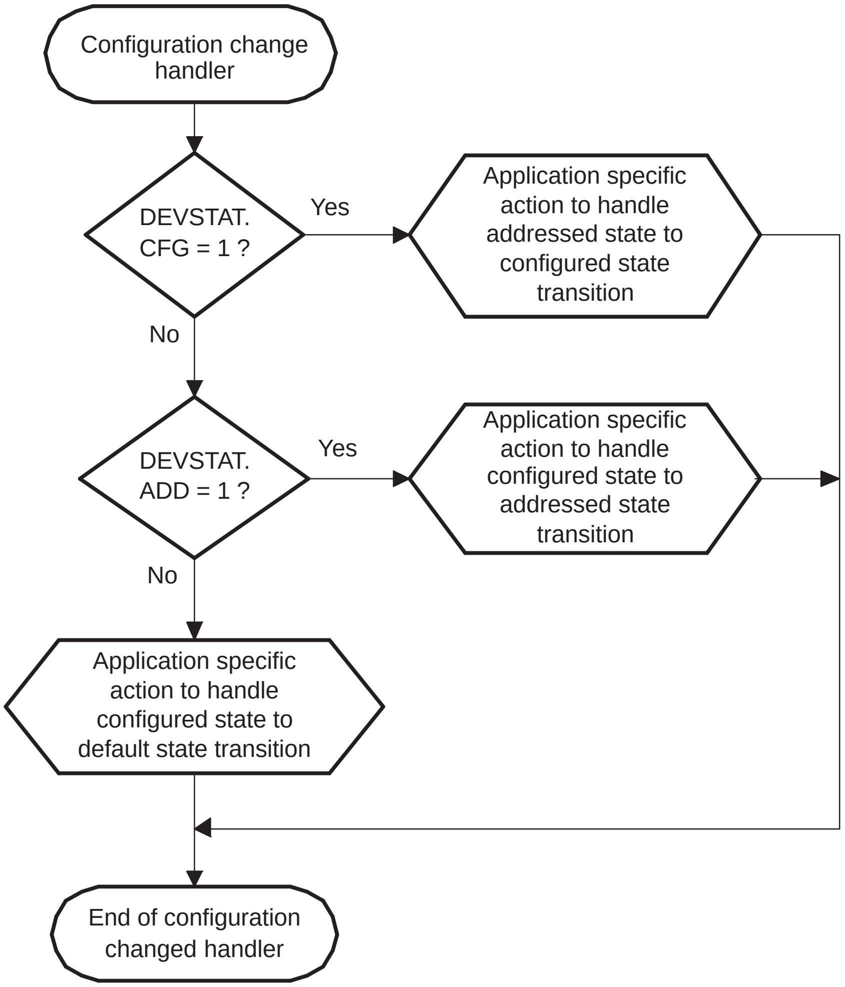
Figure 32-71. Attached/Unattached Handler


Attached/unattached handler
DEVSTAT.ATT = 0?
No
Inform application that the device has been connected to a USB host or hub.
Set IRQ_SRC.DS_CHG =1 to clear the interrupt.
Yes
Set IRQ_SRC.DS_CHG =1 to clear the interrupt.
Must be done prior to remove the LH clock
Inform application that the device has been disconnected from a USB host or hub.
The device can now be put in deep sleep (LH clock can be shutoff).
End of attached/ unattached handler

32.3.18 Device State Configuration Changed Handler

When a configuration changed interrupt occurs, the USB device has received a set configuration operation. When this occurs, the configuration-changed handler performs the operations shown in Figure 32-72.

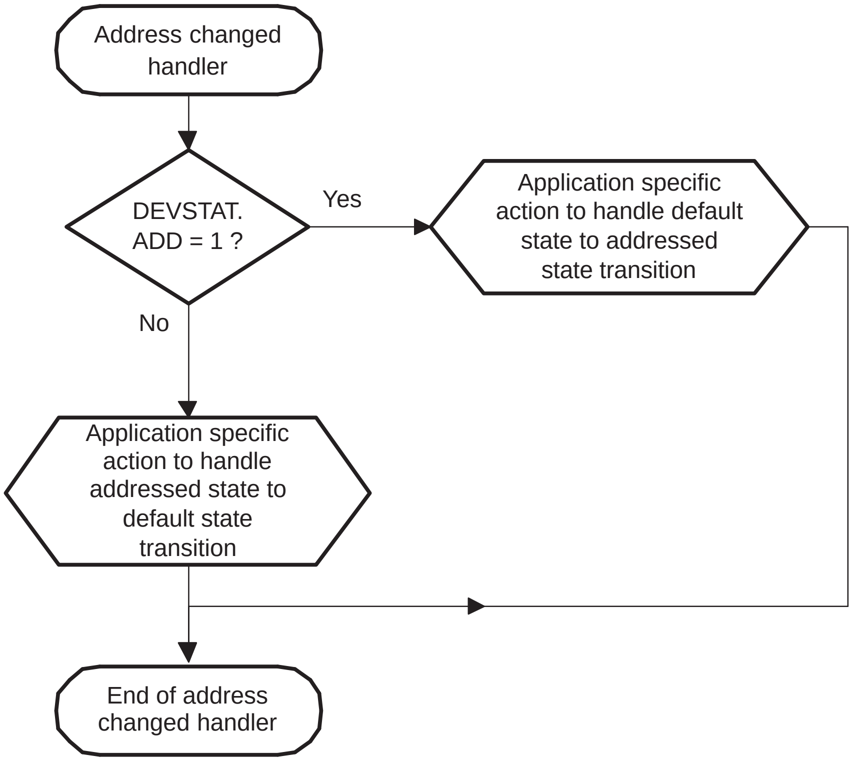
Figure 32-72. Configuration Changed Handler


Configuration change handler
DEVSTAT. CFG = 1 ?
Yes
Application specific action to handle addressed state to configured state transition
No
DEVSTAT. ADD = 1 ?
Yes
Application specific action to handle configured state to addressed state transition
No
Application specific action to handle configured state to default state transition
End of configuration changed handler

32.3.19 Device State Address Changed Handler

When a address changed interrupt occurs, the USB device has received a set address operation. When this occurs, the address-changed handler performs the operations shown in Figure 32-73.

Figure 32-73. Address Changed Handler

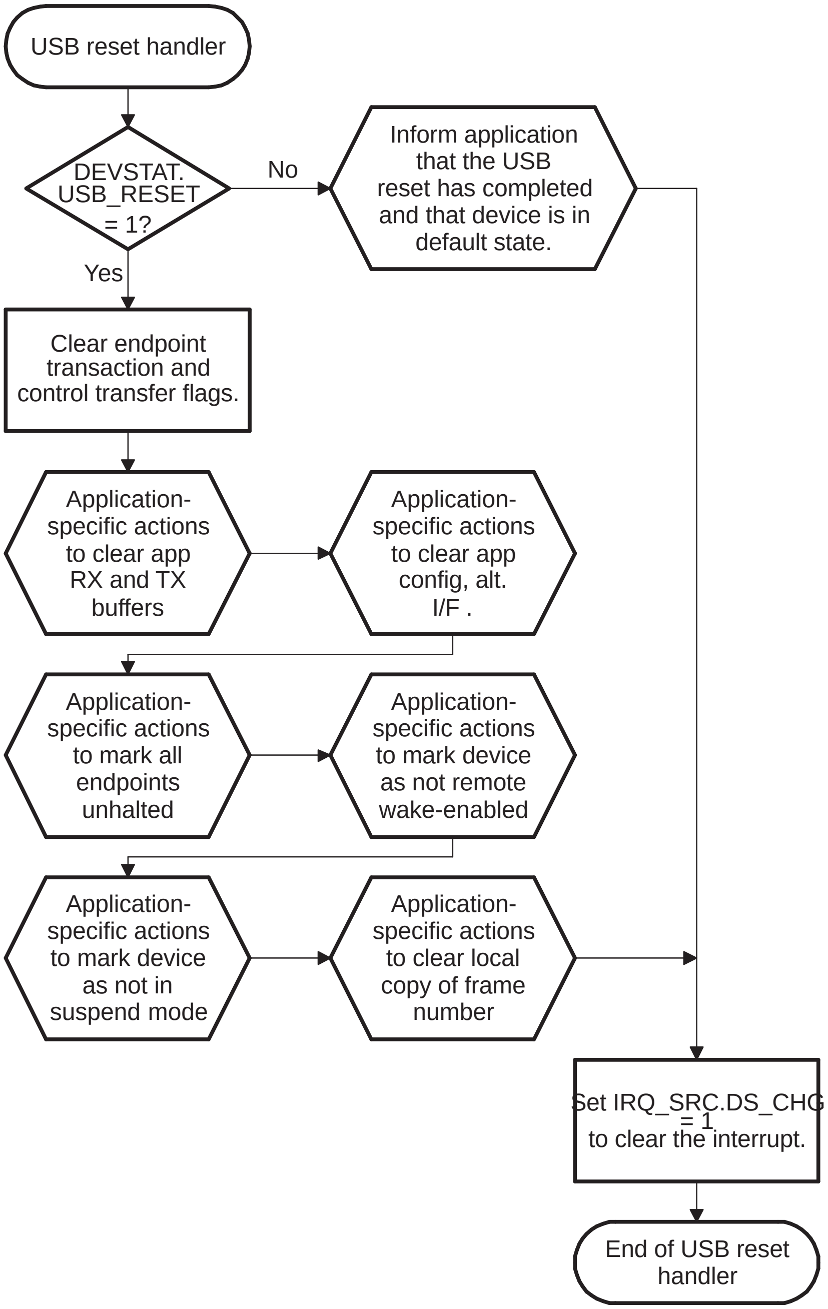

Address changed handler
DEVSTAT. ADD = 1 ?
Yes
Application specific action to handle default state to addressed state transition
No
Application specific action to handle addressed state to default state transition
End of address changed handler

32.3.20 USB Device Reset Interrupt Handler

When a USB reset occurs, the USB module generates a general USB interrupt to the CPU (see Figure 32-74). The CPU responds to this interrupt by performing the following operations:

- Cancels any ongoing USB transaction and/or control transfer handling
- Clears any copies that the application has of configuration number or alternate interface numbers
- Clears any application-specific information relating to halted endpoints
- Clears any application-specific information relating to the remote wake enable flag
- Clears any application-specific information relating to the suspend mode flag
- Clears any application-specific copy of the frame number

Figure 32-74. USB Device Reset Handler Flowchart


USB reset handler
DEVSTAT. USB_RESET = 1?
No
Yes
Inform application that the USB reset has completed and that device is in default state.
Clear endpoint transaction and control transfer flags.
Application-specific actions to clear app RX and TX buffers
Application-specific actions to clear app config, alt. I/F .
Application-specific actions to mark all endpoints unhalted
Application-specific actions to mark device as not remote wake-enabled
Application-specific actions to mark device as not in suspend mode
Application-specific actions to clear local copy of frame number
Set IRQ_SRC.DS_CHG = 1 to clear the interrupt.
End of USB reset handler

32.3.21 Suspend/Resume Interrupt Handler

When a USB device suspend/resume general USB interrupt occurs, the USB module has either entered or left suspend mode. The CPU code must determine which and react appropriately (see Figure 32-75).

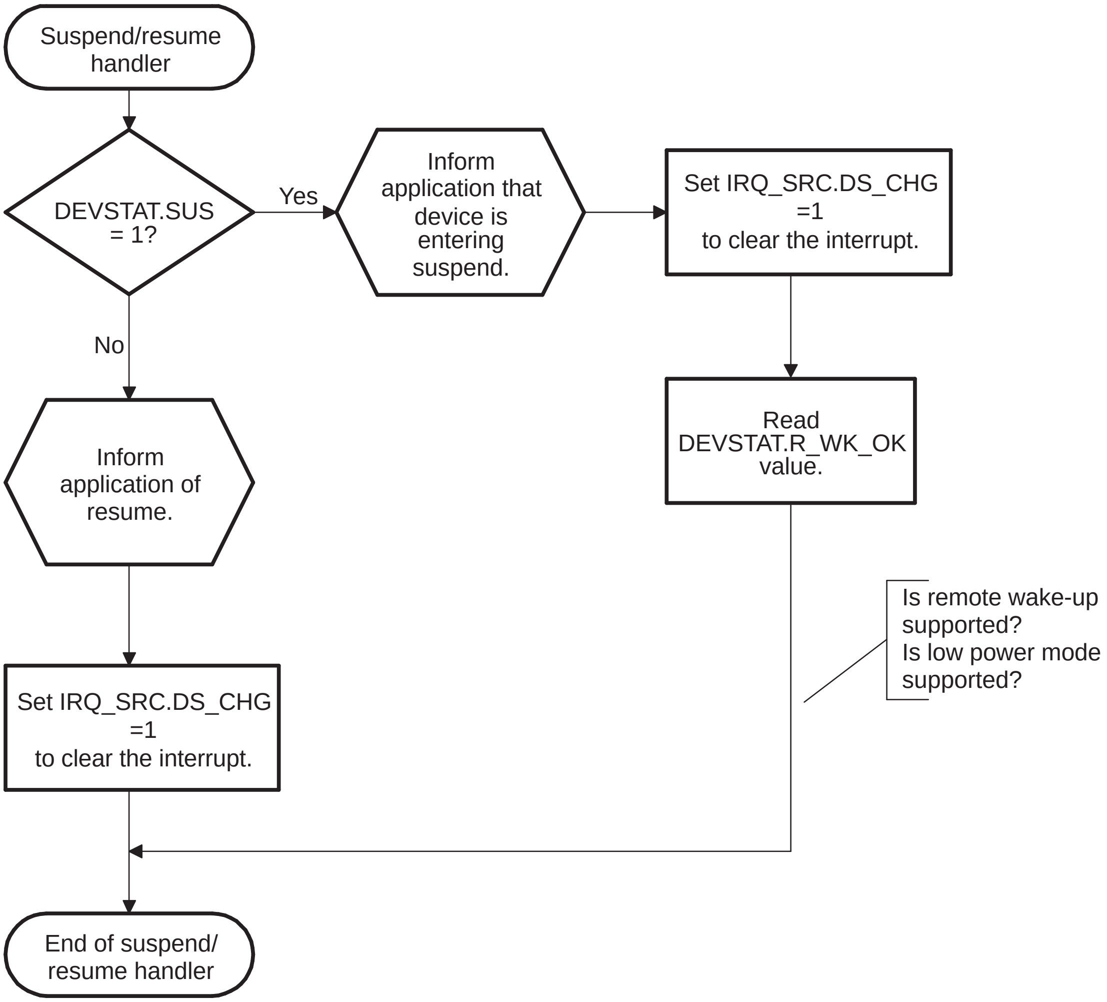
The suspend sense hardware is implemented to trigger only after 5 ms of bus idle. This forces compliance with the *USB Specification Version 1.1* T_{WTRSM} timing parameter (3 ms of IDLE to identify suspend, 2 ms before the remote device can signal resume).

If the CPU wants to wake the device from suspend mode and remote wake-up enable is set (DEVSTAT.R_WK_OK = 1), it must first turn its clock on (if stopped), and then set SYSCON2.RMT_WKP. The device then drives resume.

If shutoff is enabled (SYSCON1.SOFF_DIS=0), the 48-MHz clock is automatically shut off at suspend and turned on at resume (USB host or CPU driven). Setting or not setting the SYSCON1.SOFF_DIS bit is part of the device configuration. However, the CPU can modify its value at suspend interrupt time if necessary.

A USB reset is also a valid way to exit suspend mode. But the suspend/resume handler and the USB reset handler do not have to take this into account, because three interrupts are generated in that case (one for resume, one for reset, and one for end of reset).

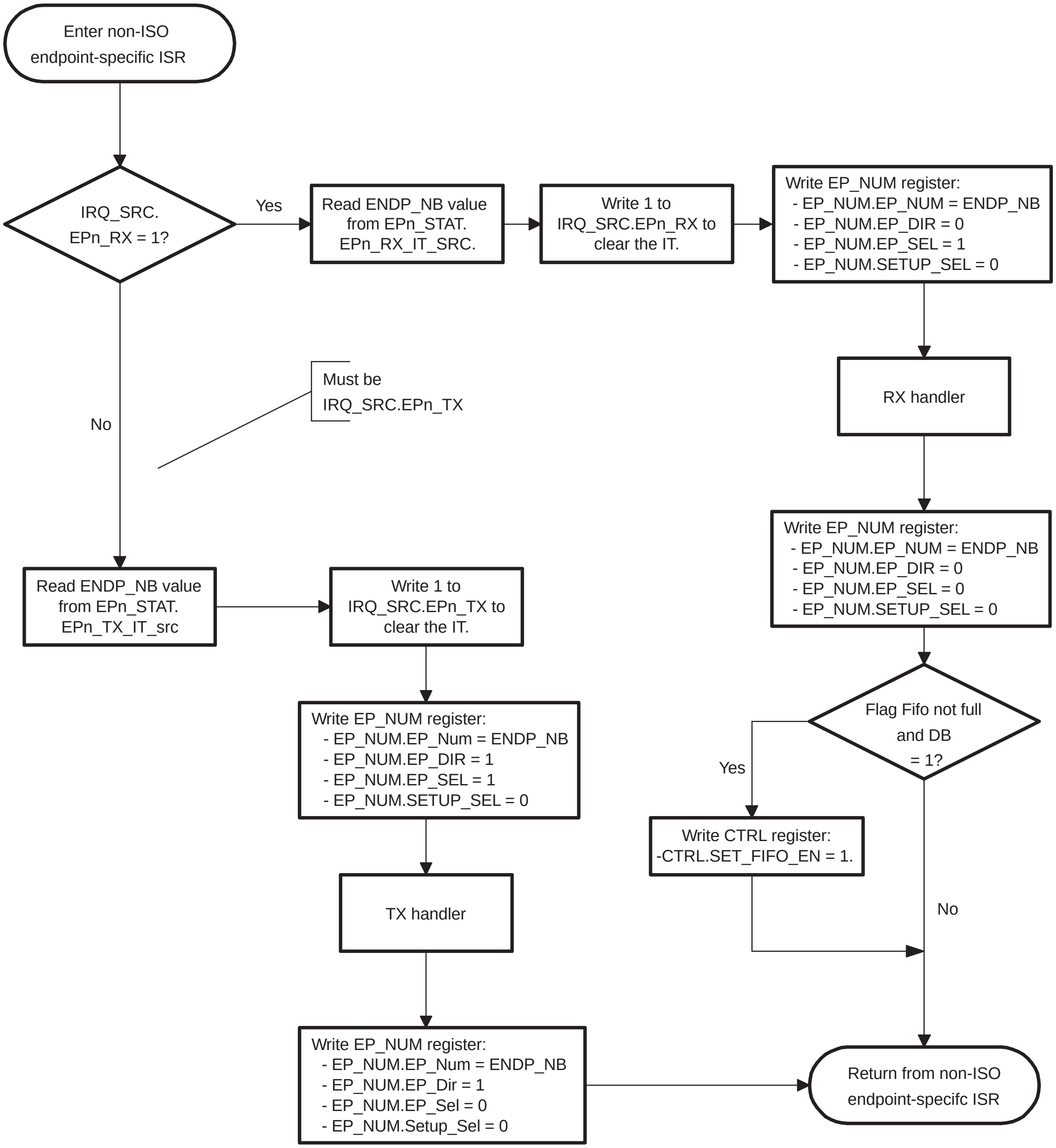
Figure 32-75. Typical Operation for USB Suspend/Resume General USB Interrupt Handler


Suspend/resume handler
DEVSTAT.SUS = 1?
Yes
Inform application that device is entering suspend.
Set IRQ_SRC.DS_CHG =1 to clear the interrupt.
No
Read DEVSTAT.R_WK_OK value.
Inform application of resume.
Is remote wake-up supported?
Is low power mode supported?
Set IRQ_SRC.DS_CHG =1 to clear the interrupt.
End of suspend/ resume handler

32.3.22 Parsing Non-ISO Endpoint-Specific Interrupt

The endpoint-specific interrupt ISR (also known as the non-isochronous interrupt, on VIM channel # 70 by default) must parse the interrupt identifier register IRQ_SRC to determine the interrupts that are active (IRQ_SRC.EPN_RX, IRQ_SRC.EPN_TX, or both). Two interrupts can be active at any time. The ISR must then read the EPN_STAT register to determine the endpoint causing the interrupt. For each direction, only one endpoint interrupt can be active at a time (see Figure 32-76).

Figure 32-76. Non-ISO Endpoint-Specific (Except EP 0) ISR Flowchart


Enter non-ISO endpoint-specific ISR
IRQ_SRC. EPn_RX = 1?
Yes
Read ENDP_NB value from EPn_STAT. EPn_RX_IT_SRC.
Write 1 to IRQ_SRC.EPn_RX to clear the IT.
Write EP_NUM register:
- EP_NUM.EP_NUM = ENDP_NB
- EP_NUM.EP_DIR = 0
- EP_NUM.EP_SEL = 1
- EP_NUM.SETUP_SEL = 0
Must be IRQ_SRC.EPn_TX
No
RX handler
Write EP_NUM register:
- EP_NUM.EP_NUM = ENDP_NB
- EP_NUM.EP_DIR = 0
- EP_NUM.EP_SEL = 0
- EP_NUM.SETUP_SEL = 0
Read ENDP_NB value from EPn_STAT. EPn_TX_IT_src
Write 1 to IRQ_SRC.EPn_TX to clear the IT.
Write EP_NUM register:
- EP_NUM.EP_Num = ENDP_NB
- EP_NUM.EP_DIR = 1
- EP_NUM.EP_SEL = 1
- EP_NUM.SETUP_SEL = 0
Flag Fifo not full and DB = 1?
Yes
Write CTRL register: -CTRL.SET_FIFO_EN = 1.
No
TX handler
Write EP_NUM register:
- EP_NUM.EP_Num = ENDP_NB
- EP_NUM.EP_Dir = 1
- EP_NUM.EP_Sel = 0
- EP_NUM.Setup_Sel = 0
Return from non-ISO endpoint-specifc ISR

32.3.23 Non-ISO, Non-Control OUT Endpoint Receive Interrupt Handler

Figure 32-77 shows the operations necessary to handle non-ISO, non-control OUT endpoint-specific receive interrupts. This flowchart shows two different RX transaction handshaking interrupts. There is a third interrupt handshaking possibility when NAK interrupts are enabled, which is not depicted here. Depending on the application-specific actions needed for various endpoints in the real system, it is possible to use one routine that is common to all of the non-ISO, non-control receive endpoints, where the only differences are in the EP_NUM register value set and the selection of the proper application RX data buffer in the read non-ISO RX FIFO data routine (see Figure 32-78).

This flowchart does not attempt to document control endpoint 0 receive interrupts, which are discussed separately because of the more complex three-stage transfer mechanism used for control writes.

Figure 32-77. Non-Isochronous Non-Control Endpoint Receive Interrupt Handler

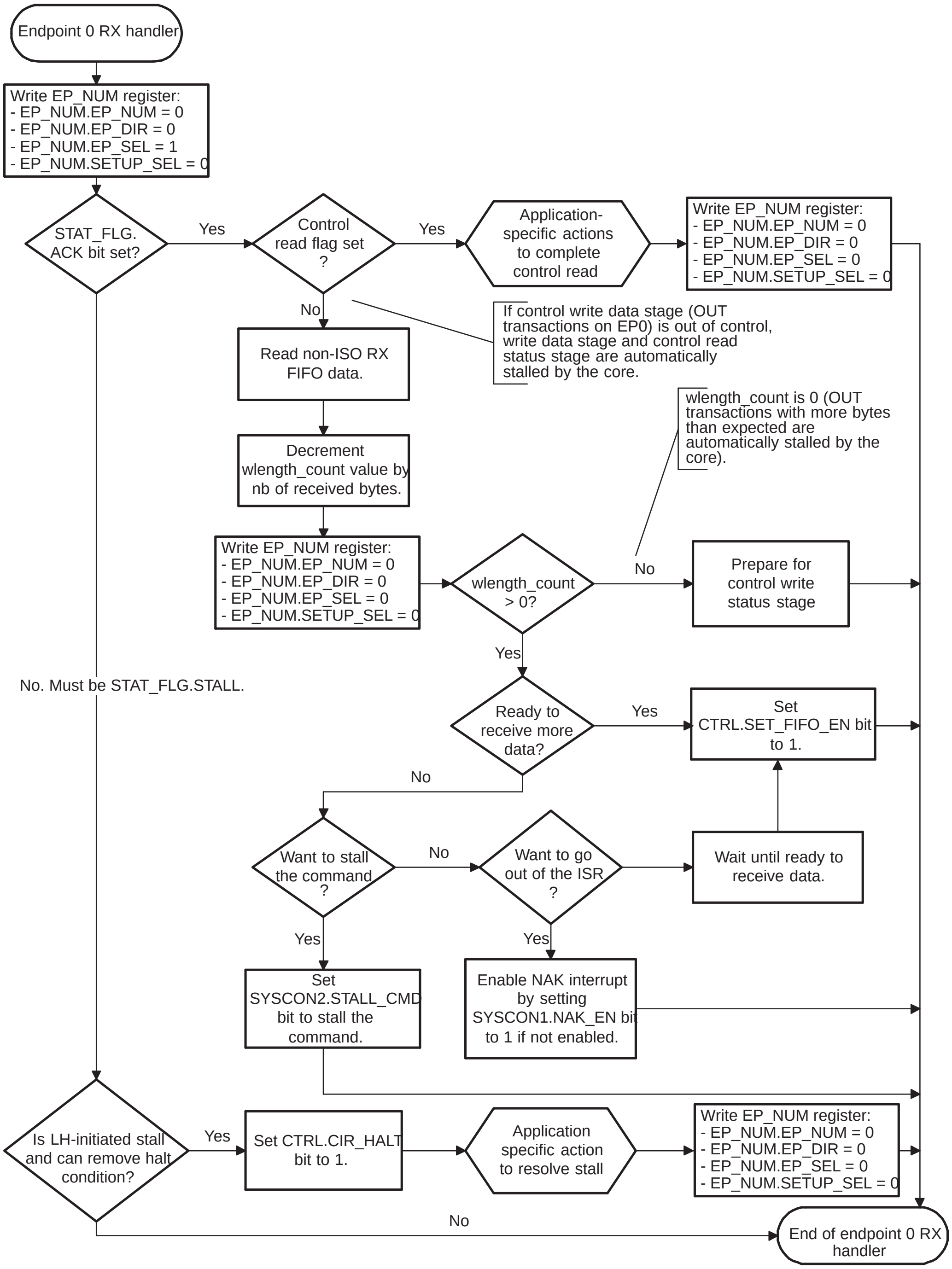
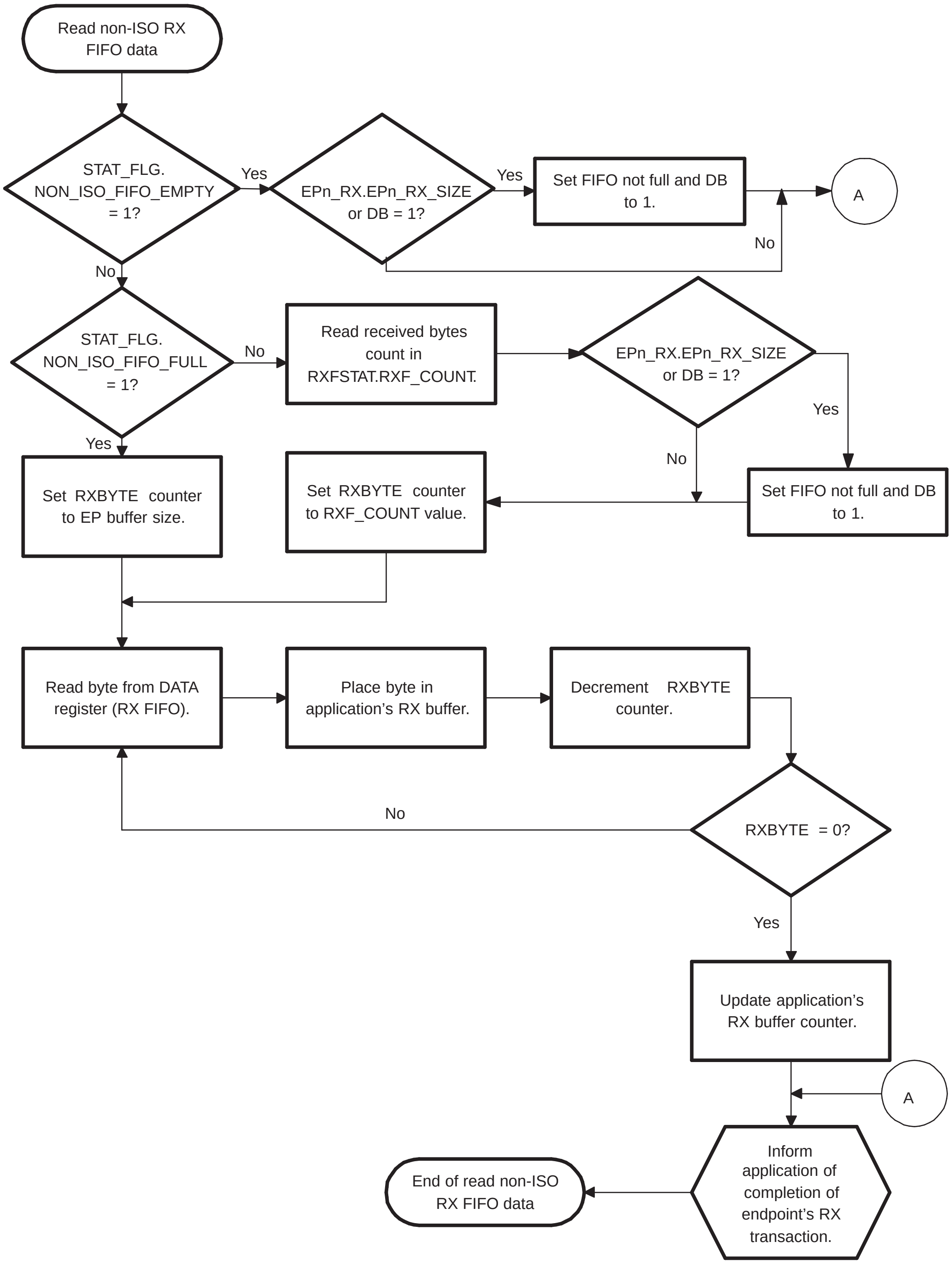

Endpoint 0 RX handler
Write EP_NUM register:
- EP_NUM.EP_NUM = 0
- EP_NUM.EP_DIR = 0
- EP_NUM.EP_SEL = 1
- EP_NUM.SETUP_SEL = 0
STAT_FLG. ACK bit set?
Yes
Control read flag set ?
Yes
Application- specific actions to complete control read
Write EP_NUM register:
- EP_NUM.EP_NUM = 0
- EP_NUM.EP_DIR = 0
- EP_NUM.EP_SEL = 0
- EP_NUM.SETUP_SEL = 0
No
If control write data stage (OUT transactions on EP0) is out of control, write data stage and control read status stage are automatically stalled by the core.
Read non-ISO RX FIFO data.
wlength_count is 0 (OUT transactions with more bytes than expected are automatically stalled by the core).
Decrement wlength_count value by nb of received bytes.
Write EP_NUM register:
- EP_NUM.EP_NUM = 0
- EP_NUM.EP_DIR = 0
- EP_NUM.EP_SEL = 0
- EP_NUM.SETUP_SEL = 0
wlength_count > 0?
No
Prepare for control write status stage
Yes
No. Must be STAT_FLG.STALL.
Ready to receive more data?
Yes
Set CTRL.SET_FIFO_EN bit to 1.
No
Want to stall the command ?
No
Want to go out of the ISR ?
Wait until ready to receive data.
Yes
Yes
Set SYSCON2.STALL_CMD bit to stall the command.
Enable NAK interrupt by setting SYSCON1.NAK_EN bit to 1 if not enabled.
Is LH-initiated stall and can remove halt condition?
Yes
Set CTRL.CIR_HALT bit to 1.
Application specific action to resolve stall
Write EP_NUM register:
- EP_NUM.EP_NUM = 0
- EP_NUM.EP_DIR = 0
- EP_NUM.EP_SEL = 0
- EP_NUM.SETUP_SEL = 0
No
End of endpoint 0 RX handler

Figure 32-78. Read Non-Isochronous RX FIFO Data Flowchart


Read non-ISO RX FIFO data
STAT_FLG. NON_ISO_FIFO_EMPTY = 1?
Yes
EPn_RX.EPn_RX_SIZE or DB = 1?
Yes
Set FIFO not full and DB to 1.
A
No
No
STAT_FLG. NON_ISO_FIFO_FULL = 1?
No
Read received bytes count in RXFSTAT.RXF_COUNT.
EPn_RX.EPn_RX_SIZE or DB = 1?
Yes
Yes
No
Set RXBYTE counter to EP buffer size.
Set RXBYTE counter to RXF_COUNT value.
Set FIFO not full and DB to 1.
Read byte from DATA register (RX FIFO).
Place byte in application's RX buffer.
Decrement RXBYTE counter.
No
RXBYTE = 0?
Yes
Update application's RX buffer counter.
A
Inform application of completion of endpoint's RX transaction.
End of read non-ISO RX FIFO data

32.3.24 Non-ISO, Non-Control IN Endpoint Transmit Interrupt Handler

Figure 32-79 shows the operations necessary to handle non-ISO, non-control IN endpoint-specific transmit interrupts. This flowchart shows two TX transaction handshaking interrupts. There is a third interrupt handshaking possibility when NAK interrupts are enabled, which is not depicted here. Depending on the application-specific actions needed for various endpoints in the real system, it is possible to use one routine that is common to all of the non-ISO, non-control transmit endpoints, where the only differences are in the EP_NUM register value set and in the routine selection of the application TX buffer (see Figure 32-80).

This flowchart does not attempt to document control endpoint 0 transmit interrupts, which are discussed separately because of the more complex three-stage transfer mechanism used for control reads.

Figure 32-79. Non-Isochronous Non-Control Endpoint Transmit Interrupt Handler

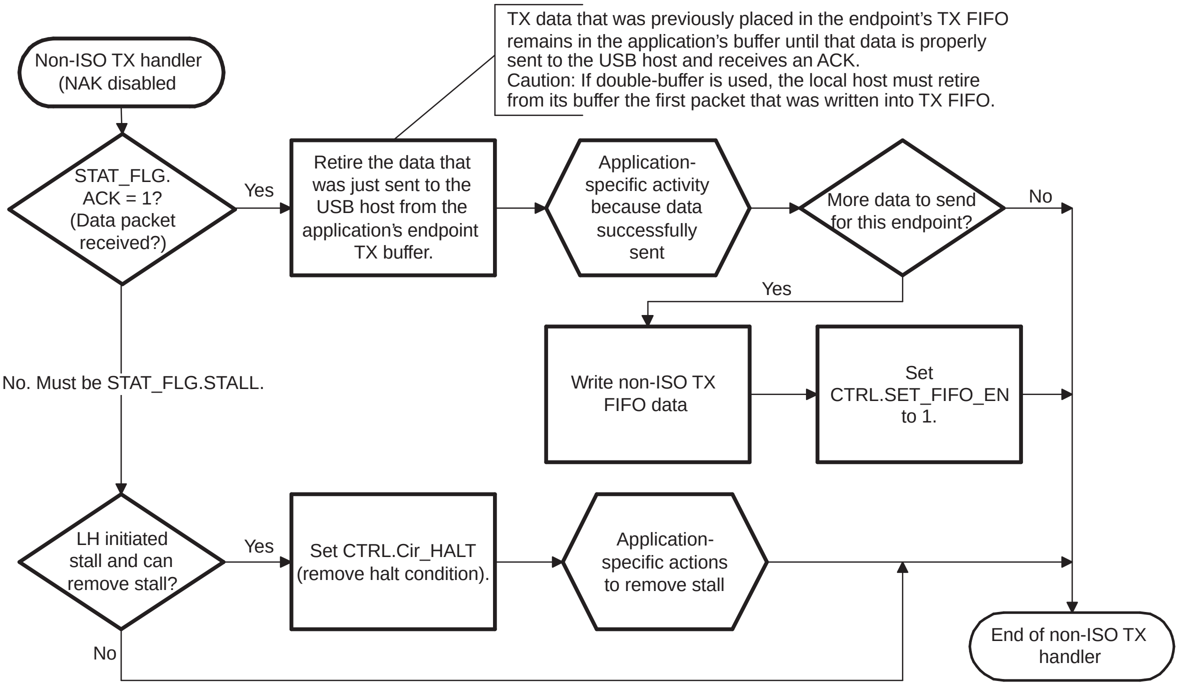
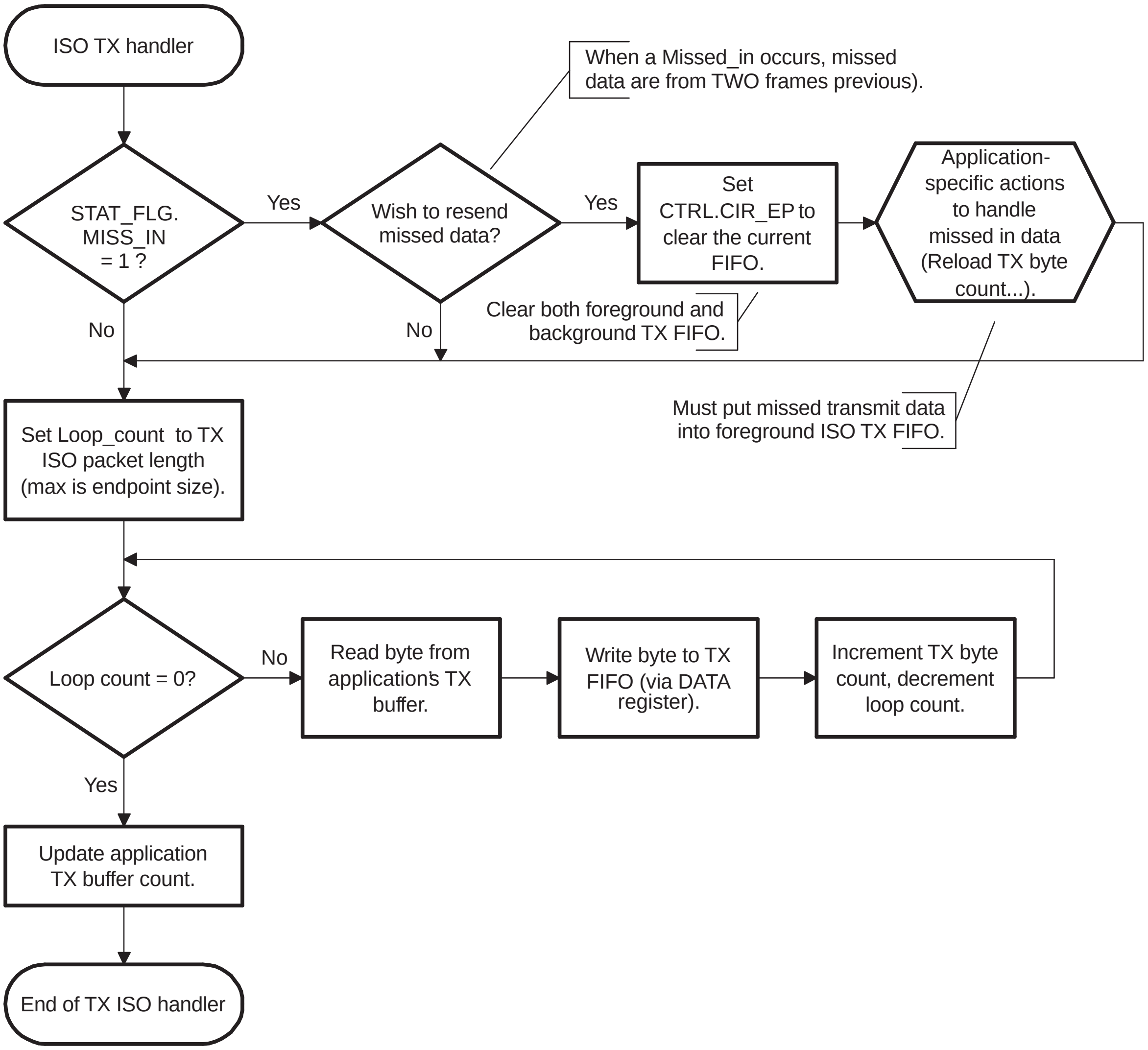

TX data that was previously placed in the endpoint's TX FIFO remains in the application's buffer until that data is properly sent to the USB host and receives an ACK.
Caution: If double-buffer is used, the local host must retire from its buffer the first packet that was written into TX FIFO.
Non-ISO TX handler (NAK disabled
STAT_FLG. ACK = 1? (Data packet received?)
Yes
Retire the data that was just sent to the USB host from the application's endpoint TX buffer.
Application-specific activity because data successfully sent
More data to send for this endpoint?
No
Yes
No. Must be STAT_FLG.STALL.
Write non-ISO TX FIFO data
Set CTRL.SET_FIFO_EN to 1.
LH initiated stall and can remove stall?
Yes
Set CTRL.Cir_HALT (remove halt condition).
Application-specific actions to remove stall
No
End of non-ISO TX handler

Figure 32-80. Write Non-Isochronous TX FIFO Data Flowchart


ISO TX handler
When a Missed_in occurs, missed data are from TWO frames previous).
STAT_FLG. MISS_IN = 1 ?
Yes
Wish to resend missed data?
Yes
Set CTRL.CIR_EP to clear the current FIFO.
Application-specific actions to handle missed in data (Reload TX byte count...).
No
No
Clear both foreground and background TX FIFO.
Must put missed transmit data into foreground ISO TX FIFO.
Set Loop_count to TX ISO packet length (max is endpoint size).
Loop count = 0?
No
Read byte from application's TX buffer.
Write byte to TX FIFO (via DATA register).
Increment TX byte count, decrement loop count.
Yes
Update application TX buffer count.
End of TX ISO handler

32.3.25 SOF Interrupt Handler

SOF interrupts to the CPU (also known as isochronous interrupts on VIM channel # 68 by default) occur once per USB frame. The CPU must handle data traffic for the isochronous endpoints at each SOF interrupt. In addition, the SOF ISR can handle any application-specific activity related to the implicit timing of the SOF interrupt. Figure 32-81 shows the SOF ISR flowchart. The read ISO RX FIFO data and write ISO TX FIFO data procedures are shown in Figure 32-82 and Figure 32-83, respectively.

Figure 32-81. SOF Interrupt Handler Flowchart

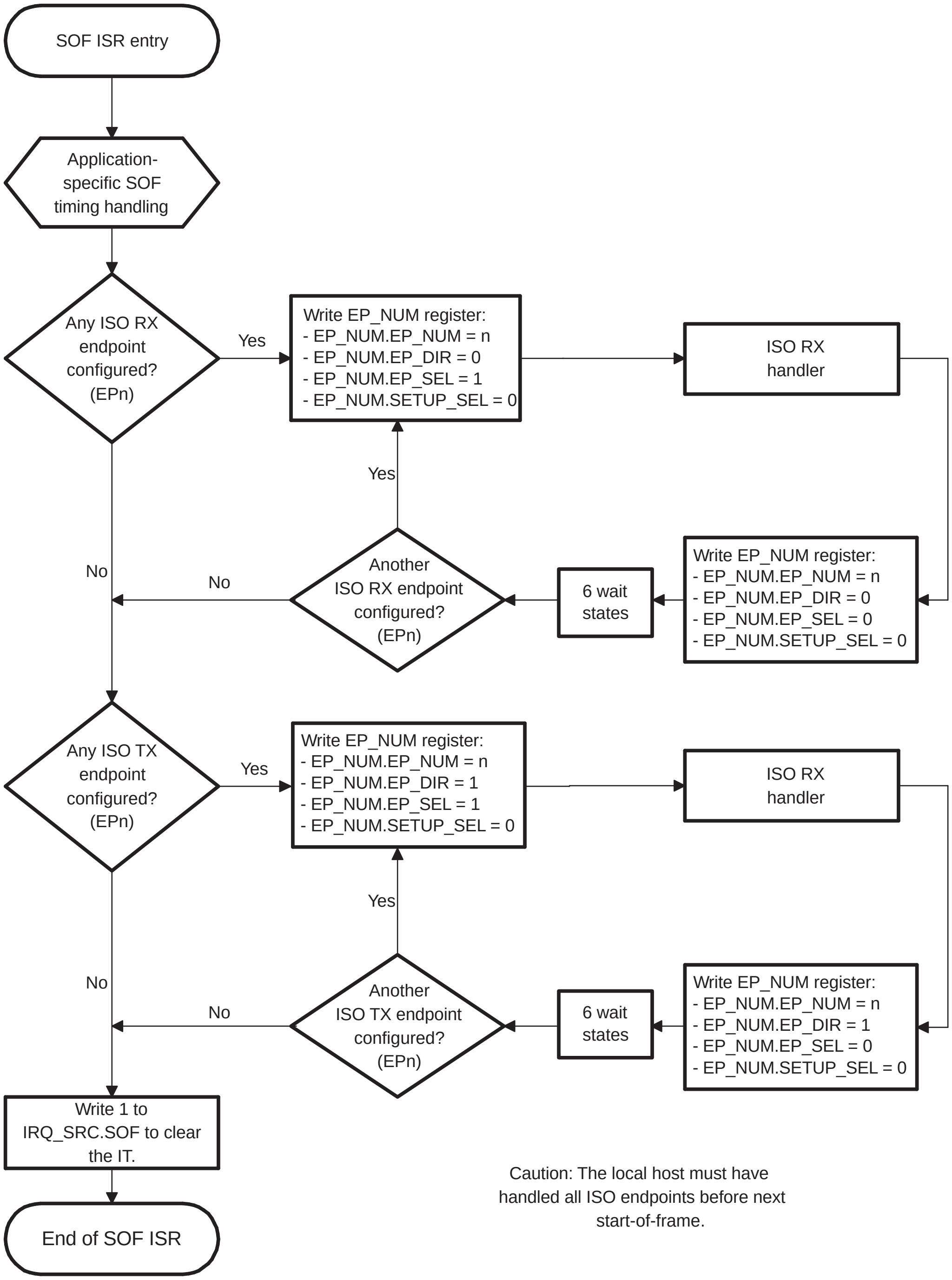


Figure 32-82. Read Isochronous RX FIFO Data Flowchart

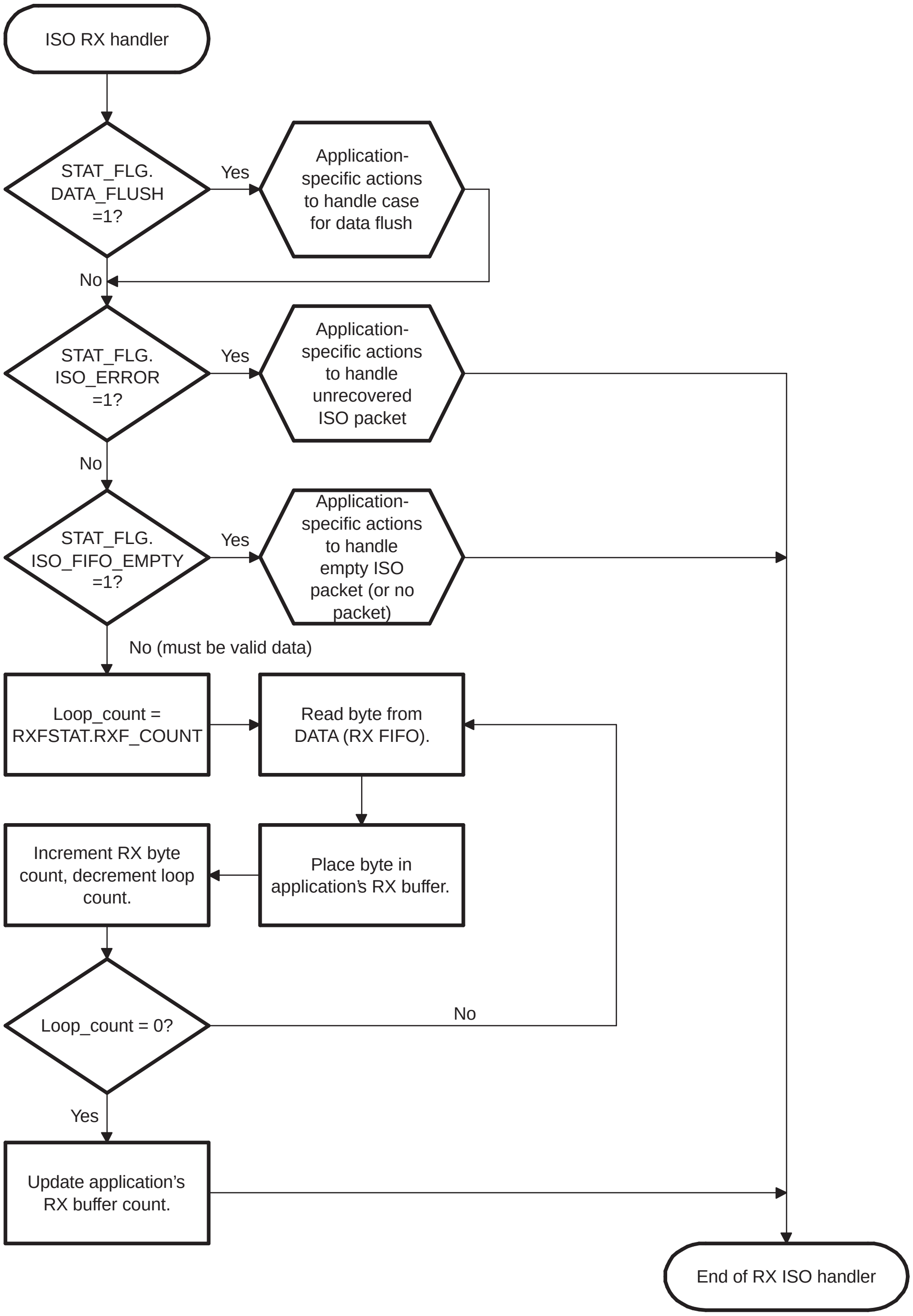
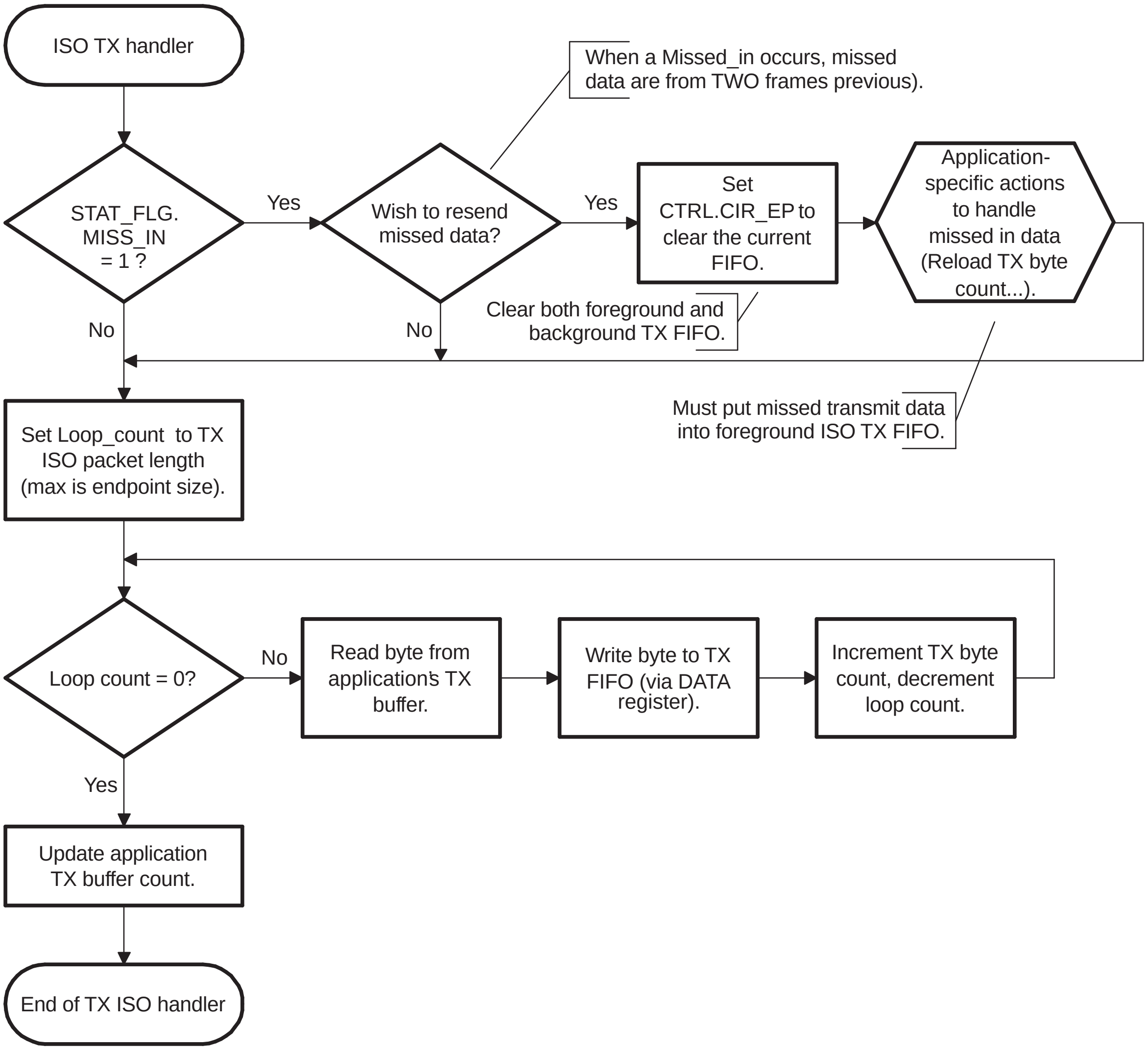

ISO RX handler
STAT_FLG. DATA_FLUSH =1?
Yes
Application-specific actions to handle case for data flush
No
STAT_FLG. ISO_ERROR =1?
Yes
Application-specific actions to handle unrecovered ISO packet
No
STAT_FLG. ISO_FIFO_EMPTY =1?
Yes
Application-specific actions to handle empty ISO packet (or no packet)
No (must be valid data)
Loop_count = RXFSTAT.RXF_COUNT
Read byte from DATA (RX FIFO).
Place byte in application's RX buffer.
Increment RX byte count, decrement loop count.
Loop_count = 0?
No
Yes
Update application's RX buffer count.
End of RX ISO handler

Figure 32-83. Write Isochronous TX FIFO Data Flowchart


ISO TX handler
When a Missed_in occurs, missed data are from TWO frames previous).
STAT_FLG. MISS_IN = 1 ?
Yes
Wish to resend missed data?
Yes
Set CTRL.CIR_EP to clear the current FIFO.
Application-specific actions to handle missed in data (Reload TX byte count...).
No
No
Clear both foreground and background TX FIFO.
Must put missed transmit data into foreground ISO TX FIFO.
Set Loop_count to TX ISO packet length (max is endpoint size).
Loop count = 0?
No
Read byte from application's TX buffer.
Write byte to TX FIFO (via DATA register).
Increment TX byte count, decrement loop count.
Yes
Update application TX buffer count.
End of TX ISO handler

32.3.26 *Summary of USB Device Controller Interrupts*

Table 32-56 lists the interrupt types by endpoint types.

Table 32-56. USB Device Controller Interrupt Type by Endpoint Type

	General USB IRQs (VIM channel # 69 by default)				EP-Specific IRQs (Non-ISO Interrupt on VIM channel # 70 by default)		SOF ISO Interrupt on VIM channel # 68 by default)
Interrupt Type	**Setup (EP0)**	**Control (EP0) Out**	**Control (EP0) In**	**Other**	**Bulk or InterruptOut**	**Bulk or InterruptIn**	**(Isochronous) SOF**
Transaction ACKed		X	X		X	X	
Transaction NAKed (if enabled)		X	X		X	X	
Transaction Stalled		X	X		X	X	
Setup	X						
SOF							X
Device state changed				X			
RX DMA EOT (non_ISO)				X			
RX DMA trans count (non_ISO)				X			
TX DMA done (non_ISO)				X			

32.3.26.1 USB Device Controller Clock Control

The device global clock module (GCM) provides a single 48-MHz clock to the USB device controller and USB host controller. This is the VCLKA3 clock domain. This clock domain can be stopped by software to reduce USB host and USB device controller power consumption when USB functionality is not needed. The GCM controls the 48-MHz clock to the USB device controller via the following memory-mapped registers:

- SYS.CDDIS
- SYS.CDDISSET
- SYS.CDDISCLR

32.3.26.2 USB Device Controller Hardware Reset

Reset of the USB device controller is provided by the SYS module. The system reset signal is used to reset the USB host controller and the USB device controller. When held in reset, the USB device controller does not recognize any USB activity, and its registers have no effect on USB functionality.

32.3.27 DMA Operation

The USB device controller provides support for six DMA channels. Three receive DMA channels are reserved to OUT transfers (ISO or non-ISO) and three transmit DMA channels are reserved to IN transfers (ISO or non-ISO). It is not possible to operate DMA transactions on control EP0.

NOTE: The CPU must not access an endpoint used in a DMA transfer through the EP_NUM, CTRL, and STAT_FLG registers (in DMA, this remark applies after the CPU has set the CTRL.SET_FIFO_EN bit to enable the RX DMA transfer). In particular, the CPU must not set the halt feature while the endpoint is selected in the RXDMA_CFG register.

To use the DMA channels properly, you must set the DMA configuration during the address state interrupt (DS_CHANGE).

The parameters used for DMA transactions (FIFO size, ISO endpoint, double-buffering, and pointers) are those defined for the associated endpoint.

32.3.27.1 Receive DMA Channels Overview

Receive DMA channels are programmed via the three RXDMA control registers. Each channel is assigned to a given endpoint number by assigning a non-zero value in RXDMA_CFG.RXDMAn_EP fields (a 0 value means the DMA channel is deselected). Received OUT data must be read when an RX DMA request is active, through the register DATA_DMA. The RX FIFO accessed is that of the endpoint for which the DMA request is active (only one RX DMA request is active at a time).

The USB device controller receive DMA channels 0, 1, and 2 are connected to the device System DMA controller requests # 15, 19, and 29, respectively.

32.3.27.2 Non-Isochronous OUT (USB HOST -> CPU) DMA Transactions

During non-ISO transfers to a DMA operated OUT endpoint, a request to the CPU DMA controller is generated when data have been placed into endpoint FIFO and must be read. ACK and NAK interrupts are always disabled automatically by the core for DMA operated endpoints.

There are two dedicated maskable interrupts per DMA channel to control non-ISO OUT transfers.

32.3.27.3 End of Transfer Interrupt (IRQ_SRC.RXn_EOT)

This interrupt signals that the core has detected an end-of-transfer (EOT). EOT occurs in the two following cases:

- When the last valid transaction to the endpoint is either an empty packet (ACK and buffer empty) or a packet whose size is less than the physical endpoint buffer size (ACK and buffer not full)
- When the number of received transactions has reached the programmed value in the RXDMAn.RXn_TC field, if the RXDMAn.RXn_STOP bit has been set by the CPU

After an end of transfer interrupt, the CPU must set CTRL.SET_FIFO_EN for the endpoint to reenable the channel.

The CPU must not initiate a new RX DMA transfer until it receives an end-of-transfer interrupt.

32.3.27.4 Transaction Count Interrupt (IRQ_SRC.RXn_CNT)

The intent of this interrupt is watermark control. It can be used by the CPU to monitor the file size of incoming transfers and take appropriate actions if, for instance, the file being received exceeds an expected size.

A transaction count interrupt does not disable the ongoing DMA transfer.

A transaction count interrupt occurs each time the number of received transactions (and not bytes) has reached the programmed value in the receive transaction counter for the DMA channel. One transaction has a size equal to the buffer size of the selected non-ISO endpoint. RXn_COUNT interrupt is asserted even if RXDMAn.RXn_STOP has been set; in that case, both RXn_COUNT and RXn_EOT interrupts are asserted (see Figure 32-84 through Figure 32-87).

The transaction count watermark is programmed in the RXDMAn register.

Figure 32-84. Non-ISO RX DMA Transaction Example (RX_TC=2)

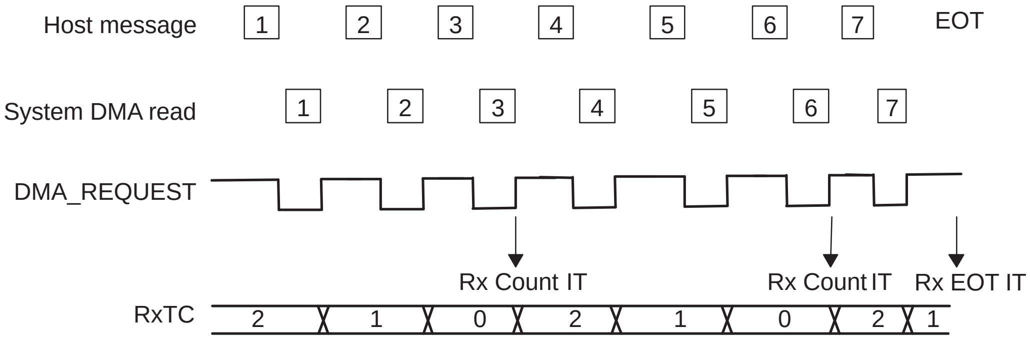


Figure 32-85. Non-ISO RX DMA Start Routine

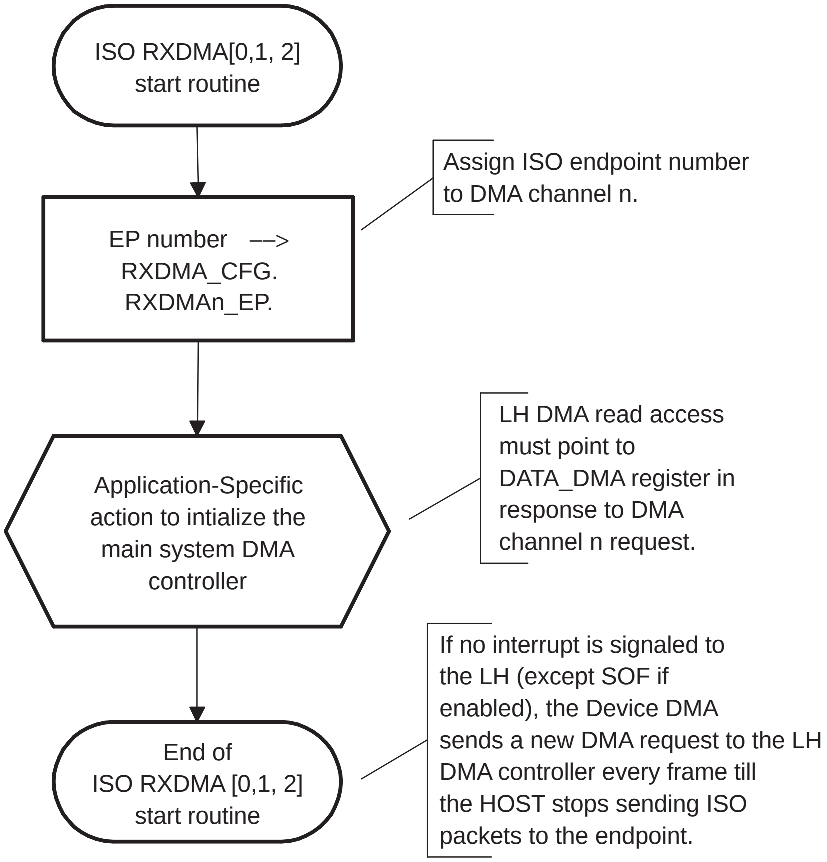


Figure 32-86. Non-ISO RX DMA EOT Interrupt Handler

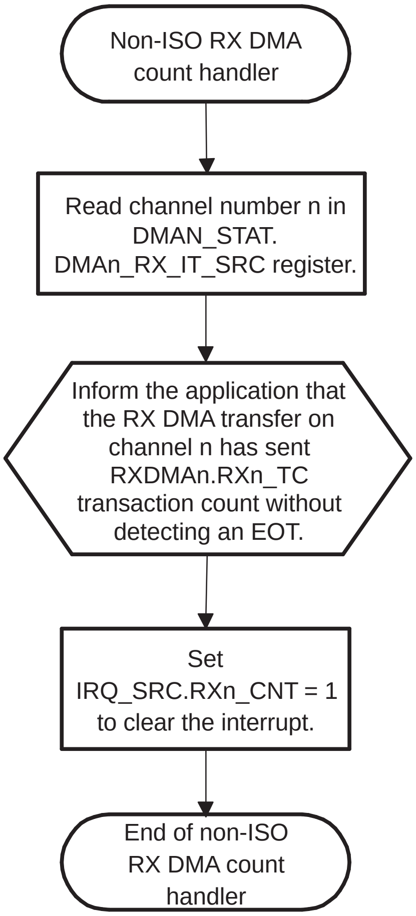
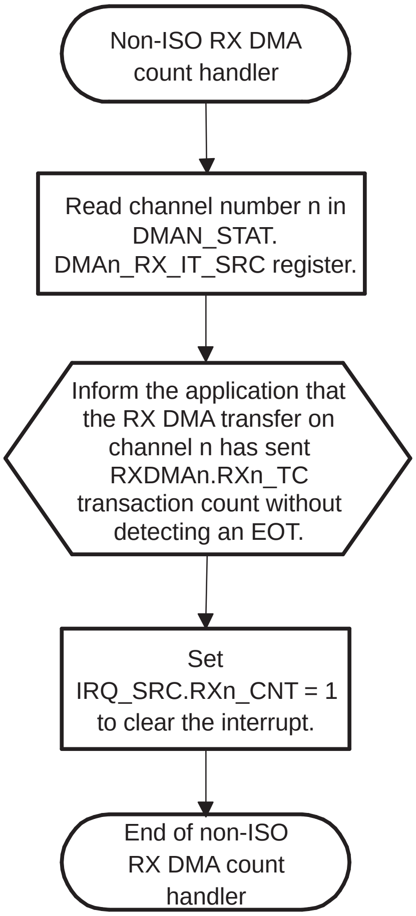

Non-ISO RX DMA count handler
Read channel number n in DMAN_STAT. DMAn_RX_IT_SRC register.
Inform the application that the RX DMA transfer on channel n has sent RXDMAn.RXn_TC transaction count without detecting an EOT.
Set IRQ_SRC.RXn_CNT = 1 to clear the interrupt.
End of non-ISO RX DMA count handler

Figure 32-87. Non-ISO RX DMA Transactions Count Interrupt Handler


Non-ISO RX DMA count handler
Read channel number n in DMAN_STAT. DMAn_RX_IT_SRC register.
Inform the application that the RX DMA transfer on channel n has sent RXDMAn.RXn_TC transaction count without detecting an EOT.
Set IRQ_SRC.RXn_CNT = 1 to clear the interrupt.
End of non-ISO RX DMA count handler

32.3.27.5 Isochronous OUT (USB HOST -> CPU) DMA Transactions

During ISO transfers to a DMA operated OUT endpoint, a request to the CPU DMA controller is generated every 1-ms frame when an isochronous data packet is received with no error. There is no interrupt associated with DMA transfer to ISO OUT endpoints (see Figure 32-88 and Figure 32-89).

Figure 32-88. ISO RX DMA Transaction

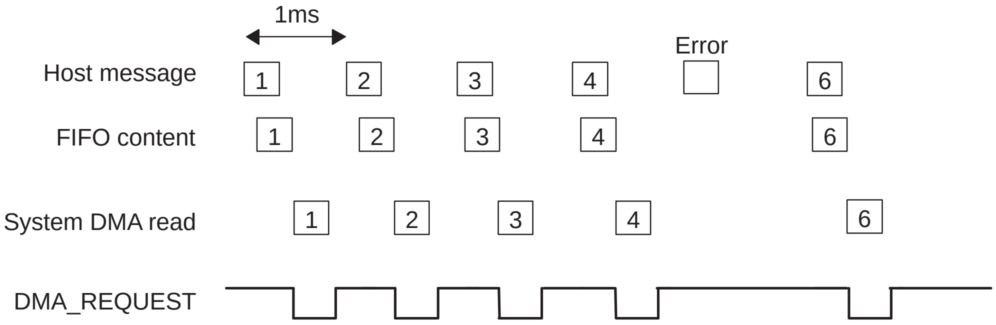
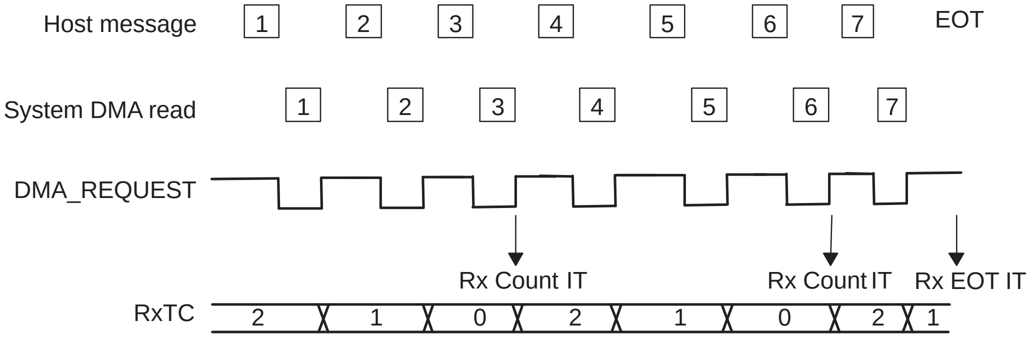


Figure 32-89. ISO RX DMA Start Routine



32.3.27.6 Transmit DMA Channels Overview

Transmit DMA channels are programmed via the three TXDMA control registers. Each channel can be assigned to a given endpoint number by assigning a non-zero value in TXDMA_CFG.TXDMAn_EP (a 0 value means the DMA channel is deselected). The other three control registers (TXDMA0, TXDMA1, and TXDMA2) operate in a different manner for ISO and non-ISO endpoints. Transmitted data must be written into the DATA_DMA when a TX DMA request is active. They are written into the TX FIFO of the endpoint associated with active request (only one TX DMA request active at a given time).

The USB device controller transmit DMA channels 0, 1, and 2 are connected to the device system DMA controller requests # 14, 20, and 28, respectively.

32.3.27.7 Non-Isochronous IN (CPU -> USB HOST) DMA Transactions

Non-ISO (bulk) TX DMA file transfers are virtually unlimited in size. The flowcharts depicted in Figure 32-90 and Figure 32-91 show how to handle small, medium, or large file transfers.

TXDMA0, TXDMA1, and TXDMA2 registers operate for non-ISO endpoints in the following manner. The transfer size counter (TXDMAn.TXN_TSC) corresponds to either the number of bytes to transmit (EOT bit set) or the number of buffers to transmit (EOT bit cleared). The buffer size corresponds to the programmed size of the TX endpoint.

A request to the CPU main DMA controller is generated when the endpoint buffer is empty initially after that the START bit is set and then each time there is space free in TX FIFO for other TX packet to be written, until TXDMAn.TXn_TSC counts down to zero. The request is removed when the buffer is full or when there are no more bytes of data to be sent.

A DMA transmit transfer done interrupt is signaled to the CPU after the last IN transaction completes successfully. This is after START bit was set and after TXDMAn.TXn_TSC equals 0 for the selected DMA channel.

The CPU must not initiate a new TX DMA transfer until it receives a TX_DONE interrupt.

A small file transfer less than 1024 bytes can be achieved in a single pass DMA signaled by a single interrupt completion. A file size equal to or greater than 1024 bytes needs two or more DMA passes, signaled by an interrupt completion after each pass.

Figure 32-90 shows the necessary steps to prepare and permit a TX DMA transfer of any size. It also effectively starts the initial DMA transfer. The completion of this DMA task is signaled to the CPU via a DONE interrupt, whose handler is shown in Figure 32-91. The start routine and the associated interrupt handler are tightly coupled.

Figure 32-90. Non-ISO TX DMA Start Routine

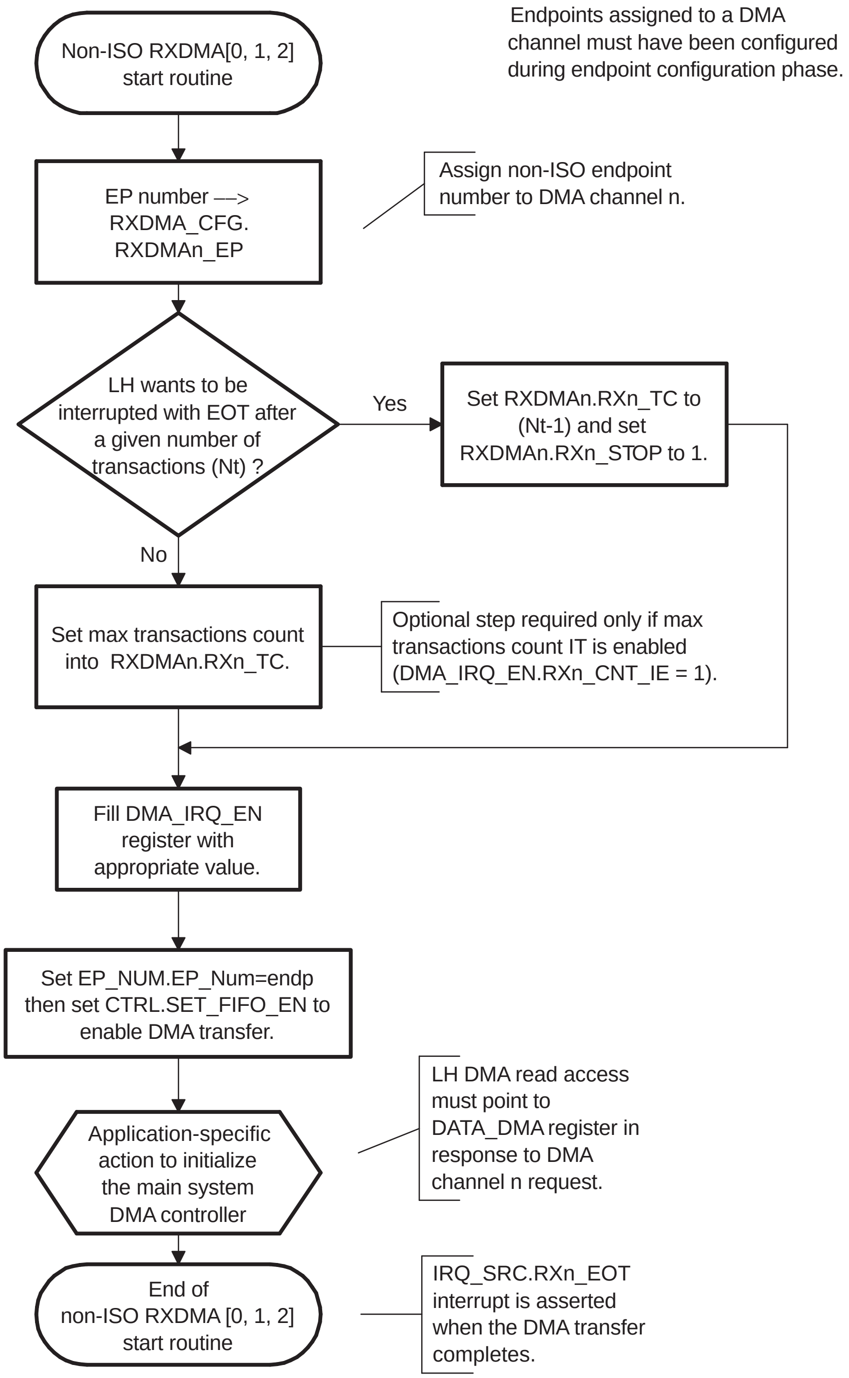
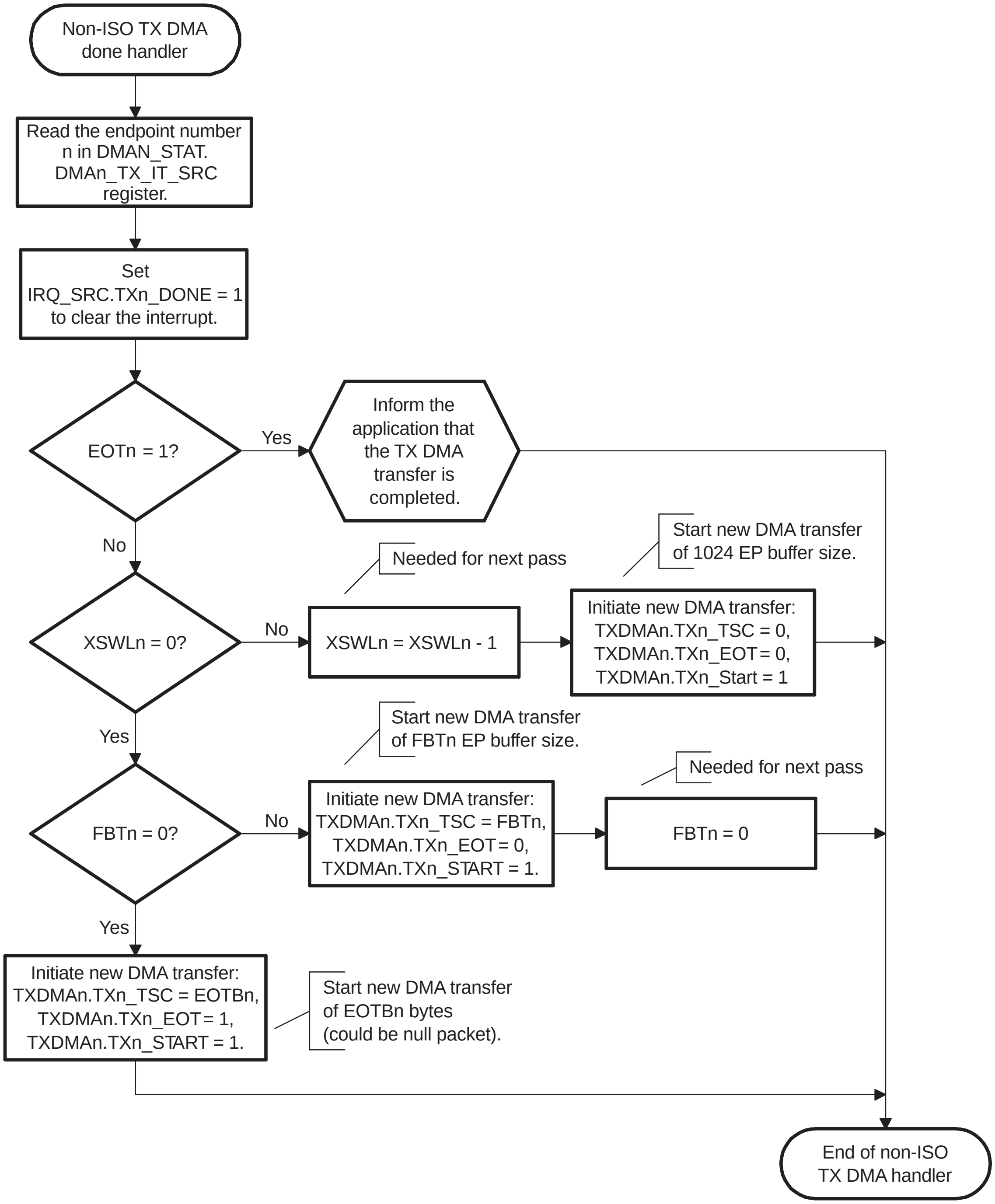

Endpoints assigned to a DMA channel must have been configured during endpoint configuration phase.
Non-ISO RXDMA[0, 1, 2] start routine
EP number —> RXDMA_CFG. RXDMAn_EP
Assign non-ISO endpoint number to DMA channel n.
LH wants to be interrupted with EOT after a given number of transactions (Nt) ?
Yes
Set RXDMAn.RXn_TC to (Nt-1) and set RXDMAn.RXn_STOP to 1.
No
Set max transactions count into RXDMAn.RXn_TC.
Optional step required only if max transactions count IT is enabled (DMA_IRQ_EN.RXn_CNT_IE = 1).
Fill DMA_IRQ_EN register with appropriate value.
Set EP_NUM.EP_Num=endp then set CTRL.SET_FIFO_EN to enable DMA transfer.
Application-specific action to initialize the main system DMA controller
LH DMA read access must point to DATA_DMA register in response to DMA channel n request.
End of non-ISO RXDMA [0, 1, 2] start routine
IRQ_SRC.RXn_EOT interrupt is asserted when the DMA transfer completes.

Figure 32-91. Non-ISO TX DMA Done Interrupt Handler


Non-ISO TX DMA done handler
Read the endpoint number n in DMAN_STAT. DMAn_TX_IT_SRC register.
Set IRQ_SRC.TXn_DONE = 1 to clear the interrupt.
EOTn = 1?
Yes
Inform the application that the TX DMA transfer is completed.
No
Needed for next pass
Start new DMA transfer of 1024 EP buffer size.
XSWLn = 0?
No
XSWLn = XSWLn - 1
Initiate new DMA transfer: TXDMAn.TXn_TSC = 0, TXDMAn.TXn_EOT= 0, TXDMAn.TXn_Start = 1
Yes
Start new DMA transfer of FBTn EP buffer size.
Needed for next pass
FBTn = 0?
No
Initiate new DMA transfer: TXDMAn.TXn_TSC = FBTn, TXDMAn.TXn_EOT= 0, TXDMAn.TXn_START = 1.
FBTn = 0
Yes
Initiate new DMA transfer: TXDMAn.TXn_TSC = EOTBn, TXDMAn.TXn_EOT= 1, TXDMAn.TXn_START = 1.
Start new DMA transfer of EOTBn bytes (could be null packet).
End of non-ISO TX DMA handler

Example 32-1. 100603 Bytes to Transfer via 32 Bytes IN Bulk Endpoint

This gives XSWL=0x3, FBT=0x47, EOTB=0x1b,

which means five passes of DMA transfer, signaled by 5 TXn_DONE interrupts, are required:

1. EOT=0, FBT=0, loop=3 32768 bytes transferred (1024 x 32 bytes)
2. EOT=0, FBT=0, loop=2 32768 bytes
3. EOT=0, FBT=0, loop=1 32768 bytes
4. EOT=0, FBT=0x47, loop=0 2272 bytes (71 x 32 bytes)
5. EOT=1, FBT=0x1B, loop=0 27 bytes

32.3.27.8 Isochronous IN (USB HOST -> CPU) DMA Transactions

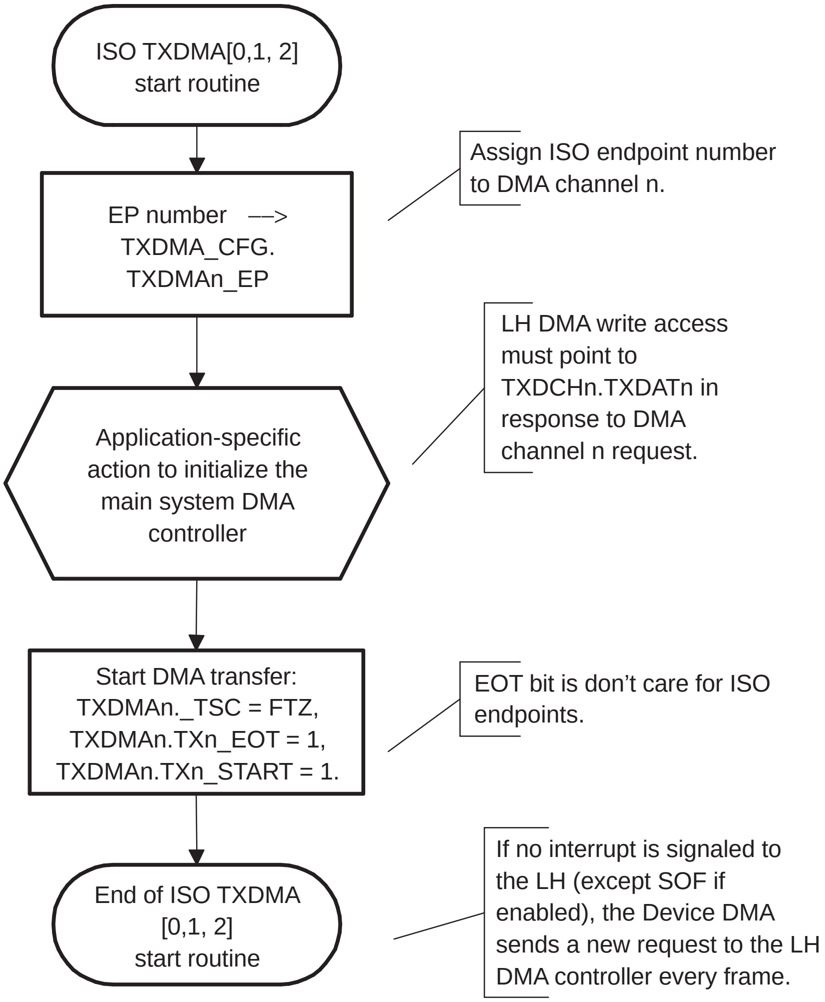
For ISO endpoints, the transfer size counter (TXDMAn.TXn_TSC) corresponds to the number of bytes to transmit. The programmed size must not exceed the programmed buffer size of the endpoint. Otherwise, the result is unpredictable (see Figure 32-92).

A request to the CPU main DMA controller is generated when the endpoint buffer is empty initially after the START bit is set, and then after each SOF (every 1 ms). The request is removed when the number of bytes written in the buffer matches the TXDMAn.TXn_TSC value.

During ISO transfers to a DMA operated IN endpoint, a request to the CPU system DMA controller is generated every 1-ms frame when an isochronous data packet is received with no error. There is no special interrupt associated with the DMA transfer.

No interrupt is signaled to the CPU during DMA operation to ISO IN endpoints.

Figure 32-92. ISO TX DMA Start Routine



32.3.27.9 Important Note on DMA Requests

For each direction, only one DMA request can be active at any time. A request must then be serviced to allow the next pending request on the same direction to be asserted. In particular, a TX DMA request is asserted at each start-of-frame if a TX DMA channel is configured for an isochronous endpoint; this request must be serviced imperatively.

32.3.27.10 Note on DMA Channels Deconfiguration

It is recommended that the CPU wait for an EOT (RX) or a DONE (TX) interrupt before disabling the channel by writing a value 0 in the TX/RXDMA_CFG register. However, if needed by the application, the CPU can unselect the endpoint number in the TX/RXDMA_CFG register during a DMA transfer. The resulting behavior is:

- For RX transfer:
  - If RX DMA request is active for the endpoint when the endpoint is unselected, deconfiguration is effective only at the end of the RX DMA request (that is, after all the DMA data have been read). When double-buffering is used, the deconfiguration is effective after both buffers have been read (if both buffers were not empty at unselection). An EOT interrupt is asserted if an end-of-transfer is detected.
  - If the RX DMA request is not active when unselection occurs, the effect is immediate.
- For TX transfer:
  - If the request is active when the endpoint is unselected, deconfiguration is effective after the TX DMA request has been handled and the TX data have been sent through an IN transaction (both buffers in case of double-buffering with both buffers full). No TX_DONE interrupt is asserted even if TXDMAn.TSC bit value is 0 after the transaction.
  - If the TX DMA request is inactive when the endpoint is unselected, deconfiguration is effective when all data available in TX buffer(s) have been sent through IN transaction(s). If the TXDMAn_TSC value is 0 at this point, no TX_DONE interrupt is asserted. If TX_DONE interrupt had already been asserted when the endpoint is deselected, configuration is effective only after the TX_DONE interrupt handling.

TX/RXDMA_CFG.TX/RXDMAn_EP reflects the endpoint value until deconfiguration is effective. The CPU must read this register to know if the channel has been disabled yet or not. It must wait until the read value is 0 before performing other actions to the endpoint. After effective deconfiguration, all transactions to the endpoint generate an endpoint-specific interrupt (if non-transparent).

If the selected endpoint is of isochronous type, deconfiguration is effective after the TX/RX request has been serviced, and the subsequent isochronous transactions are handled at SOF interrupt through the endpoint registers (EP_NUM and STAT_FLG).

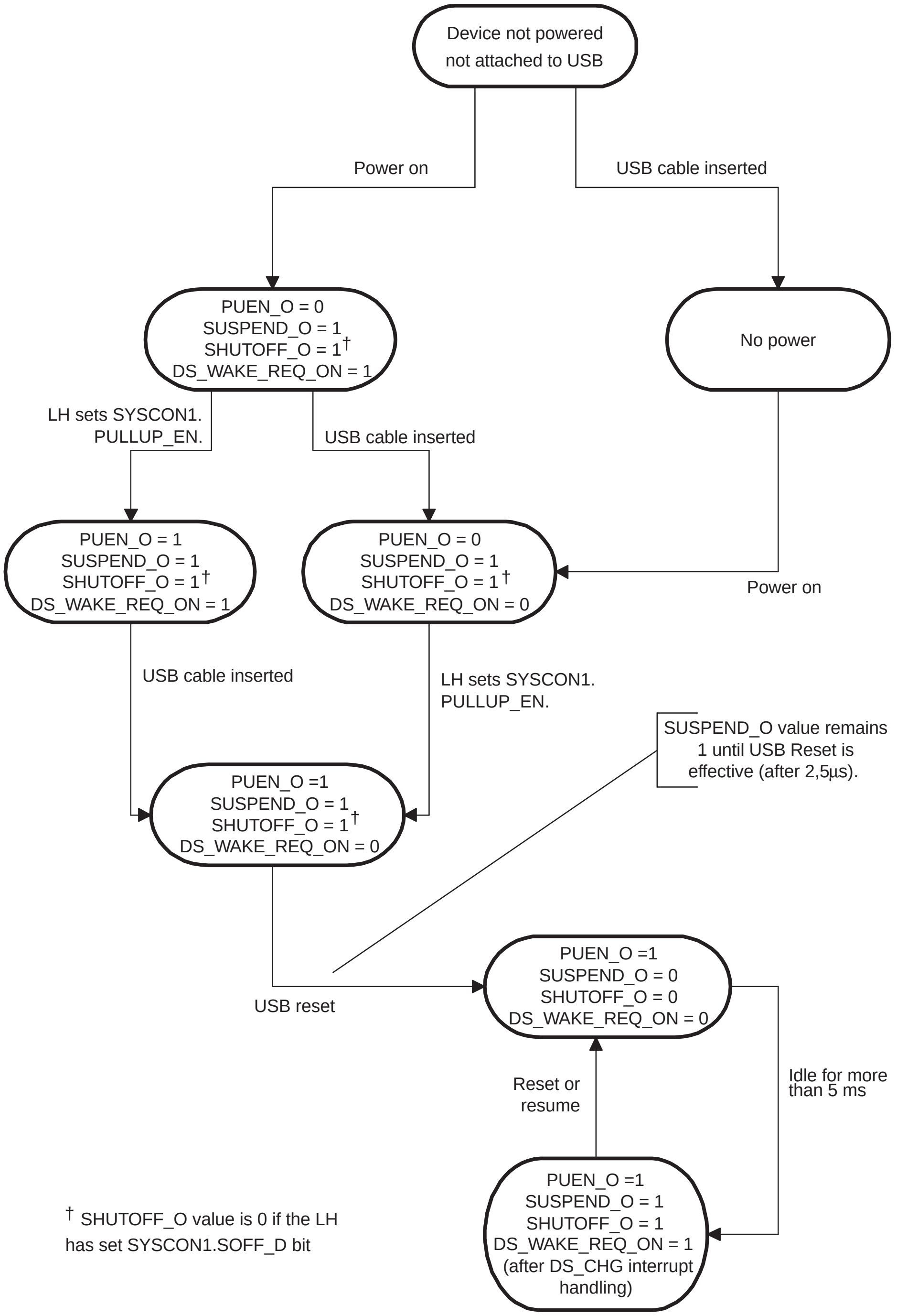
32.3.28 Power Management

Figure 32-93 shows the values assigned to the USB device controller signals concerned with power management, in the function of the device state. These signals are:

- PUEN_O: Pullup enable signal, always reflecting the SYSCON1.PULLUP_EN register bit
- SHUTOFF_O: Power circuitry shutoff signal, controlled by the core and the SYSCON1.SOFF_DIS bit
- DS_WAKE_REQ_ON: Deep-sleep wake request, asserted low when the interface clock is needed
- SUSPEND_O: Suspend signal, asserted high when the device is in suspend mode

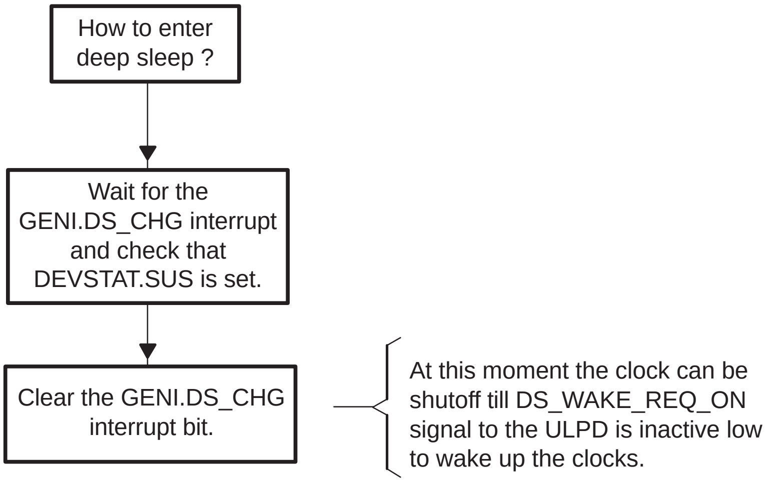
From a software point of view, Figure 32-94 shows the reaction. This flowchart does not need to be implemented; it only reflects the way the module can enter deep sleep.

Figure 32-93. Power Management Signal Values



A SHUTOFF_O value is 0 is the LH has set SYSCON1.SOFF_D bit.

Figure 32-94. Power Management Flowchart


How to enter deep sleep ?
Wait for the GENI.DS_CHG interrupt and check that DEVSTAT.SUS is set.
Clear the GENI.DS_CHG interrupt bit.
At this moment the clock can be shutoff till DS_WAKE_REQ_ON signal to the ULPD is inactive low to wake up the clocks.

32.4 USB Connectivity

This section includes:

- Transceiver interface
- Selecting and configuring USB connectivity
- Transceiver signaling
- Host controller connectivity with USB transceivers
- USB function controller connectivity with USB transceivers
- USB hardware considerations

32.4.1 Transceiver Interface

The device provides three USB ports, two of which may be used simultaneously. In a USB system, a USB transceiver is needed for each USB port. It converts between signaling appropriate for the USB controllers and signaling appropriate for the USB cable. This device does not provide an integrated USB transceiver.

Several different types of external transceiver signaling are supported. Signaling between the USB controller and the external USB transceiver for monitoring and controlling the differential USB signal is done via a 6-wire signaling interface, with two or more additional control signals provided either by additional signals or via an I2C link.

32.4.2 Selecting and Configuring USB Connectivity

The process of selecting desired USB connectivity and configuring the device for that connectivity can be done using the steps listed below.

32.4.2.1 Choose the Desired USB Functionality

Choose the desired USB functionality. The USB module on this microcontroller supports the following configurations:

- 1 USB Host Port + 1 USB Device Port, OR
- 2 USB Host Ports

Interface signals for the USB host port #2 are multiplexed with interface signals for the USB device port. If the application requires the use of the USB device port, then the USB host port #2 must first be powered down.

32.4.2.2 Select How USB Functionality Is Multiplexed to Pins

Refer to Table 32-1 for information on multiplexing and control of USB host ports and USB device port interface signals at the top level.

32.4.3 Transceiver Signaling

The USB module on this device uses a unidirectional (6-wire) signaling type for the transceiver. The signaling is shown in Table 32-57.

Table 32-57. Signaling Between USB Controller and Unidirectional USB Transceiver Using DAT/SE0 Signaling

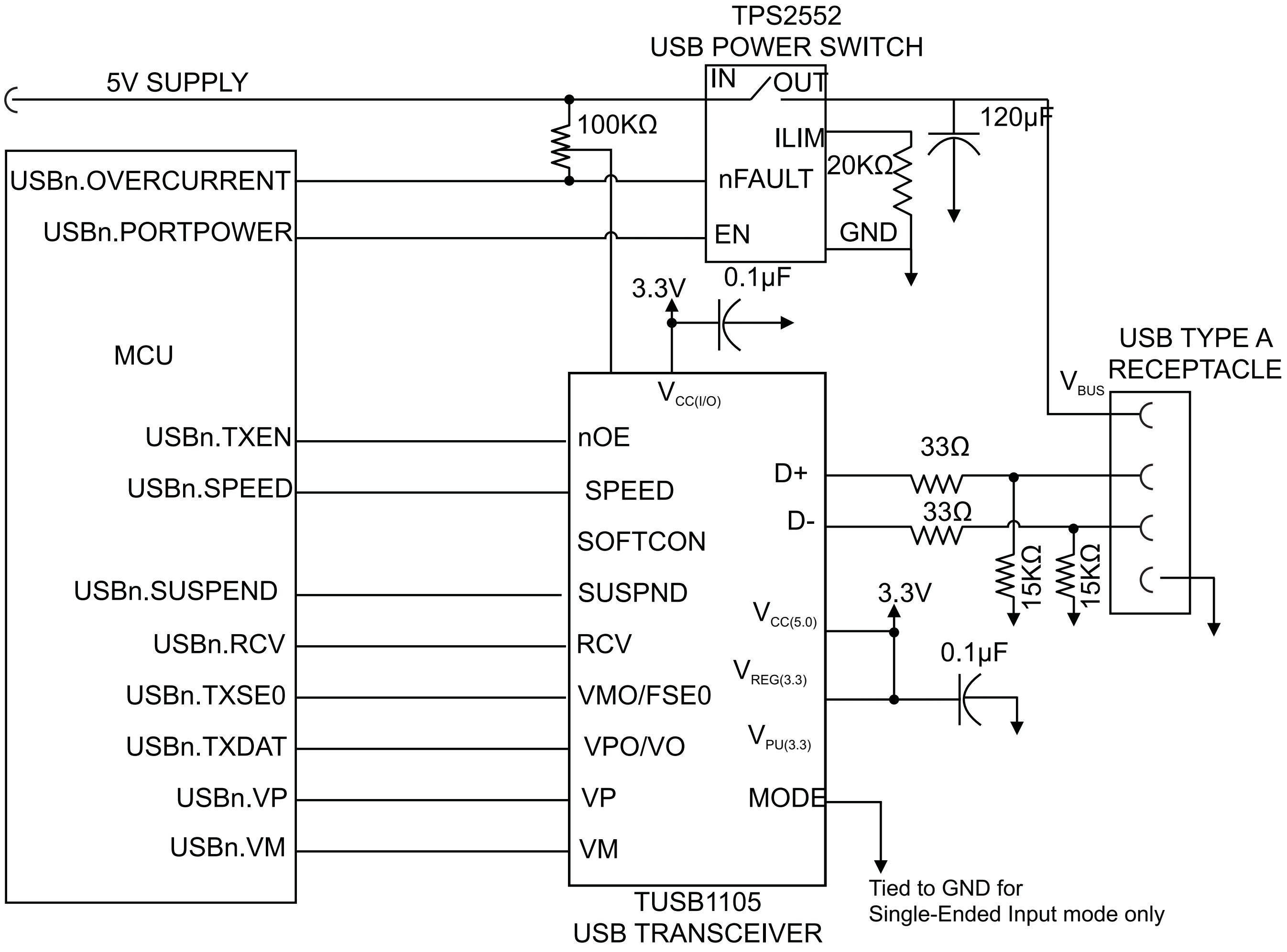
<table>
<tr><th>Logical Signal Name(s)</th><th>Pin Direction</th><th>Transceiver Pin Direction</th><th>Description</th></tr>
<tr><td>GZO</td><td>Output</td><td>Input</td><td>When low, USB transceiver drives D+ and D-.</td></tr>
<tr><td>DAT and SE0</td><td>Output</td><td>Input</td><td>Controls the values output by the USB transceiver on D+ and D- when GZO is low. Ignored when GZO is high.
<table>
<tr><th>GZO</th><th>DAT</th><th>SE0</th><th>D+</th><th>D-</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td></td><td>1</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td></td><td>X</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>X</td><td>X</td><td>Undriven</td><td>Undriven</td></tr>
</table></td></tr>
<tr><td>RCV</td><td>Input</td><td>Output</td><td>Output from transceiver differential receiver.
<table>
<tr><th>D+</th><th>D-</th><th>RCV</th></tr>
<tr><td>0</td><td>0</td><td>X</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>X</td></tr>
</table></td></tr>
<tr><td>VP</td><td>Input</td><td>Output</td><td>Output from transceiver single-ended D+ signal receiver .
<table>
<tr><th>D+</th><th>VP</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
</table></td></tr>
<tr><td>VM</td><td>Input</td><td>Output</td><td>Output from transceiver single-ended D- signal receiver.
<table>
<tr><th>D-</th><th>VM</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
</table></td></tr>
<tr><td>SPEED</td><td>Output</td><td>Input</td><td>Determines transceiver D+, D- output characteristics.
0 = Low speed.
1 = Full speed.
Transceivers with I^{2}C interfaces can implement this functionality as a control bit rather than a pin.</td></tr>
<tr><td>SUSPEND</td><td>Output</td><td>Input</td><td>Controls transceiver power down modes.
0 = Active mode.
1 = Low-power mode.
Transceivers with I^{2}C interfaces can implement this functionality as a control bit rather than a pin.</td></tr>
</table>

This device does not support unidirectional transceivers that implement VPO/VMO signaling.

32.4.4 Host Controller Connectivity With USB Transceivers

To provide a robust USB solution, a system that provides a USB host controller must implement certain features. These features include a type-A receptacle, power on the VBUS signal (can be switched or unswitched power), transient suppression, pulldown resistors, and USB-compatible downstream port transceiver.

Figure 32-95. Pin Group 0 USB Host Using 6-Wire USB Transceiver



Proper initialization of the device to support this mode of operation requires proper setting of the top-level pin multiplexing (see Section 32.4.2.2).

32.4.5 USB Function Controller Connectivity With USB Transceivers

To provide a robust USB solution, a system that provides a USB device controller port must implement certain features. These features include:

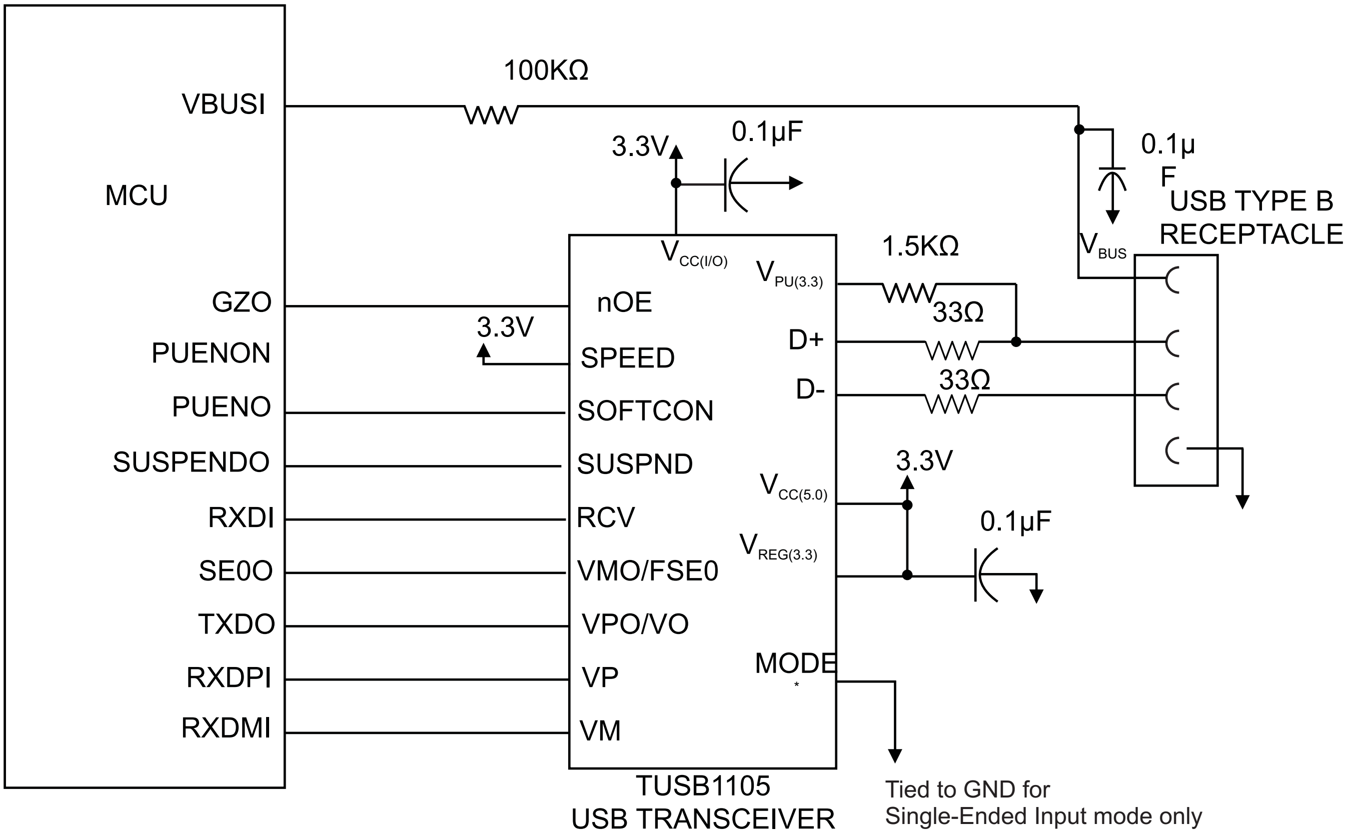
- A USB type-B or mini-B receptacle
- VBUS power detection
- Transient suppression
- A controllable pullup resistor to the D+ or D- line
- USB-compatible upstream port transceiver

Optional weak pulldown resistors can be used to hold the D+ and D- signals at voltages below the USB transceiver VIL level when there is no USB host connected to the USB Type B connector. By keeping D+ and D- voltages below VIL, the USB transceiver IDDQ can be reduced. Choice of value for the pulldown resistors must be made carefully so that the circuit meets the requirements of the *USB Specification*.

The USB function controller only supports implementation as a full-speed USB device. As such, the pullup resistor must be connected to the D+ signal to indicate implementation of a full-speed USB device.

This device supports a 6-wire connection to the USB transceiver.

Figure 32-96. USB Device Connections on USB Pin Group 0 Using 6-Wire Transceiver



Proper initialization of the device to support this mode of operation requires proper setting of the top-level pin multiplexing (see Section 32.4.2.2).

32.4.6 USB Hardware Considerations

32.4.6.1 VBUS Power Switching for USB Type A Host Receptacles

The USB specification places several VBUS requirements on USB hosts, including current capability, droop, and other important characteristics. Circuits that meet the USB specification requirements can be implemented using Texas Instruments devices such as the TPS2552 and TPS2553 power distribution switch devices. Further information, including data sheets and application notes, can be found at the Texas Instruments web site.

32.4.6.2 Transient Suppression for USB Connectors

It is important to provide transient suppression for USB connectors. Electrostatic discharge that occurs when a user connects or disconnects a USB cable can have a dramatic effect on a system if not suppressed. Texas Instruments offers several devices for transient suppression on USB connections, such as the SN65220, SN65240, and SN75240 universal serial bus port transient suppressor devices. Further information, including data sheets and application notes, can be found at the Texas Instruments web site.

32.4.6.3 VBUS Monitoring for USB Function Controller

A USB function controller must be capable of monitoring the VBUS voltage provided by the upstream USB host controller. The device provides the input pin USB_VBUS, which is provided to the USB function controller. This input is a CMOS input that is not rated for the full VBUS range defined by the USB specification. An external signal level shifter is required to convert the VBUS signal range to a range suitable for the USB_VBUS pin.

32.4.6.4 USB D+ Pullup Enable for USB Function Controller

The USB_PUEN pin is driven high when the pullup is active and is driven low when the pullup is inactive. In this mode of operation, the pullup resistor can be connected directly between the USB_PUEN and the D+ signal. Some transceivers such as the TUSB1105 from Texas Instruments have an additional input pin called SOFTCON. When this pin is driven High, the D+ gets pulled up through an external resistor to 3.3V. The USB_PUEN can be connected to the SOFTCON pin in this case.

32.4.6.5 USB D+, D- Pulldown for USB Function Controller

System implementations that use the USB function controller and are sensitive to supply current requirements can implement weak pulldown resistors on the USB D+ and D- signals associated with the USB Type-B receptacle. When there is no host controller attached upstream of the USB Type-B receptacle, the undriven D+ and D- wires can float to voltages that cause excessive current consumption by the USB transceiver. Weak pulldowns can help prevent this problem. Selection of pulldown resistors depends on transceiver characteristics, D+ pullup resistor implementation, and board layout, and must be designed to meet USB D+ and D- signal requirements.

eFuse Controller

This chapter describes the eFuse controller.

Topic **Page**

33.1 Overview 1944
33.2 Introduction 1944
33.3 eFuse Controller Testing 1944
33.4 eFuse Controller Registers 1947

33.1 Overview

Electrically programmable fuses (eFuses) are used to configure the device after deassertion of PORRST. The eFuse values are read and loaded into internal registers as part of the power-on-reset sequence. The eFuse values are protected with single bit error correction, double bit error detection (SECDED) codes. These fuses are programmed during the initial factory test of the device. The eFuse controller is designed so that the state of the eFuses cannot be changed once the device is packaged.

33.2 Introduction

The eFuse controller automatically reads the values of the eFuses and shifts them into registers during the power-on reset sequence. No action is required from the application code. However, in a safety critical application, the user code should check to see if a correctable or an uncorrectable error was detected during the reset sequence and then preform a self-test on the eFuse controller ECC logic.

33.3 eFuse Controller Testing

33.3.1 eFuse Controller Connections to ESM

There are three connections from the eFuse controller to the Error Signaling Module (ESM). If an uncorrectable error occurs during the loading of the eFuse values after reset, a group three, channel one error and a group one channel 40 error are sent to the ESM. The group three error will cause the ERROR pin to go low. If during the eFuse loading a correctable error occurs, only a group one channel 40 error is sent to the ESM. If an error occurs during the eFuse controller self test, then a group one channel 41 error and a group one channel 40 error are sent to the ESM. After reset, by default, the group one errors do not affect the ERROR pin. If the software enables the appropriate bit in the appropriate ESM Influence Error Pin Set/Status Register (ESMIEPSRn) while the group one error is set, the ERROR pin will go low.

Table 33-1. ESM Signals Set by eFuse Controller

ESM Signal	Uncorrected Load Failure	Correctable Load Error	Self Test		
			eFuse Self Test	eFuse stuck at 0 Test	
				Version a: with Error pin	Version b: without Error pin
Group 3 Channel 1	X			X	
Group 1 Channel 40	X	X	X		
Group 1 Channel 41			X	X	X

33.3.2 Checking for eFuse Errors After Power Up

For safety critical systems, it is required that you check the status of the eFuse controller after a device reset. A suggested flow chart for checking the eFuse controller after device reset is shown in Figure 33-1. Failures during the eFuse self test can be grouped into three levels of severity. Depending on the safety critical application, the error handling for each error type may be different.

33.3.2.1 Class 1 Error

A class 1 error of the eFuse controller means that there was a failure during the autoload sequence. The values read from the eFuses cannot be relied on. All device operation is suspect. A class 1 error is indicated by a signal to group 3 channel 1 of the ESM. This will cause the ERROR pin to go active low.

33.3.2.2 Class 2 Errors

A class 2 error is an indication that the safety checks of the eFuse controller did not work. These are also serious errors because you can no longer guarantee that a more severe error did not occur.

33.3.2.3 Class 3 Error

A class 3 error indicates that there was a single bit failure reading the eFuses that was corrected by ECC bits. Proper operation is still likely, but the system is now at a higher risk for a future non-correctable error. When a correctable error occurs, ESM group 1, channel 40 will be set. In the suggested flow chart shown in Figure 33-1 below, the single bit error is determined by directly reading the eFuse error status register, and not depending on the integrity of the connections between the eFuse controller and the ESM.

33.3.2.4 Stuck at Zero Test

The purpose of the stuck at zero test is to verify that the eFuse controller could signal the ESM if an autoload error did occur. It basically verifies the path through the eFuse controller and to the ESM. This is done by writing a special instruction to the eFuse controller boundary register, then verifying that the proper bits are set in the eFuse controller pins register. Upon successful completion of this test ESM group 1 channel 41 and ESM group 3 channel 1 will be set. This will force the $\overline{\text{ERROR}}$ pin low.

- Version A
  - Write boundary register (address 0xFFF8C01C) with 0x003FC000 to set the error signals.
  - Read pins register (address 0xFFF8C02C) and verify that bits 14, 12, 11 and 10 are set.
  - Write boundary register (address 0xFFF8C01C) with 0x003C0000, to clear the error signals.
  - Verify that ESM group 1 channel 41 and group 3 channel 1 are set, then clear them.

If the system cannot support a test which causes the ERROR pin to go low, then the stuck at zero test can be modified as follows:

- Version B
  - Write boundary register (address 0xFFF8C01C) with 0x003BC000.
  - Read pins register (address 0xFFF8C02C) and verify that bits 14, 12, and 11 are set.
  - Write boundary register (address 0xFFF8C01C) with 0x003C0000, to clear the error signals.
  - Verify that ESM group 1 channel 41 is set, then clear it.

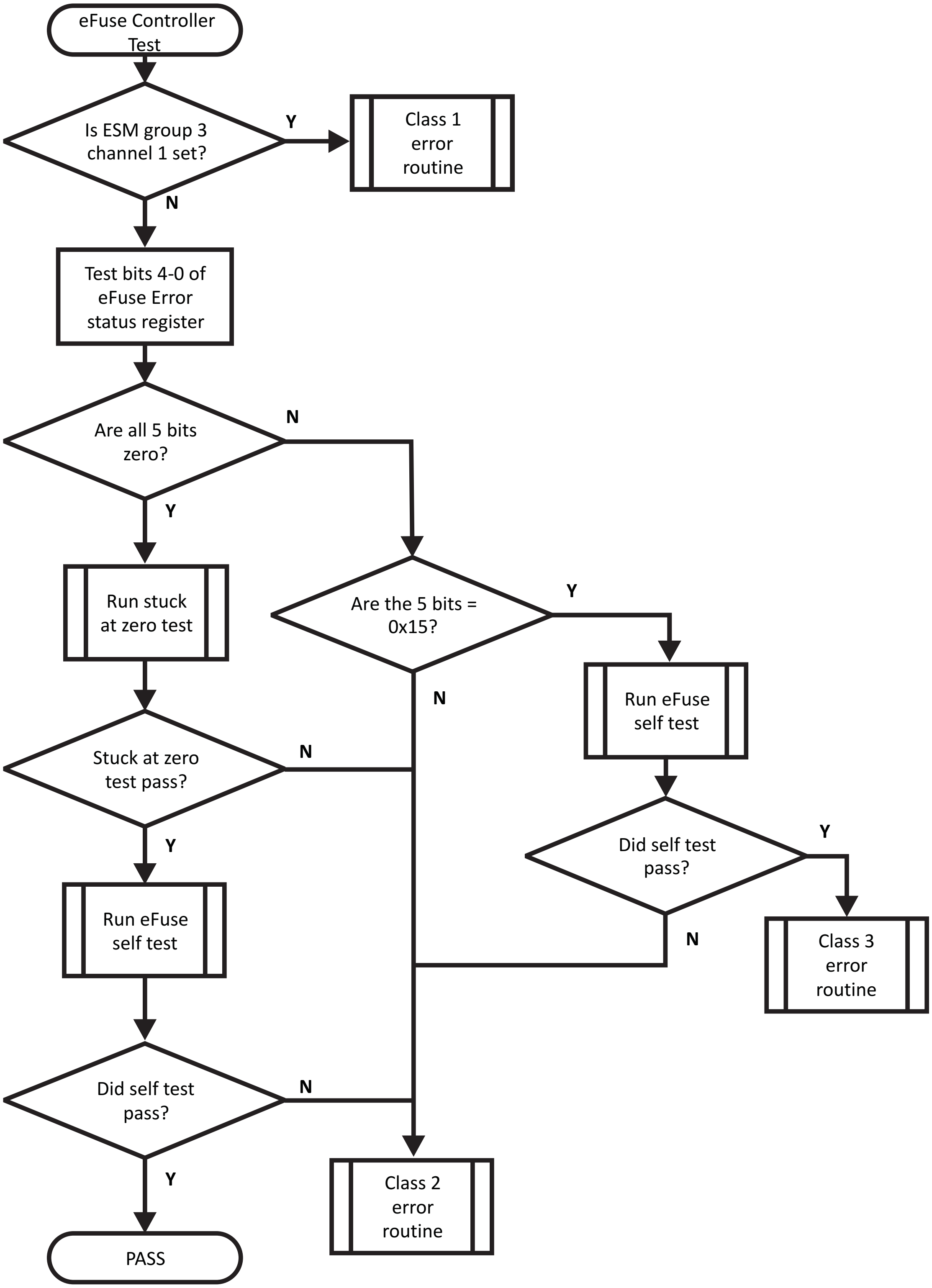
This alternate method provides less test coverage because the path from the uncorrectable error signal from the eFuse controller to the ESM is not specifically tested. However, even if this path is broken, reading the five eFuse error status bits will indicate that an error occurred.

33.3.2.5 eFuse ECC Logic Self Test

The eFuse controller self test performs extensive validation of the ECC logic in the eFuse controller. This test should only be performed once for every device $\overline{\text{PORRST}}$ cycle. Perform the self test by following these steps:

- Write 0x00000258 to the self test cycles register (EFCSTCY) at address 0xFFF8C048.
- Write 0x5362F97F to the self test signature register (EFCSTSIG) at address 0xFFF8C04C.
- Write 0x0000200F to the boundary register at address 0xFFF8C01C. This triggers the self test. The test takes 610 VCLK cycles to complete. The application can poll bit 15 of the pins register at address 0xFFF8C02C to wait for the test to complete.
- Check ESM group 1 channels 40 and 41 for any errors, neither should be set.
- Verify that bits 4 to 0 of the eFuse Error Status register at address 0xFFF8C03C are zero.

Figure 33-1. eFuse Self Test Flow Chart



33.4 eFuse Controller Registers

All registers in the eFuse Controller module are 32-bit, word-aligned; 8-bit, 16-bit and 32-bit accesses are allowed. Table 33-2 provides a quick reference to each of these registers. Specific bit descriptions are discussed in the following subsections. The base address for the control registers is FFF8 C000h.

Table 33-2. eFuse Controller Registers

Offset	Acronym	Register Description	Section
1Ch	EFCBOUND	EFC Boundary Control Register	Section 33.4.1
2Ch	EFCPINS	EFC Pins Register	Section 33.4.2
3Ch	EFCERRSTAT	EFC Error Status Register	Section 33.4.3
48h	EFCSTCY	EFC Self Test Cycles Register	Section 33.4.4
4Ch	EFCSTSIG	EFC Self Test Signature Register	Section 33.4.5

33.4.1 EFC Boundary Control Register (EFCBOUND)

Figure 33-2 and Table 33-3 describe the EFCBOUND register. The eFuse Boundary Control Register is used to test the connections between the eFuse controller and the ESM module. The eFuse Boundary Control Register is also used to initiate an eFuse controller ECC self-test.

Figure 33-2. EFC Boundary Control Register (EFCBOUND) [offset = 1Ch]

31	24
Reserved	
R-0	

23	22	21	20	19	18	17	16
Reserved		EFC Self Test Error	EFC Single Bit Error	EFC Instruction Error	EFC Autoload Error	Self Test Error OE	Single Bit Error OE
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15	14	13	12	8
Instruction Error OE	Autoload Error OE	EFC ECC Selftest Enable	Reserved	
R/W-0	R/W-0	R/W-0	R-0	

7	4	3	0
Reserved		Input Enable	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after power-on reset (nPORRST)

Table 33-3. EFC Boundary Register (EFCBOUND) Field Descriptions

Bit	Field	Value	Description
31-22	Reserved	0	Read returns 0. Writes have no effect.
21	EFC Self Test Error		This bit drives the self test error signal when bit 17 (Self Test Error OE) is high. This signal is attached to ESM error Group 1, Channel 41.
		0	Drives the self test error signal low, if Self Test OE is high.
		1	Drives the self test error signal high, if Self Test OE is high.
20	EFC Single Bit Error		This bit drives the single bit error signal when bit 16 (Single bit Error OE) is high. This signal is attached to ESM error Group 1, Channel 40.
		0	Drives the self test error signal low, if Single Bit Error OE is high.
		1	Drives the self test error signal high, if Single Bit Error OE is high.

Table 33-3. EFC Boundary Register (EFCBOUND) Field Descriptions (continued)

Bit	Field	Value	Description
19	EFC Instruction Error		This bit drives the instruction error signal when bit 15 (Instruction Error OE) is high. This signal is used to denote an error occurred during e-fuse programming. This signal is not attached to the ESM.
		0	Drives the Instruction Error signal low, if Instruction Error OE is high.
		1	Drives the Instruction Error signal high, if Instruction Error OE is high.
18	EFC Autoload Error		This bit drives the Autoload Error signal when bit 14 (Autoload Error OE) is high. This signal is attached to ESM error Group 3, Channel 1.
		0	Drives the Autoload Error signal low, if Autoload Error OE is high.
		1	Drives the Autoload Error signal high, if Autoload Error OE is high.
17	Self Test Error OE		The Self Test Error Output Enable bit determines if the EFC Self Test signal comes from the eFuse controller or from bit 21 of the boundary register.
		0	EFC Self Test Error comes from eFuse controller.
		1	EFC Self Test Error comes from the boundary register.
16	Single Bit Error OE		The single bit error output enable signal determines if the EFC Single Bit Error signal comes from the eFuse controller or from bit 20 of the boundary register.
		0	EFC Single Bit Error comes from eFuse controller.
		1	EFC Single Bit Error comes from the boundary register.
15	Instruction Error OE		The instruction error output enable signal determines if the EFC Instruction Error signal comes from the eFuse controller or from bit 19 of the boundary register.
		0	EFC Instruction Error comes from eFuse controller.
		1	EFC Instruction Error comes from the boundary register.
14	Autoload Error OE		The autoload error output enable signal determines if the EFC Autoload Error signal comes from the eFuse controller or from bit 18 of the boundary register.
		0	EFC Autoload Error comes from eFuse controller.
		1	EFC Autoload Error comes from the boundary register.
13	EFC ECC Selftest Enable		The eFuse Controller ECC Selftest Enable bit starts the selftest of the ECC logic if the four input enable bits (EFCBOUND[3:0) are all 1s.
		0	No action
		1	Start ECC selftest if EFCBOUND[3:0] are Fh.
12-4	Reserved	0	Read returns 0. Writes have no effect.
3-0	Input Enable		The eFuse Controller ECC Selftest Enable bit starts the selftest of the ECC logic if the four input enable bits (EFCBOUND[3:0) are all 1s.
		Fh	ECC selftest can be started if EFC ECC Selftest Enable, bit 13, is set
		All others	ECC selftest cannot be started.

33.4.2 EFC Pins Register (EFCPINS)

Figure 33-3 and Figure 33-3 describe the EFCPINS register.

Figure 33-3. EFC Pins Register (EFCPINS) [offset = 2Ch]

31–16
Reserved
R-0

15	14	13	12	11	10	9–8
EFC Selftest Done	EFC Selftest Error	Reserved	EFC Single Bit Error	EFC Instruction Error	EFC Autoload Error	Reserved
R-0	R-0	R-0	R-x	R-0	R-x	R-x

7–0
Reserved
R-x

LEGEND: R = Read only; -*n* = value after power-on reset (nPORRST); x = Indeterminate

Table 33-4. EFC Pins Register (EFCPINS) Field Descriptions

Bit	Name	Value	Description
31–16	Reserved	0	Reads return zeros, writes have no effect.
15	EFC Selftest Done		This bit can be polled to determine when the EFC ECC selftest is complete
		0	EFC ECC selftest is not complete.
		1	EFC ECC selftest is complete.
14	EFC Selftest Error		This bit indicates the pass/fail status of the EFC ECC Selftest once the EFC Selftest Done bit (bit 15) is set.
		0	EFC ECC Selftest passed.
		1	EFC ECC Selftest failed.
13	Reserved	0	Reads return zeros. Do NOT write a 1 to this bit.
12	EFC Single Bit Error		This bit indicates if a single bit error was corrected by the ECC logic during the autoload after reset.
		0	No single bit error was detected.
		1	A single bit error was detected and corrected.
11	EFC Instruction Error		This bit indicates an error occurred during a factory test or program operation. This bit should not be set from normal use.
		0	No instruction error detected.
		1	An error occurred during a factory test or program operation.
10	EFC Autoload Error		This bit indicates that some non-correctable error occurred during the autoload sequence after reset. This bit also sets ESM group 3, channel 1.
		0	The autoload function completed successfully.
		1	There were non-correctable errors during the autoload sequence.
9-0	Reserved	0-1	After reset, these bits are indeterminate and reads return either a 1 or 0.

33.4.3 EFC Error Status Register (EFCERRSTAT)

Figure 33-4 and Table 33-5 describe the EFCERRSTAT register.

Figure 33-4. EFC Error Status Register (EFCERRSTAT) [offset = 3Ch]

31 … 8
Reserved
R-0

7 6	5	4 … 0
Reserved	Instruc Done	Error Code
R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after power-on reset (nPORRST)

Table 33-5. EFC Error Status Register (EFCERRSTAT) Field Descriptions

Bit	Name	Value	Description
31–6	Reserved	0	Reads return zeros, writes have no effect.
5	Instruc Done		Instruction done. Used to indicate that the eFuse self test has completed
		0	The eFuse controller is still executing.
		1	The eFuse controller has completed executing.
4-0	Error Code		The error status of the last instruction executed by the eFuse Controller
		0	No error.
		5h	An uncorrectable (multibit) error was detected during the power-on autoload sequence.
		15h	At least one single bit error was detected and corrected during the power-on autoload sequence.
		18h	The signature generated by the ECC self-test logic did not match the golden signature written in the EFCSTSIG register. The EDAC circuitry might have a fault.
		All other values	All other values are reserved for e-fuse system tests and are not expected to occur in normal system use.

33.4.4 EFC Self Test Cycles Register (EFCSTCY)

Figure 33-5 and Table 33-6 describe the EFCSTCY register.

Figure 33-5. EFC Self Test Cycles Register (EFCSTCY) [offset = 48h]

31 … 16
Cycles
R/W-0

15 … 0
Cycles
R/W-0

LEGEND: R/W = Read/Write; -*n* = value after power-on reset (nPORRST)

Table 33-6. EFC Self Test Cycles Register (EFCSTCY) Field Descriptions

Bit	Name	Description
31–0	Cycles	This register is used to determine the number of cycles to run the eFuse controller ECC logic self test. It is recommended to use a value of 600 (0x00000258).

33.4.5 EFC Self Test Signature Register (EFCSTSIG)

Figure 33-6 and Table 33-7 describe the EFCSTSIG register.

Figure 33-6. EFC Self Test Cycles Register (EFCSTSIG) [offset = 4Ch]

31	16
Signature	
R/W-0	

15	0
Signature	
R/W-0	

LEGEND: R/W = Read/Write; -*n* = value after power-on reset (nPORRST)

Table 33-7. EFC Self Test Cycles Register (EFCSTSIG) Field Descriptions

Bit	Name	Description
31–0	Signature	This register is used to hold the expected signature for the eFuse ECC logic self test. It is recommended to write a value of 0x5362F97F to this register and a value of 600 (0x00000258) to the EFCSTCY register. If after running the eFuse ECC logic self test, the calculated signature does not match the expected signature in the EFCSTSIG register, then a value of 18h is stored in the EFCERRSTAT register.

Revision History

Changes from April 17, 2015 to February 28, 2018 **Page**

- **Chapter 1: Introduction** 95
- **Chapter 2: Architecture** 100
- Table 2-2: Changed table. Added Access Mode column 104
- Table 2-3: Updated Actual Size column for Flash Bus2 Interface: Customer OTP–ECC and TI OTP–ECC EEPROM Banks 106
- Table 2-6: Changed table. Corrected values in Valid RAM Groups and Valid RINFO Register Value columns 113
- Table 2-7: Changed NHET2 RAM Address Range End to 0xFF45FFFF 116
- Table 2-30: Changed Description of bits for Value = 0 (Read) to enabled 139
- Section 2.5.1.13: Added second paragraph to NOTE 140
- Table 2-34: Changed Description of GHVSRC bit. Removed "on wakeup" 146
- Table 2-40: Updated MSTGENA and MINITGENA values to Ah for MSIENA = 1 152
- Table 2-44: Corrected Description of PLLMUL bit. Value = 0h is ×1, Value = 100h is ×2 156
- Table 2-53: Changed Description of PLL1_FBSLIP_FILTER_ COUNT and PLL1_FBSLIP_FILTER_ KEY bits 167
- Section 2.5.1.44: Changed paragraph 174
- Section 2.5.1.44: Added NOTE 174
- Section 2.5.1.45: Changed NOTE 176
- Table 2-63: Added Note to VCLK2R and VCLKR bits 176
- Section 2.5.1.48: Added NOTE 178
- Figure 2-53: Changed Reserved bits to 2-0 179
- Figure 2-53: Deleted MPMODE bit 179
- Table 2-67: Changed Description of WDRST bit. Added (DBGRST) 179
- Table 2-67: Changed Reserved bits to 2-0 179
- Table 2-67: Deleted MPMODE bit 179
- Table 2-77: Changed Description of VCLKA4S bit for Value = 8h-Fh 188
- Table 2-78: Changed Description of PLL1_SLIP_FILTER_COUNT and PLL1_SLIP_FILTER_KEY bits 190
- Section 2.5.2.7: Changed paragraph 191
- Figure 2-65: Corrected register bit fields 191
- Table 2-80: Changed table to reflect updated register bit fields 191
- Section 2.5.2.8: Changed paragraph 192
- Figure 2-66: Corrected register bit fields 192
- Table 2-81: Changed table to reflect updated register bit fields 192
- Table 2-89: Corrected register names in Description of PROTSET bit for Value = 1 (Write) 197
- Table 2-90: Corrected register names in Description of PROTSET bit for Value = 1 (Write) 198
- Table 2-91: Corrected register names in Description of PROTSET bit for Value = 1 (Write) 198
- Table 2-92: Corrected register names in Description of PROTSET bit for Value = 1 (Write) 199
- Table 2-93: Corrected register names in Description of PROTCLR bit for Value = 1 (Write) 199
- Table 2-94: Corrected register names in Description of PROTCLR bit for Value = 1 (Write) 200
- Table 2-95: Corrected register names in Description of PROTCLR bit for Value = 1 (Write) 200
- Table 2-96: Corrected register names in Description of PROTCLR bit for Value = 1 (Write) 201
- **Chapter 3: Power Management Module (PMM)** 209
- **Chapter 4: I/O Multiplexing and Control Module (IOMM)** 239
- Table 4-7: Corrected bit value in middle column to = 1 250
- **Chapter 5: F021 Flash Module Controller** 267
- Section 5.1.2: Added definition for ATCM 268
- Table 5-1: Updated format 271
- Figure 5-1: Changed figure 274
- Section 5.6.2: Updated second paragraph 278
- Table 5-8: Updated Description of test modes 278
- Section 5.6.2.1: Updated eighth sentence in second paragraph. Changed ECC_MUL_ERR to ERR_PRF_FLG 279

- Section 5.6.2.4: Updated second pargarph ... 280
- Section 5.6.2.5: Added NOTE ... 281
- Section 5.6.2.5: Updated second paragraph ... 281
- Section 5.6.2.6: Updated first sentence in fourth paragraph. Changed ECC_MUL_ERR to B1_UNC_ERR ... 282
- Section 5.6.2.6: Updated steps in fifth paragraph ... 282
- Table 5-9: Updated Name of test modes ... 283
- Table 5-9: Updated test mode 7. Changed ECC_MUL_ERR to B1_UNC_ERR ... 283
- Table 5-10: Updated Name of test modes ... 283
- Table 5-10: Updated test mode 1. Deleted D_MUL_ERR ... 283
- Table 5-10: Updated test mode 1. Changed ERR_MUL_FLG to ERR_PRF_FLG ... 283
- Table 5-10: Updated test mode 1. Deleted EE_D_MUL_ERR ... 283
- Table 5-10: Updated test mode 1. Changed EE_ERR_MUL_FLG to EE_ERR_PRF_FLG ... 283
- Table 5-10: Updated test mode 3. Changed EE_MAL_ERR to EE_CME ... 283
- Table 5-10: Updated test mode 4. Changed EE_COM_MAL_GOOD to EE_CMG and EE_MAL_ERR to EE_CME ... 283
- Table 5-11: Updated Name of test modes ... 284
- Table 5-14: Changed Description of EDACMODE bit ... 287
- Table 5-24: Changed Description of PROTL1DIS bit ... 298
- Section 5.7.22: Added paragraph ... 307
- Table 5-34: Changed Description of EMU_DMSW bit ... 307
- Section 5.7.23: Added paragraph ... 307
- Table 5-35: Changed Description of EMU_DLSW bit ... 307
- Section 5.7.24: Added paragraph ... 308
- Table 5-36: Changed Description of EMU_ECC bit ... 308
- Section 5.7.25: Added paragraph ... 309
- Table 5-37: Changed Description of EMU_ADDR bit ... 309
- Section 5.7.26: Deleted second paragraph ... 310
- Table 5-38: Updated Description of DIAG_MODE bit ... 310
- Section 5.7.30: Deleted paragraph ... 314
- Section 5.7.35: Changed paragraph ... 318
- Table 5-47: Changed Description of EE_EDACMODE bit ... 318
- Table 5-52: Changed Description of EE_CMG bit ... 323
- Table 5-52: Added Description for bit 4, EE_CME ... 323
- **Chapter 6: Tightly-Coupled RAM (TCRAM) Module** ... 326
- Section 6.4: Updated paragraph to include INIT_DOMAIN register ... 330
- Table 6-1: Added Power Domain Enable Register (INIT_DOMAIN REG) ... 332
- Section 6.7.11: Added subsection ... 346
- **Chapter 7: Programmable Built-In Self-Test (PBIST) Module** ... 347
- Section 7.3.1: Changed step 6. Deleted ROM interface clock ... 351
- Section 7.3.1: Changed step 12. Deleted ROM clock ... 351
- Section 7.5: Changed second paragraph (write 1h) ... 354
- Section 7.6.1: Changed step 5. Deleted ROM interface clock ... 367
- Section 7.6.1: Changed step 12. Deleted ROM clock ... 367
- Section 7.6.2: Changed step 5. Deleted ROM interface clock ... 368
- Section 7.6.2: Changed step 11. Deleted ROM clock ... 368
- **Chapter 8: CPU Self-Test Controller (STC) Module** ... 369
- **Chapter 9: CPU Compare Module for Cortex-R4F (CCM-R4F)** ... 385
- Section 9.2: Added last sentences about the signals compared by the CCM-R4F ... 386
- **Chapter 10: Oscillator and PLL** ... 393
- Section 10.5: Updated third paragraph. Changed $f_{(HCLK)}$ to $f_{(GCLK)}$ ... 401
- Table 10-1: Updated Frequency Limit value to $f_{(GCLK)}$ for $f_{PLL\ CLK}$ ... 401
- Table 10-2: Updated formula for NF ... 402
- Section 10.5.4: Added last sentence to step 3 in both paragraphs ... 408
- Section 10.8: Changed step 3, second sentence ... 417

- **Chapter 11: Dual-Clock Comparator (DCC) Module** 419
- Table 11-8: Corrected table title 431
- Table 11-11: Corrected table title 433
- Table 11-11: Changed Description of KEY bit for Value = Any other value 433
- **Chapter 12: Error Signaling Module (ESM)** 435
- Section 12.1.2: Changed first paragraph 436
- Table 12-14: Changed Description of INTOFFL bit for values 21h to 60h 450
- **Chapter 13: Real-Time Interrupt (RTI) Module** 457
- Equation 24: Corrected first eqution (if RTICPUCy ≠ 0) 461
- Figure 13-18: Updated Read/Write value of bits to R/WP-0 475
- Figure 13-18: Updated LEGEND to include WP 475
- **Chapter 14: Cyclic Redundancy Check (CRC) Controller Module** 499
- **Chapter 15: Vectored Interrupt Manager (VIM) Module** 538
- Section 15.4: Added NOTE 546
- Figure 15-13: Corrected reset value of Interrupt Vector Table offset bits 15-9 555
- Table 15-4: Corrected Description of Interrupt Vector Table offset bits. Reads are always FFF8 2xxxh 555
- **Chapter 16: Direct Memory Access Controller (DMA) Module** 574
- Chapter 16: Global: Changed index pointer to offset value 574
- Section 16.2.2: Changed second bullet 576
- Figure 16-5: Updated figure (changed Index Pointer to Offset Value) 578
- Section 16.2.7: Added last two paragraphs 586
- Table 16-2: Added table. Subsequent tables renumbered 586
- Section 16.2.12Changed fourth paragraph 590
- Table 16-21: Updated Value column for all bits. Added 20h-3Fh = Reserved 604
- Table 16-22: Updated Value column for all bits. Added 20h-3Fh = Reserved 605
- Table 16-23: Updated Value column for all bits. Added 20h-3Fh = Reserved 606
- Table 16-24: Updated Value column for all bits. Added 20h-3Fh = Reserved 607
- Table 16-33: Changed Description of LFSINTENA bit = 1h to channel is enabled 613
- Table 16-38: Changed Description of BTCINTDIS bit for Value = 0 (Read). Channel is disabled 615
- Table 16-38: Changed Description of BTCINTDIS bit for Value = 1 (Read). Channel is enabled 615
- Figure 16-48: Updated Read/Write value of bits to R/W1CP-0 616
- Figure 16-49: Updated Read/Write value of bits to R/W1CP-0 617
- Figure 16-50: Updated Read/Write value of bits to R/W1CP-0 617
- Figure 16-51: Updated Read/Write value of bits to R/W1CP-0 618
- Table 16-44: Updated Value column of FTCA bit. Added 11h-3Fh = Reserved 619
- Table 16-45: Updated Value column of LFSA bit. Added 11h-3Fh = Reserved 619
- Table 16-46: Updated Value column of HBCA bit. Added 11h-3Fh = Reserved 621
- Table 16-47: Updated Value column of BTCA bit. Added 11h-3Fh = Reserved 621
- Table 16-48: Updated Value column of FTCB bit. Added 11h-3Fh = Reserved 623
- Table 16-49: Updated Value column of LFSB bit. Added 11h-3Fh = Reserved 623
- Table 16-50: Updated Value column of HBCB bit. Added 11h-3Fh = Reserved 625
- Table 16-51: Updated Value column of BTCB bit. Added 11h-3Fh = Reserved 625
- Figure 16-71: Updated Read/Write value of bits to R/W1C-0 637
- Figure 16-71: Updated LEGEND to include W1C 637
- Table 16-63: Changed Description of bits. Added 0 (Write) = No effect 637
- Table 16-75: Changed Description of TTYPE bit. (A request triggers) 644
- **Chapter 17: External Memory Interface (EMIF)** 648
- Figure 17-18: Updated reset value of RR bit to 60h 686
- **Chapter 18: Parameter Overlay Module (POM)** 704
- Figure 18-3: Updated LEGEND to include WP 709
- Figure 18-6: Updated Read/Write value of TO bit to R/W1CP-0 711
- Figure 18-6: Updated LEGEND to include W1CP 711
- Table 18-5: Changed Description of TO bit 711

- Figure 18-7: Updated LEGEND to include WP 712
- **Chapter 19: Enhanced Pulse Width Modulator (ePWM) Module** 724
- **Chapter 20: Enhanced Capture (eCAP) Module** 851
- **Chapter 21: Enhanced Quadrature Encoder Pulse (eQEP) Module** 885
- **Chapter 22: Analog To Digital Converter (ADC) Module** 934
- Figure 22-2: Corrected Event Trigger Generation signal 936
- Figure 22-2: Corrected EXT_SEL and EXT_ENA signals 936
- Figure 22-2: Deleted EXT_nENA signal 936
- Section 22.2.3: Updated third and fifth sentences in third paragraph 937
- Section 22.3.3: Added last sentence (reference) to fourth paragraph 939
- Figure 22-10: Corrected bit range of the EV_CURRENT_COUNT, EV_MAX_COUNT, G1_CURRENT_COUNT, G1_MAX_COUNT, G2_CURRENT_COUNT, and G2_MAX_COUNT bits to 4-0 946
- Table 22-12: Changed Description of FRZ_EV bit. (The Event Group conversion is kept frozen while the Group1 or Group2 conversion is active,) 976
- Table 22-13: Changed Description of FRZ_G1 bit. (The Group1 conversion is kept frozen while the Event Group or Group2 conversion is active,) 979
- Table 22-14: Changed Description of FRZ_G2 bit. (The Group2 conversion is kept frozen while the Event Group or Group1 conversion is active,) 982
- Figure 22-52: Changed RESERVED bits to 31-24 and G2_SEL bits to 23-0 1013
- Table 22-40: Changed RESERVED bits to 31-24 and G2_SEL bits to 23-0 1013
- **Chapter 23: High-End Timer (N2HET) Module** 1071
- Section 23.2.5.4: Changed first sentence 1091
- Table 23-9: Changed Pull Control and Input Buffer = Enabled when device is under reset 1103
- Section 23.2.9: Updated first paragraph 1108
- Section 23.3.2: Added Hardware Angle Generator (HWAG) subsection. Subsequent figures and tables renumbered 1113
- Table 23-26: Changed Description of HETPRY bit 1144
- Section 23.5: Added HWAG Registers section. Subsequent section, figures, and tables renumbered 1162
- Section 23.6.3.8: Updated Description of CNT instruction. The data field [D31:7] is incremented unconditionally on each execution of the instruction 1204
- Section 23.6.3.19: Updated Description of RCNT instruction. For example, choosing M = 100 allows the input period to be expressed as a percentage (%) of the reference period 1236
- **Chapter 24: High-End Timer Transfer Unit (HTU) Module** 1249
- Section 24.2: Updated third bullet in fifth paragraph. Added per request 1252
- Section 24.2: Added fourth bullet in fifth paragraph 1252
- Section 24.2.4.1: Updated first sentence in third paragraph 1259
- Section 24.2.4.1: Updated fourth sentence in seventh paragraph. If the signal frequency would increase, then a wrong pair [22,23] could be read 1260
- Figure 24-13: Changed position of third rising edge in waveform 1263
- Table 24-24: Corrected bit range of the Reserved bit to 31-10 and the INTTYPE0 bit to 9-8 1276
- Table 24-25: Corrected bit range of the Reserved bit to 31-10 and the INTTYPE1 bit to 9-8 1277
- Section 24.5.7: Updated paragraph 1298
- Table 24-49: Corrected table title 1298
- **Chapter 25: General-Purpose Input/Output (GIO) Module** 1301
- Table 25-22: Changed Pull Control = Enabled when device is under reset 1327
- **Chapter 26: Controller Area Network (DCAN) Module** 1328
- Section 26.3.2.3: Changed t_q = 1 µs 1335
- Section 26.16: Changed subsection 1363
- Table 26-6: Added Core Release Register 1363
- Table 26-9: Updated Description of REC bit (Values from 0 to 127) 1369
- Table 26-11: Corrected Value column range of Int1ID and Int0ID bits to 1h-40h 1371
- Table 26-13: Corrected Value column range of Message Number bit to 1h-FFh 1373
- Table 26-13: Updated Description of Message Number bit. Only values 1h-40h are valid. Values 41h-FFh are invalid 1373
- Section 26.17.8: Added subsection. Subsequent subsections, figures, and tables renumbered 1373
- Table 26-21: Corrected Value column range of Message Number bit to 1h-40h 1384

- Section 26.17.24: Changed sixth paragraph 1391
- Table 26-31: Changed the Func Bit description for Value = 1. (as an input to receive CAN data) 1399
- **Chapter 27: Multi-Buffered Serial Peripheral Interface Module (MibSPI) with Parallel Pin Option (MibSPIP)** 1401
- Chapter 27: Global: Changed $\overline{\text{SPISCS}}$ to $\overline{\text{SPICS}}$ 1401
- Section 27.2: Changed first sentence in second paragraph 1403
- Section 27.2: Added NOTE 1403
- Table 27-1: Changed SPIENA enabled description in Slave Mode 1404
- Figure 27-6: Corrected figure title 1410
- Figure 27-6: Corrected bits D11-D8 to 1110 1410
- Figure 27-7: Corrected figure title 1410
- Section 27.2.9: Updated first two paragraphs 1414
- Section 27.8.1: Changed second sentence 1430
- Section 27.8.1: Changed second sentence 1431
- Section 27.9: Global: Updated all VBUSPCLK signals to VCLK 1432
- Table 27-9: Changed Description of CLKMOD bit for Value = 1. ($\overline{\text{SPIENA}}$ is an input.) 1434
- Table 27-14: Corrected Description of SIMODIR0 bit 1442
- Section 27.9.16: Added NOTE 1455
- Table 27-23: Updated Description of CSNR and TXDATA bits 1455
- Table 27-24: Added table. Subsequent tables renumbered 1457
- Table 27-25: Corrected Description of RXEMPTY bit. (SPIBUF to RXDATA) 1458
- Table 27-27: Corrected Value column for all bits to 0-FFh 1460
- Table 27-27: Updated Description of C2TDELAY bit. Deleted first NOTE 1460
- Table 27-35: Updated bit Descriptions to clarify register bit number corresponding to TG number 1474
- Table 27-36: Updated bit Descriptions to clarify register bit number corresponding to TG number 1475
- Table 27-36: Changed Description of CLRINTENRDY and CLRINTENSUS bits for Value = 1 (Write). The interrupt does not get generated 1475
- Table 27-37: Updated bit Descriptions to clarify register bit number corresponding to TG number 1476
- Table 27-38: Updated bit Descriptions to clarify register bit number corresponding to TG number 1477
- Table 27-38: Changed Description of CLRINTLVLRDY and CLRINTLVLSUS bits for Value = 1 (Write). Clear the TGx interrupt INT0 1477
- Table 27-39: Updated bit Descriptions to clarify register bit number corresponding to TG number 1478
- Table 27-40: Changed Description of TICKENA bit for Value = 1. Deleted second sentence 1479
- Table 27-41: Changed Note in LPEND bit 1480
- Figure 27-74: Changed format 1494
- Figure 27-75: Changed format 1496
- Section 27.10.3: Added NOTE 1500
- Table 27-55: Updated Description of CSHOLD and TXDATA bits 1500
- Table 27-56: Added table. Subsequent tables renumbered 1502
- Table 27-57: Corrected bits descriptions (SPIBUF to RXRAM) 1503
- **Chapter 28: Serial Communication Interface (SCI)/Local Interconnect Network (LIN) Module** 1512
- Section 28.4.1: Changed third paragraph 1529
- Section 28.4.2: Changed second sentence in second paragraph 1529
- Section 28.5: Updated both paragraphs 1530
- Section 28.5: Updated procedure in second paragraph 1530
- Section 28.5.1.1: Updated first and last paragraphs 1530
- Section 28.5.1.2: Updated paragraph 1531
- Section 28.5.2.1: Updated first and last paragraphs 1531
- Section 28.5.2.1: Changed number 2 in second paragraph to Transmit Interrupt 1531
- Section 28.5.2.2: Updated paragraph 1531
- Section 28.10: Updated paragraph 1552
- Section 28.10: Updated procedure 1552
- Section 28.10.1.1: Updated first and last paragraphs 1552
- Section 28.10.1.2: Updated paragraph 1553
- Section 28.10.2.1: Updated first and last paragraphs 1553

- Section 28.10.2.1: Changed number 2 in second paragraph to Transmit Interrupt ... 1553
- Section 28.10.2.2: Updated paragraph ... 1554
- Table 28-27: Corrected bit name in Note to SET TX INT ... 1590
- Section 28.14.2: Changed first bullet ... 1606
- Table 28-48: Changed Pull Control = Enabled when device is under reset ... 1607
- **Chapter 29: Serial Communication Interface (SCI) Module** ... 1608
- Equation 57: Updated equation. Changed VBUSPCLK to VCLK ... 1613
- Equation 58: Updated equation. Changed VBUSPCLK to VCLK ... 1613
- Section 29.5: Updated both paragraphs ... 1620
- Section 29.5: Updated procedure in second paragraph ... 1620
- Section 29.5.1: Updated last paragraph ... 1620
- Section 29.5.2: Updated last paragraph ... 1621
- Section 29.5.2: Changed number 2 in second paragraph to Transmit Interrupt ... 1621
- Equation 59: Updated equation. Changed VBUSPLCK to VCLK ... 1641
- Equation 60: Updated equation. Changed VBUSPLCK to VCLK ... 1641
- Section 29.8.2: Changed first bullet ... 1653
- Table 29-33: Changed Pull Control = Enabled when device is under reset ... 1654
- **Chapter 30: Inter-Integrated Circuit (I2C) Module** ... 1655
- Table 30-7: Changed fourth paragraph in Description of XSMT bit ... 1674
- Section 30.6.21: Changed paragraph ... 1688
- Section 30.6.22: Changed paragraph ... 1689
- Section 30.6.23: Changed paragraph ... 1689
- Table 30-32: Updated Description of SDAPDR and SCLPDR bits. 0 = enabled; 1 = disabled ... 1689
- Section 30.6.24: Changed paragraph ... 1690
- Section 30.6.25: Changed paragraph ... 1690
- Table 30-35: Changed Pull Control = Enabled when device is under reset ... 1691
- **Chapter 31: EMAC/MDIO Module** ... 1693
- Figure 31-7: Added figure. Subsequent figures renumbered ... 1705
- Figure 31-8: Added figure. Subsequent figures renumbered ... 1706
- Figure 31-9: Added figure. Subsequent figures renumbered ... 1707
- Table 31-68: Changed Description of GMIIEN bit ... 1793
- **Chapter 32: Universal Serial Bus (USB)** ... 1816
- Table 32-57: Corrected signal names (GZO, VP, and VM) ... 1939
- **Chapter 33: eFuse Controller** ... 1943

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated (‘TI”) technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, “TI Resources”) are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI’s provision of TI Resources does not expand or otherwise alter TI’s applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources 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.

TI RESOURCES ARE PROVIDED “AS IS” AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI’s standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265